An Extended Vector Space Model for XML Information Retrieval

Guo Yongming
College of information science and technology
Donghua University
Shanghai, China
guoymsh@gmail.com

Chen Dehua, and Le JIajin
College of computer science and technology
Donghua University
Shanghai, China
csdehua@gmail.com, lejiajin@dhu.edu.cn

***Abstract*—With the emergence of more and more XML documents, effectively and efficiently retrieving information from XML documents has become an active research area. Since XML documents lie between structured data and unstructured data which describe both content and structure, it is a huge challenge for effectively and efficiently retrieving information from XML documents. This paper develops a novel retrieval model named as Extend Vector Space Model which effectively combines XPath and Vector Space Model for XML information retrieval. A prototype system for XML information retrieval based on this retrieval model has been implemented, and several corresponding algorithms have been introduced. The experiments show that this model has effectively improved recall and precision.**

Keywords-XML ; informatiion retrieval; XPath; Extended Vectoe Space Model

I. INTRODUCTION

During the last few years XML has become a de factor standard for information representation and data exchange over the Internet [1]. With the emergence of more and more XML documents, how to effectively retrieve information from XML documents has become an active research area. Since XML documents are a kind of semi-structured data lied between structured data and unstructured data, how to efficiently retrieve XML documents has attracted more attentions from both database (DB) and information retrieval (IR) community.

The DB community regards XML as a standard format for structure data exchange, and XML documents are regarded as highly structured data. Various XML query languages derived from structured data have been proposed, such as Xquery [2], XPath [3], and many query evaluation and optimization algorithms have correspondingly been investigated [5].

However, the IR community views XML as a format for representing the logical structure of a document. Thus, XML documents are regarded as a collection of text documents with additional tags and relations between these tags. Various traditional IR techniques and model have been extended to retrieve XML documents [4].

As discussed hereinbefore, the DB community has focused on developing query languages and efficient evaluation algorithms for highly structured content, while the IR community traditionally focused on searching text content and ranking query results and evaluating their effectiveness. Thus, how to integrate IR and XML query techniques for more sophisticated search on the structure as well as the content of XML documents is not widely researched.

In this paper, we develop a novel model named as Extended Vector Space Model(EVSM) through integrating XPath and the traditional Vector Space Model, and try to overcome the deficiency of both and make use of both strength so that effectively take into consideration both content and structure.

The rest of this paper is organized as follows. The next Section presents the preliminary technology. A description of Extended Vector Space Model for XML information retrieval is provided in Section 3. The implementation of the prototype system is presented in Section 4. Then, the result of experiment is discussed in Section 5. In Section 6, we review related work for XML information retrieval. Finally, we present our conclusion and further work in Section 7.

II. PRELIMINARIES

A. XPath language

An XML document is composed of nested elements, where each element consists of a pair of tags and sub-element or data between tags, and Fig. 1 is an example of XML document. Furthermore, an XML document is represented as a tree in which each element corresponds to a node, and Fig. 2 is a tree representation of the document in Fig. 1.

XPath is an XML query language used for describing term's path (absolute path or relative path), by which users are able to query document with structure and locate node. This paper specifies term's path using a simple form of XPath, and only retain four operations from XPath:"/","//","*","|", to describe path of a term. A path is a series of node which is traversed from a certain node to the leaf node. For example, '/book/title/XML', denotes that traverses 'title' node from 'book' node to 'XML' leaf node.

B. Vector space model

The vector space model is one of the most popular models in IR. It is based on the comparison of the query term vector with the document term vectors. Each term has a certain weight which reflects its descriptiveness with respect to the query or document. The relevance of the document D to the query Q, denoted below sim(Q,D) , is then usually evaluated by using a measure of similarity between vectors such as the cosine measure,

978-0-7695-3543-2/09 $25.00 © 2009 IEEE
DOI 10.1109/WKDD.2009.218

$$sim(Q,D) = \sum_{ti \in Q \cap D} \frac{wQ(ti)wQ(ti)}{\|Q\|\|D\|} \qquad (1)$$

where, $w_Q(t_i)$ stands for the "weight" of term t_i in query Q; $w_D(t_i)$ stands for the "weight" of term t_i in document D; $\|Q\|$ denotes the number of terms in Q; $\|D\|$ denotes the number of terms in D.

```
<?xml version="1.0" encoding="UTF-8"?>
<articles>
   <article>
      <year>2000</year>
      <author>
         <name>John</name>
      </author>
      <title>xml retrieval</title>
      <section>
         <paragraph>
            XML information retrieval …
         </paragraph>
      </section>
      <section>
         …
      </section>
      …
   </article>
  <article>
  …
  </article>
  …
</articles>
```

