Effectively Return Query Results for Keyword Search on XML Data

Teng He, Guoqing Wu, Qingsong Chen, Junfeng Zhou, and Ziyang Chen

School of Information Science and Engineering, Yanshan University, Qinhuangdao, China
{804743850,842676860,1832414806}@qq.com,
{zhoujf,zychen}@ysu.edu.cn

1 Introduction

Recently, XML keyword search has attracted much attention and has become a popular paradigm for information retrieval over XML data. Because of its convenience, users don't need to know a complex query language or the underlying data schema. However, due to the inherent ambiguity, a keyword query usually corresponds to a large number of results that may be classified into different types, or different search intentions, among which only a few meet users' search intentions.

To address this problem, existing methods [1, 3] try to firstly infer users' search intention, i.e., the targeting result type, then return results of that type as query answers. The formula used to compute the score of a result type T, i.e., $C_{T,Q}$, w.r.t. a given keyword query Q is as Formula 1 which is designed based on three guidelines [1], where $A = \log(1 + \prod_{k \in Q} f_k^T)$, k is a keyword of Q, f_k^T is the number of T-typed nodes that contain k in their subtrees; $B = r^{depth(T)}$, r is the reduction factor with range $(0, 1]$, and $depth(T)$ represents the depth of T-typed nodes. Even though [3] has found that this formula suffers from inconsistency and abnormality problems, and made improvement on this formula, in practice, both [1] and [3] still suffer from some of the following problems:

$$C_{T,Q} = A \cdot B \tag{1}$$

1. A node type that is taken as a result type may have many node instances that do not contain all query keywords.
   Consider keyword query Q_1 = {Tom, XML} issued on the sample data in Fig. 1. Most likely, it is used to find papers about "XML" written by Tom; hence the result type of Q_1 should be "lab/person/paper". However, [1, 3] take "lab/person" as the result type. Note that node 24 doesn't contain all query keywords.
2. The result type recommended by [1, 3] may contain some query keywords that are descendant of its descendant LCA nodes.
   Consider Q_1 again. [1,3] suggest result type as "lab/person". In fact, node 10 should not be a query result of Q_1, because after removing the subtree rooted at node 12, the subtree rooted at node 10 does not contain any query keyword of Q_1, that is, node 10 is not an LCA node.

Besides [1, 3], [2] still cannot work effectively in some cases. Consider query Q_2 = {paper, Mike} issued on the sample data in Fig. 1. Most likely, it is intended to find

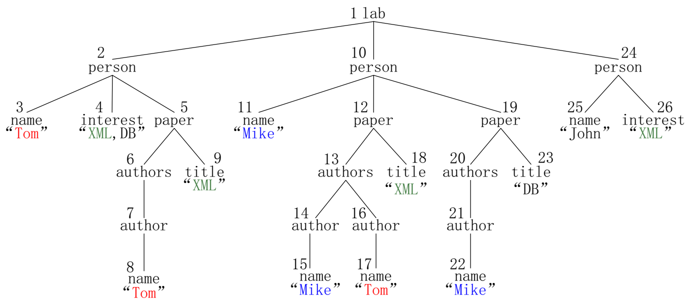

Fig. 1. A sample XML document D

papers written by Mike; hence the result type should be "lab/person/paper". But [2] still suggest result type as "lab/person".

As a comparison, our method avoids all the above problems by using a new ranking function that takes the number of ELCA [4] nodes into account.

2 Overview

2.1 Ranking Function for Result Types

Intuitively, our method takes all ELCA nodes as input to compute the score of each result type, the more the number of ELCA nodes of type T, the more possibility type T be the promising result type of Q. As shown by Formula 2, where S^T_{ELCA} is the set of ELCA nodes of type T.

$$C_{T,Q} = \log |S^T_{ELCA}| \tag{2}$$

2.2 Ranking Function for Results

The main idea of TF*IDF (Term Frequency * Inverse Document Frequency) similarity is: a keyword appearing in many documents should be regarded as being less important than a keyword appearing in a few, while at the same time, a document with more occurrences of a query keyword should be regarded as being more important for that keyword than a document that has less. The formula of TF*IDF similarity is as follows, where Q is the keyword query, d is a document, N is the number of documents, f_k is the number of documents which contain query keyword k, and $f_{d,k}$ is the number of occurrences of keyword k in d.

$$\rho_{Q,d} = \log(\sum_{k \in Q} (\frac{N}{f_k} \cdot f_{d,k})) \tag{3}$$

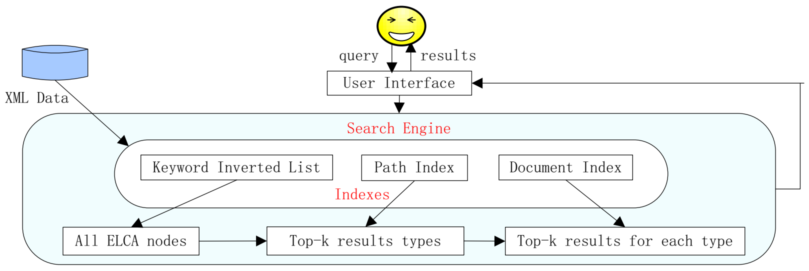

Fig. 2. System architecture

Even though TF*IDF similarity is suitable for flat documents, it only considers the content, regardless of the structure of XML documents. Hence, it cannot be directly applied to XML documents with hierarchical structure. We consider the following rules to compute score of query results.

Rule 1: The higher the relevance of a result subtree and a query, the higher the score of the result is.
Rule 2: The smaller the distance of query keywords and ELCA node, the higher the score of the result is.
Rule 3: The more compact the result subtree, the higher the score of the result is.

The formula of ranking results is shown by Formula 4, where d is a result subtree rooted at an ELCA node, $\rho_{Q,d}$ is the relevance of query Q and d, and $|E_d|$ is the sum of the lengths of the shortest path from each query keyword to the ELCA node.The edges that are repeated on the path are counted only once, which addresses Rule 3.

$$S_{Q,d} = \frac{\rho_{Q,d}}{|E_d| + 1} \tag{4}$$

2.3 Demonstration

Fig. 2 shows the system architecture. The Indexes in Search Engine are created in advance. After user issues a query, we firstly find all ELCA nodes by using Keyword Inverted List, then classify these results into different types according to their node type from Path Index. After that, we compute scores for all these node types by Formula 2 and get top-k node types with the highest score as search intentions to the given keyword query. Finally, for ELCA nodes of each type, we compute scores by Formula 4 and return the first result(the subtree rooted at ELCA node) of each result type to users. The Document Index records the informations contained in each node.

Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (No. 61073060, 61272124), the Research Funds from Education Department of Hebei Province (No. Y2012014), and the Science and Technology research and development program of Hebei Province (No. 11213578).

References

1. Bao, Z., Ling, T.W., Chen, B., Lu, J.: Effective XML keyword search with relevance oriented ranking. In: ICDE, pp. 517–528 (2009)
2. Li, J., Liu, C., Zhou, R., Wang, W.: Suggestion of promising result types for XML keyword search. In: EDBT, pp. 561–572 (2010)
3. Li, J., Wang, J.: Effectively inferring the search-for node type in XML keyword search. In: Kitagawa, H., Ishikawa, Y., Li, Q., Watanabe, C. (eds.) DASFAA 2010, Part I. LNCS, vol. 5981, pp. 110–124. Springer, Heidelberg (2010)
4. Xu, Y., Papakonstantinou, Y.: Efficient LCA based keyword search in XML data. In: EDBT, pp. 535–546 (2008)