Entropy-Based Similarity For Efficient Keyword Search In XML Databases

S. Selvaganesan*, Su-Cheng Haw, Lay-Ki Soon
Faculty of Computing and Informatics, Multimedia University, 63100 Cyberjaya, Malaysia

Abstract— With the rapid emergence of XML as a data exchange and data transfer medium over the Web, an effective keyword-based search approach to locate the relevant information is indeed essential. The major challenge with keyword-based search is to ensure that (1) it returns only the relevant results to the query, and (2) how the query results could be ranked properly. In this paper, we propose a two-level indexing scheme based on mutual information. To demonstrate how our proposed indexing works, we use the University Course Listing dataset as an example.

Index Terms—XML, Keyword Search, XML Database, Mutual Information, Entropy, Entropy-Based Similarity.

I. INTRODUCTION

Index structures are well-known to overcome the performance degradation due to excessive join in the relational databases. Similarity, index structures [1], [2], [3], [4] have been introduced in XML to speed up the query processing by minimizing the search space in the XML tree. The keyword-based [5], [6], [7] indexing technique allows users to find information they are interested in without having to know the underlying database schema or complex query language. As a result, research on keyword search in XML database is on demand. A lot of research has been conducted in XML keyword search [6], [8], [9], [10] and in these approaches, the returned answers may be meaningful but they may be irrelevant to user search intention. The main problem of effective XML keyword search is to identify the user search intention accurately in the presence of keyword ambiguities [11]. Hence, our main contribution in this paper is the proposed two-level indexing scheme based on entropy-based similarity. To demonstrate how our proposed indexing works, we use the University Course Listing dataset obtained from the repository of University of Washington [12] as an example.

The rest of the paper is structured as follows. In Section II, we briefly present related work. Section III is the core of the paper where it gives an overview of our proposed approach. In Section IV, we present the algorithm. Lastly, Section V concludes the paper and suggests some future works.

II. RELATED WORK

Sun et al. [13] presented the multiway-SLCA (Smallest Lowest Common Ancestor) approach to process more general keyword search queries to support both AND and OR Boolean operators. Liu and Chen [9] proposed MaxMatch, a novel semantics for identifying relevant matches and efficient algorithm to realize this semantics. Bao et al. [14] proposed an IR-style approach utilizing the statistics of underlying XML data to address the challenges, namely, identification of user search intention, resolving keyword ambiguity problems and estimation of result relevance to a given query. Based on three guidelines proposed in the approach, they designed formulae to identify the search for nodes and search via nodes of a query, and presented a XML TF*IDF ranking strategy to rank the individual matches of all possible search intentions. Their proposed techniques have been implemented in an XML keyword search engine called XReal, and extensive experiments show the effectiveness of their approach. Li et al. [15] designed an adaptive XML keyword search approach, called XBridge, that can derive the semantics of a keyword query and generate a set of effective structured queries by analyzing the given keyword query and the schemas of XML data sources. Li et al. [16] presented an XML keyword search system XKMis, which is based on minimal information segments (MISs) and not on LCA (Lowest Common Ancestor) or its variant. Li and Wang [17] explored the application of query suggestion in XML keyword search and proposed an interactive XML query system XQsuggest.

Bao et al. [18] modeled XML document as interconnected object-trees, based on which they proposed two main matching semantics called Interested Single Object (ISO) and Interested Related Object (IRO), to capture different user search concerns. A customized ranking scheme was proposed by taking both the structure and content of the results into account. They proposed efficient algorithms to compute and rank the query results in one phase. Bao et al. [11] made several updates to [14] as an extension. To complement the result ranking framework, they considered the popularity of the results that have comparable relevance scores. An efficient algorithm based on a new indexing technique was designed to compute the popularity score and more experiments have been conducted to prove the effectiveness of the approach. Inspired by Bao et al. [11], we propose a new approach for efficient XML keyword search, in continuation of [19].