Figure 1. An example of XML document

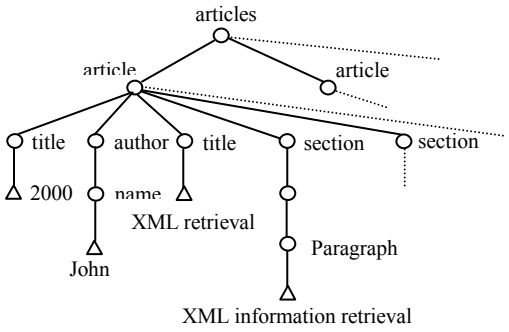

Figure 2. XML document tree

III. EXTENDED VECTOR SPACE MODEL

An XML document is essentially a text document with a tree-like structure, composed of structure and content, and structure is in fact composed of tags which are used for restricting contents. Thus, an XML document can be regarded as a set of terms with structural constraints. On the basis of this case, in this paper we proposed an Extended Vector Space Model, where the general process is described as follows.

First, each XML document is mapped to a labeled tree, where each element corresponds to a node. In order to simple the model, we treated attribute node same as element node. Furthermore, a series of processing is done on contents of elements (text data), such as separating words, removing stop words and stemming. After this processing, contents of elements are represented as a set of terms, where each term corresponds to a leaf node of the labeled tree, and the frequency of each term occurrence in the element is computed. Additionally, attribute value is processed just as element contents. Second, each term corresponding to a leaf node of the labeled tree has a path from the root node or ancestor node to the element node comprising itself. In fact, we take comprising relation between nodes into more consideration, rather than sibling relation between nodes. Thus, we can simply regard an XML document as a set of terms with path. Through extending the concept of term to term with path, which is also called structural term, we translate the similarity of the query to the XML document into the similarity between query vector and document vector.

In this paper, we adaptively determined a part of the whole XML document as retrieval unit by introducing logical document [6]. And the size of the logical document is determined by user at query time or is determined by the system in run. The logical document is a common ancestor comprising all terms with path, and the label of the root note of the logical document is called the type of the logical document. In order to explicitly describe the model, we introduce the following concept.

Through extending the term concept to the concept of the term with path, we introduce the concept of the structural term.

Definition 1 (Structural term) A structural term t is a term with a path from root node or certain node to term itself.

In figure 2, the term 'john' and its path 'article\author\name' constitute a structural term (john, article\author\name\john).If there exits a path from the root of the structural term t to the term itself in the logical document d, we say that t occurs in d.

In this paper, we represent the logical document as a bag of structural terms. The importance of the term to the document is measured by the classic tf/idf weight schema.

Definition 2 (term frequency) Let t be a structural term, and d be a logical document of type p. Let $freq_t(d)$ be the number of occurrences of t in d, and maxfreq(d) be the maximal occurrences of any term in d. The term frequency $tf_t(d)$ of t in d is defined by

$$tf(t) = \frac{freq_d(t)}{\max freq(d)} \qquad (2)$$

Term frequency represents the number of occurrence of the structural term in the logical document.

Definition 3 (inverse document frequency) let t be a structural term, and p be a type. Let $|C^p|$ be the number of logical documents of type p in C, and n_t be the number of documents contains t. The inverse document frequency idf_t^p of t is defined by

$$idf_t^p = \log \frac{|C^p|}{n_t} + 1 \quad (3)$$

We combine the definitions of term frequency and inverse document frequency, to produce a composite weight for each term in each logical document.

Definition 4 (Term weight) Let d be a logical document of which type is p, the weights of the structural term is defined by

$w_t = tf_t(d) \cdot idf_t^p$

The user query is also composed of several structural terms and the weight of each structural term is given by the user. For example, the query q is denoted by ((john, article\author\name),0.4), (XML retrieval，article\title,),0.6).