III. PROPOSED APPROACH

A. Background and Our Observations

Bao et al. [11] introduced an IR-style approach utilizing the statistics of XML database to address the problem of XML keyword search, namely, search intention identification, result retrieval and relevance oriented ranking. This approach builds two indices, namely, keyword inverted list and frequency
table. Of these indices, the keyword inverted list retrieves a list of data nodes in document order whose values contain the input keyword. Each inverted list has an index (eg. B+-Tree) on the top and contains the input keyword in the form of a tuple \(<\text{DeweyID}, \text{prefixPath}, f_{ak}, W_c>\). For each keyword in the query, the inverted list returns a set of nodes \(a\) in the document order. Though many inverted lists are very short, one whole disk page may be wasted for indexing a short inverted list. This will cause space requirements and will be expensive in terms of space. In the second index built, namely, frequency table, only the frequency \(f_k^T\), i.e., number of \(T\)-typed nodes that contain keyword \(k\) in their sub trees in XML data, is stored for each combination of keyword \(k\) and node type \(T\) in XML document. For each keyword in the given query, the approach gets the value of \(f_k^T\) without specifying whether the keyword is a tag or a data value. Moreover, the approach does not deal with each tag and data node separately and thereby makes query processing more complex. XML keyword search based on this will be time consuming. In fact, frequencies \(f_{ak}\) and \(f_k^T\) stored in both indices built (inverted list and frequency table) in the approach are not dependent. There will be uncertainty of information between the two frequencies. To resolve the problem of not dealing with tag and data node separately, we propose a new two-level indexing that builds two indices, namely, \(\text{tag}_\text{info}\) table and \(\text{data}_\text{info}\) table, for structural nodes and data nodes respectively in XML database thereby speeding up the query processing. Also, keyword ambiguity problems and space requirement problem will be resolved. To efficiently use the frequency information, we define a new formula for mutual information between keyword matching tags and tags containing data values with respect to XML keyword queries. By utilizing the concept of entropy, we propose an entropy formula to measure the probability of occurrence of keyword matching data value contained in a leaf tag, path-wise. Also, we design an entropy-based similarity formula to evaluate the accuracy of occurrence of query keywords in XML database so as to find relevant path.

**B. Our Proposed Approach based on Two-level indexing**

This section presents our two-level indexing approach for keyword search in XML database is based on entropy-based similarity. It involves identifying the type of query keywords, namely, tag or data value, using two-level matching and selection of possible \(T\)-typed nodes. Furthermore, it includes the design of mutual information measure, entropy and entropy-based similarity measure so as to find relevant path for a given XML keyword query. The stages of the proposed two-level indexing approach are shown in Figure 1.

1) **Stage 1-Construction of A Two-level indexing:** With a single frequency table [11], ambiguity exists in whether a query keyword is a tag or a data, making the query process complex. This could be overcome by using the proposed two-level indexing. In order to find whether a query keyword is a tag or a data, each tag (structural node) and data node in XML database will be extracted in this proposed approach. After the pre-processing, two-level indexing is created namely tag info table and data info table, for expediting the processing of keyword query.

2) **Stage 2-Identification of query keywords:** Using the two-level matching between the two indices, all the possible \(T\)-typed nodes will be selected for a query. When searching for an input query, each query keyword is initially searched in the \(\text{tag}_\text{info}\) table. If the query keyword matches with tag(s) in the \(\text{tag}_\text{info}\) table, it is identified as a tag keyword. If the query keyword is not a tag, it is searched in the \(\text{data}_\text{info}\) table until it finds match(es). If a match exists, it is identified as a data keyword.

3) **Stage 3-Selectionof T-typed nodes:** In XML database, the keyword matching tag and the keyword matching data values may occur once or many times in different \(T\)-typed nodes and their sub trees.

For example, in the uwm.xml [12] dataset shown in Figure 2(a), the keyword \textit{Wednesday} appears as a data value of comments and restrictions node in root,course\_listing. In Figure 2(b), the keyword \textit{course} appears as a tag name in root,course\_listing and also as a data value of comments node in root,course\_listing. It also occurs as a data value of restrictions in root,course\_listing. Sometimes, a keyword may appear as a tag name many times with different meanings in XML datasets. Since each tag and data value of XML database are stored separately in two indices, keyword search in our approach is confined to these tables enabling to resolve keyword ambiguity problems [11] in XML keyword search.

In case of a query keyword with more number of matches in XML database, query processing will be complex and arithmetic formulae are to be formulated to filter out the optimum \(T\)-typed node. For an input keyword query, we have keyword matching tag(s) and tag(s) containing keyword matching data values retrieved from \(\text{tag}_\text{info}\) table and \(\text{data}_\text{info}\) table respectively. In our approach, retrieved tags containing keyword matching data values are dependent on the retrieved keyword matching tags. In order to measure mutual dependence between retrieved tags, the concept of mutual information [20] is applied in the approach.
Fig. 2 Portions of data tree for uwm XML dataset

In our approach, data_info table is dependent on tag_info table. Intuitively, the amount of information provided by the occurrence of query keyword matching tags about the occurrence of leaf tags containing keyword matching data values is considered as Point wise MI. This reduces uncertainty of leaf tags containing keyword matching data with information obtained from keyword matching tags. By incorporating the concept of mutual information, we define $I(t_d)$, which is the point wise mutual information (PMI) between query keyword matching tags ($t_k$) and leaf tags containing query keyword matching data values ($t_d$) as follows:

$$I(t_k, t_d) = \log_2 \left( \frac{f(t_k, t_d)}{f(t_k)f(t_d)} \right)$$

(1)

where $t$ denotes keyword matching tags, $t_d$ denotes leaf tags containing keyword matching data values, $f(t_k)$ is the sum of frequencies of keyword matching tags, $f(t_d)$ is the sum of frequencies of leaf tags containing keyword matching data values, $f(t_k, t_d)$ is the combined frequency of $t_k$ and $t_d$, $r$ is a reduction factor with range (0,1) and $depth(T)$ is the depth of $T$-typed nodes in XML database. The reduction factor $r^{depth(T)}$ in Equation (1) is used to reduce mutual information of the node types that are deeply nested in the XML database.

If the given query keyword matches with tag(s) in the tag_info table, the approach will retrieve $t_m$, $f_i$ and $path$ from the tag_info table. Frequency of leaf tags of each keyword matching data value will be added together to get $f(t)$. Based on the combination of keywords $t_k$ and $a_i$ in a given query, for each extracted prefix path, combined frequency $f(t_k, t_d)$ will be calculated using execution of designed structured query from the tag_info table and the data_info table. Mutual information between retrieved tags will then be computed for every prefix path extracted. PMI will have negative values if events $t_k$ and $t_d$ co-occur less frequently than by random chance.

As mentioned earlier, for a given query, the query keywords may occur in different node types and the corresponding prefix paths will be extracted. For every extracted prefix path, the proposed approach will find mutual information using Equation (1). Based on the mutual information Equation (1), it seems that query keywords occur in the prefix path with the highest mutual information. If either number of keyword matching tags or number of leaf tags containing keyword matching data values in a path is comparatively more, the corresponding mutual information value will be obviously higher. However, it does not guarantee that query keywords occur certainly in such prefix path with highest mutual information. Because keyword matching data values and related information such as the number of occurrence of data value contained in the leaf tag and the corresponding prefix path are not specifically considered in the mutual information calculation. For this reason, the relevant path for a query may be determined exactly by calculating entropy-based similarity measure in the next stage.