The similarity sim(q,d) between the query q and the logical document d is computed by (4)

$$sim(q,d) = \sum_{ti \in q \cap d} \frac{wQ(ti)wQ(ti)}{|q| \cdot |d|} \quad (4)$$

where |q| denotes the number of structural terms in q, and |d| denotes the number of structural terms in d.

IV. IMPLEMENTATION OF THE SYSTEM

A. Encoding and indexing document

All element nodes of the document tree are encoded using Dewey codes [11]. Every node in the document is represented by a combination of its parent label and an integer number. If u is the x^{th} child of s in XML tree then label of u, label(u), is concatenation of label of s and x which is presented as label(s).x. For example if element label for u is 2.5.3 then its 5th child label will be 2.5.3.5.

In this paper, we adopt inverted index and hash index to index documents. The index of the system is composed of four parts: term(text or value) index, path index, element index and document index.

Term index is an inverted index that maps each term to a posting that stores references to all paths containing the term. The term index is composed of two parts: vocabulary and posting. The Vocabulary is a list contains all terms of text, whereas the posting is represented by a triple of integers: the id of the path; the id of the document; the number of the occurrence of the term in the path of the document. Figure 3 exemplifies this index in details.

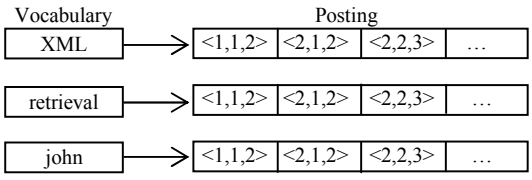

Figure 3. term inverted index

Path index consists of an inverse path and the id of the path which is determined by a hash function.

Element index comprises element name, element id, level, did and other statistic information (such as the number of structural term in the element and its size).

Document index is made up of did (document id), document name, document location, document DTD, document description.

B. Query processing

In this section, we present the algorithm to the query processing. For a query composed of several structural terms, we compute respectively the weight of each structural term in the logical document and then composite an overall score using (4). This query processing algorithm is also called EVSM Algorithm. Figure 4 describes this algorithm, and depicts how to compute the similarity between the query and the logical document in details.

Algorithm EVSM(query)
For each structural term in the query, do 1-4.
1. Compute the hash value of each query path.
2. Compute $freq_t(d)$ and n_t according the term index and path index.
3. Compute maxfreq(d) and $|C^p|$ accessing the element index.
4. Compute $w_t = tf_t(d) \cdot idf_t^p$.
Last, compute the overall sim(q,d) according to Equation(4).

Figure 4. Algorithm EVSM

C. Relaxing the matching of path

In the basic extended vector model, a path matching is full matching from the root node to the leaf node. In this variant we relax the path matching by computing the path similarity between the query and the logical document. In the basic extended vector space model, when there exits a path from the root node of the structural term t to the term itself in the logical document d, we say that t occurs once in d. In the variant (REVSM), the occurrence of the structural term in the logical document is a numeric value from α to 1. When the path of the structural term in the query completely matches the path of the structural term in the logical document, the numeric value is 1. While the path completely no matches, the numeric value is α. This numeric value of the occurrence is computed by (5).

$$occur(t) = \alpha + sim_p \times (1 - \alpha) \quad (5)$$

where sim_p denotes the similarity of the path of the structural term in the logical document to the path of the structural term in the query. The path similarity between p1 and p2 denoted by $sim_p(p_1, p_2)$ is computed by (6)

$$sim_p(p_1, p_2) = \frac{eql(p_1, p_2)}{\max(|p_1|, |p_2|)} \quad (6)$$

where eql(p_1,p_2) denotes the number of identical nodes between p_1 and p_2, , and $|p_1|$ (or $|p_2|$) denotes the number of nodes in the path p_1 (or p_2).

V. EXPERIMENT EVALUATION

In this section, we present an experiment evaluation of the basic Extended Vector Space Model (EVSM) and its variant (REVSM) on the INEX IEEE Collections [7]. We also compared EVSM and its variant with other model, such as FXML [5] and VSM [4]. We adopt Precision-recall curves to evaluate the retrieval model. From fig. 5, we can observe our approach has significantly high precision at low recall regions compared with other two approaches. Maybe the reason is because our model takes into consideration the containing relation between structure and content. We also find that REVSM have some improvement than EVSM. The reason behind this may be because REVSM contains more terms than EVSM and accordingly the weight of the structural term is much higher than EVSM.