4) Stage 4-Finding $T$-typed node and Relevant Path: The similar matching data values may have one of the following cases: (i) exact keyword matching data values (ii) data values containing keyword matching data value (iii) combination of (i) and (ii). Relevant leaf nodes of these similar matching data values may be the desired search via nodes, i.e., keyword matching tags (nodes) or other undesired leaf tags (nodes). These other undesired leaf tags (nodes) need to be filtered out in order to find the optimum relevant path for query through entropy-based similarity.

Entropy [20], [21], [22] $H(X)$, a measure of uncertainty, of a random variable $X$ with $n$ outcomes $\{x_i; i = 1, ..., n\}$, is defined as

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

(2)

Where $p$ is the probability mass function of $x_i$. In this definition of entropy, the convention $0 \log_0 0 = 0$ is adopted.

While searching for a keyword query in XML database, it is necessary to have co-occurrence of keyword matching data value and desired leaf tag. To measure path-wise, uncertainty of frequency of occurrences of keyword matching data value contained in a leaf tag, based on the definition of entropy, we propose an entropy formula as follows:
\[ H(a_i) = - \sum_{j=1}^{n} f(a_i) \log f(a_i) \]  

(3)

Where \( H(a_i) \) is entropy of keyword matching data value contained in a leaf tag, \( f(a_i) \) is the frequency of occurrences of keyword matching data value contained in leaf node, \( a \) represents a data keyword in query, and \( n \) is the number of data keywords in query.

For each relevant leaf tag, its path will be compared with all paths extracted for the query keywords. When the path through which relevant leaf node occurs is same with the extracted path, the mutual information calculated in Stage 3 for the extracted path is incorporated into the corresponding entropy to get the entropy-based similarity measure. Based on this, we propose an entropy-based similarity (ES) formula to compute similarity between query keyword matching tags and leaf tags containing keyword matching data values and query as follows:

\[ ES(t,a) = \text{abs}(I(t,t_d)) + H(a) \]  

(4)

where \( I(t,t_d) \) is mutual information between \( t \) and \( t_d \) and \( H(a) \) is entropy of keyword matching data value contained in a leaf tag as given in Equation (3). \( ES(t,a) \) is based on path-wise mutual information value between keyword matching tags and leaf tags containing keyword matching data values and entropy of keyword matching data value contained in a leaf tag. The proposed entropy-based similarity measure is used to evaluate the accuracy of occurrence of query keywords path-wise so as to find relevant path for a given query.

### IV. ALGORITHMS

#### A. QUERY PROCESSING

Algorithms for the proposed approach include three stages, i.e., select path, findMutualInformation and find Similarity. Initially, the select Path stores the path of each query keyword. The findMutualInformation process computes the mutual information between keyword matching tags and leaf tags containing keyword matching data value for every prefix path extracted. The findSimilarity process computes the similarity between query keyword matching tags and leaf tags containing keyword matching data values and query for every prefix path extracted, based on the entropy-based similarity formula.

Due to space constraints, we will only be presenting the main algorithm which is findMutualInformation (see Algorithm 1). The procedure findMutualInformation() computes the mutual information between query keyword matching tags \( t_i \) and leaf tags containing query keyword matching data values \( t_d \). First, it computes the sum of frequencies of keyword matching tags \( f(t_i) \) by calling function getTagFrequency() (line 1-2). Second, it computes the sum of frequencies of leaf tag containing keyword matching data values \( f(t_d) \) by calling function getDataFrequency() (line 3-4).

Then, it calls function getCombinedFrequency() to compute the combined frequency of the tags \( t_i \) and \( t_d \) by executing the designed structured query based on tag keyword and data keyword combination in input keyword query (line 5-15). Using the Equation (1), it calculates the mutual information between \( t_i \) and \( t_d \) for every prefix path extracted (line 16-18). Finally it adds path and path-wise mutual information value to mlist() (line 19).