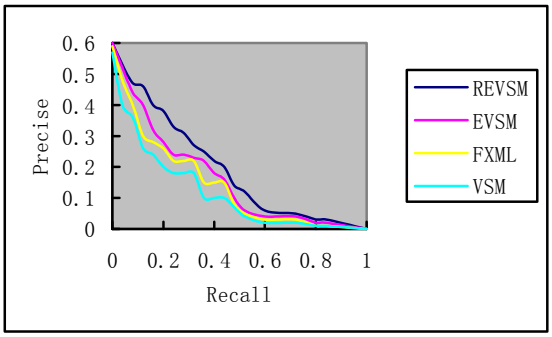

Figure 5. Recall/Precise Curve(α=0.2)

VI. RELATED WORK

Due to its simple theory and better retrieval effect, vector space model has gained popular in traditional information retrieval. Thus, it is very natural idea to extend the vector space model for XML retrieval.

Schlider et al. [3] regard an XML document (or query) as a vector made of some sub-trees of the document (or query) tree, and compute the similarity of the query vector of the document vector by adopting standard vector space model. However, regarding every sub-tree of the document tree corresponding to a component of a vector is very complex for computation of the similarity. Maria et al. [5] introduce an fxml factor to improve the vector space model for XML retrieval, but complete path matching is not considered. In [9], the author incorporates vector space model in their user-driven XML retrieval system. The query is segregated into structural constraints and content part, and the similarities between the query and the document are measured separately for meta-data and content considering different term-space depending on structure/level. Final RSV is calculated as a linear combination of these two similarity-values and this RSV is used for ranking. However, they also not consider effectively combining content and structure. In [10], the author makes a naive effort to apply VSM in their EXTIRP system, where information coded in tags is outright ignored while considering content-oriented queries. Two indices are built, one for words and the other for phrase. The entire document collection is divided into disjoint fragments which are independently treated as full-documents under traditional tf-idf based VSM. The schema suffers from the problem of fragment combination as indexed fragments are static. However, this method completely ignores the structure of XML documents.

This paper extends the concept of terms to structural terms by regarding terms with its path as terms in vector space model. It has acquired better retrieval effect by naturally combing contents and structures of XML documents.

VII. CONCLUSION AND FURTHER WORK

This paper presented an Extended Vector Space Model for XML information retrieval by combining Vector Space Model and XPath language. The Experiments show this model acquire better retrieval effect. However, in order to simple the model, we do not consider the sibling relation between nodes, this affects retrieval precise in some certain degree. In the future, we will take sibling relation into consideration and introduce related factor to improve retrieval precise.

ACKNOWLEDGMENT

This work is supported by the Scientific Research Common Program of Beijing Municipal Commission of Education, No. KM200810011008.

REFERENCES

[1] Extensible Markup Language (XML) http://www.w3.org/XML/.

[2] W3C (1999) "XML Path Language". http://www.w3.org/TR/xpath.

[3] W3C (2001) "XML query Language". http://www.w3.org/TR/xquery.

[4] W. Robert, H. leong, S. Tharam, T. Alvin. A survey in indexing and searching XML documents. JASIS 53(6), p 415-437, 2002.

[5] I. Maria, P. Lucas, Z. Nivio. A Universal Model for XML information retrieval. INEX2004, LNCS 3493, p311-321, 2005.

[6] B. Ribeiro-Neto, R. Baeza-Yates. Modern Information Retrieval. 1999.

[7] N. fuhr, M. Lalmas. INEX document Collecion.

[8] T. Schlider & H. Meuss. Querying and ranking XML documents. JASIS 53(6), p 489-503, 2002.

[9] M. Hassler and A. Bouchachia. Searching XML document – preliminary work. Inex 2005, pages 95-109, 2005.

[10] M. Lehtonen. When a few highly relevant answers are enough. INEX2005, p215-216,2005.

[11] J. Lu, T. Wang Ling, C. Chan, T. Chen : From Region Encoding To Extended Dewey: On Efficient Processing of XML Twig Pattern Matching. VLDB (2005) 193-204.