For each data keyword \( a_i \) in a query, the procedure findSimilarity() finds similar matching data values from data info table and stores their information such as keyword matching data value \( (d) \), leaf tags containing keyword matching data values \( (t_d) \), frequency of occurrences of data value contained in the leaf tags \( (f_d) \) and id in dsimilar(). Using Equation (4), it calculates entropy-based similarity \( ES(t,a) \).

**Algorithm 1: findMutualInformation(query)**

//\( t_k \) - keywordMatchingTag
//\( t_d \) - leafTag containing keywordMatchingDataValue
//Find tag frequency
1 for each \( t\exists \in Q \) do
2 \( f(t_d) \) = getTagFrequency\((T)\)
//Find data frequency
3 for each \( D\in Q \) do
4 \( f(t_d) \) = getDataFrequency\((D)\)
//Find combined frequency
5 For each tag and data combination in query
6 Assign a value to cvalue
7 for each result \( \in \text{resultList} \) do
8 Split path from \( \text{resultList} \)
9 tagpath.add(path)
10 Based on cvalue
11 Construct queryString
12 queries.add(queryString)
13 for each queryString \( \in \text{queries} \) do
14 \( d = \text{getCombinedFrequency}(\text{queryString}) \)
15 \( f(t_i,t_d) += d \)
16 \( r = 0.8 \)
17 \( T = \text{path}\.\text{length} \)
18 \( \text{minfo} = (\log_2(f(t_i,t_d)f(t_d)))/\text{depth}T \)
19 Add data path and minfo to mlist()
20 return mlist()

For example, consider a keyword query ‘course, 216-239’ on the uwm.xml dataset. As mentioned in Section III-B3, the keyword course occurs as tag name and data value in the uwm.xml document. While searching for the query using the proposed approach, course and 216-239 are searched separately in the tag info table and data info table. Our algorithm identifies course as a tag and 216-239 as a data value. Hence, the proposed two-level indexing approach confines the searching of keywords separately in the two indices to resolve such keyword ambiguity. Mutual information between keyword matching tags and tag containing keyword matching data value will be computed by efficiently using tag frequencies. For example, consider the query ‘comments, Wednesday’ in the uwm XML dataset. The
keyword Wednesday occurs as data values of restrictions and comments. The approach identifies comments as tag and Wednesday as a data value. However, co-occurrence of keyword Wednesday with desired node type, i.e., comments needs to be considered. In order to achieve this, matching between all the leaf tags of similar matching data values and query tag keywords is carried out.

For data keyword, similar matching data values are found in the data_info table. For these matching data values, relevant information, namely, leaf tags containing keyword matching data values, frequency of occurrences of data value contained in the leaf tags and id of keyword matching data value contained in the leaf tags are obtained. For each data keyword, we have only keyword matching data values contained in leaf nodes from which relevant leaf tag (node) with highest frequency of occurrence of data values in the leaf tags will be selected. Uncertainty of probability of occurrences of keyword matching data values contained in leaf node(s) will be measured using entropy formula. Similarity between retrieved tags (nodes), namely, query keyword matching tags (nodes) and leaf tags (nodes) containing keyword matching data values and query, will be computed to find relevant path for a given input query.

V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a two-level indexing mechanism to identify the relevant information in a XML database, which contains a University Listing Course Dataset. Our approach considers uncertainty of frequencies of tag and data value in the query processing. By incorporating the concept of mutual information and dependence of two indices, we define mutual information which is a measure of statistical dependence between query keyword matching tags \( t_k \) and leaf tags containing keyword matching data values \( t_d \). By adopting the concept of entropy, we design an entropy-based similarity formula to compute similarity measure between \( t_k \) and \( t_d \), and query in order to evaluate the accuracy of occurrence of keyword queries path-wise in XML database. Our future work includes experimentation with algorithms and designing the popularity of query results that have comparable relevance scores and evaluating the results to prove the efficiency of the proposed approach in XML keyword search. Our further research objective is to evaluate the designed approach on various XML databases and evaluate the approach with some existing keyword-based search algorithms.

REFERENCES


