Research on Similarity for XML Information Retrieval

Ren Xueli
School of Information Engineer
Qujing Normal University
Qujing,China
oliveleave@126.com

Dai Yubiao
School of Information Engineer
Qujing Normal University
Qujing,China
abiaodai@126.com

Abstract—With the continuous development of Internet and rich resources emerging on the Web, information retrieval based on XML has emerged; the similarity of documents is the basis of information retrieval. A new method SC-Similarity is proposed to compute similarity of XML documents from structure and content in the paper. XML document is expressed as a collection of tuple, the paths are extracted and delete the recurring in order to improve efficiency, and matching fuzzy path using dynamic programming and WordNet; and then the structure similarity between documents are calculated using Hungarian algorithm; the content similarity are estimated by set matching. Finally, the similarity of XML documents is estimated. Two experiments are done to show that the method is effective, the experiment 1 test structural similarity; the information retrieval is test using automatically generated documentation sets and real data sets in the experiment 2, and results show the accuracy may arrive at 95%.

Keywords—XML; Information Retrieval; Structure Similarity; Sematic; Content

I. INTRODUCTION

XML has become extremely popular in management and mining of semistructured/hierarchical text data due to its abilities in representing information in a well-defined, extensible and machine readable format [1-2]. The use of XML covers data representation and storage, database information interchange, data filtering, as well as web services interaction. With the development of web and the wide usage of XML, XML-based similarity/comparison becomes a central issue in the database and information retrieval communities. It has become a hot research topic currently how to realize the automatic retrieval, classification and clustering of XML documents effectively. The similarity of XML is an important basis to retrieve information, so it is studied in the paper.

II. SIMILARITY MEASUREMENT OF XML DOCUMENTS

A wide range of algorithms for computing similarity in XML documents have been proposed in the literature. They are Tag matching, Edge matching, Path matching, Edit Distance and so on[2-9]. Tag matching is known as the simplest measure for XML similarity, as it only considers the intersection of the sets of tags over the union between the documents being compared. Nonetheless, using tag matching, the structure of the documents is completely ignored, thus attaining low clustering quality. Edge matching considers the edges connecting XML nodes, the father-son relations in the comparison process; so it's more accurate than tag similarity. But the comparison uses match perfectly, and not consider the nodes semantic. The authors in [3] describe the structure of an XML document as a set of paths, compute similarity by taking into account all the paths in the path set of the second XML tree. If the more paths two XML documents share in common, then the more similar they are. It's more accurate than edge similarity, as it considers not only father-son relation but also grandchild relationship. But the result of similarity uses exact match, and not consider the semantic information of nodes. Viewing XML documents as trees, Nierman and Jagadish [6] use the graph edit distance measure to compute the structural similarity between two XML documents. The algorithm for this distance measure was derived from one for the edit distance between strings. Given a set of graph edit operations, such as deletion, insertion, and substitution, the edit distance is defined as the shortest sequence of edit operations that transform one tree into the other. In practice, a cost may be assigned to each individual operation to reflect its importance. Typical tree distance algorithms include [7] and [8]. Flesca et al. [9] represent XMl documents as time series and compute the structural similarity between two documents by exploiting Discrete Fourier Transform of the corresponding signals.

III. OUR APPROACH

As stated in the above, a new method called SC-Similarity is proposed which determines similarity considering both the structure and the content of XML documents, and it is effective on classifying XML documents which are created from the same DTD. The structure similarity between the two XML documents is computed by path matching, and the content similarity is computed by optimal matching. The similarity between query Q and document D is defined as

$$sim(Q, D) = CSim(Q, D) \times \alpha + SSim(Q, D) \times \beta \quad (1)$$

In that: $\alpha + \beta = 1$, and they depend on the importance of the content similarity and structure similarity. If α =0, then the similarity is only considered structure similarity and no content similarity;

If α =0, then the similarity is only considered content similarity and no structure similarity. It has the following characteristics:

$Sim(Q, D) \in [0,1]$

$Sim(Q, D) = Sim(D, Q)$

$Sim(Q, D) = 1 \quad A = B$

The process to compute similarity is listed in Fig .1 .

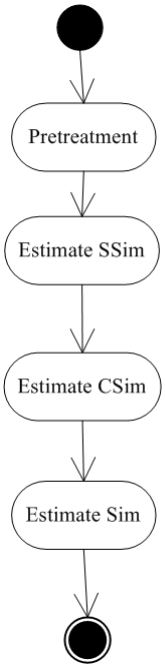

Figure 1. the process to compute similarity

The following definitions are given firstly for the convenience of description.

Definition 1 XML document tree: an XML document tree is an ordered labeled tree parsed from an XML document where content isn't included in the tree. The top node is root and the lowest node is leave, every level in XML document tree from root to leave is numbered using integer from 1.For example, Fig.3 is a tree, note is the root, George, John, Reminder and Don't forget the meeting are leave.

Definition 2 tuple: the document tree is traversed by depth-first and describe as tuples with the form <path,content>. In that a path is a element sequence from the root element; content is a set with leave node of the path. As an example,< note/to,{ George}> is a tuple in Fig.3.

Definition 3 tuple set: It is a set including the whole tuples of XML document. In the set if paths are same, then the contents are merged into the content set of a tuple. The tuple set of Fig.3 is {<note/to,{ George }>,<note/from,{John}>,< note/heading,{Reminder}>,< note/body,{ Don't forget the meeting!}>}

Definition 4 Path Set(PS) : PS is the whole path set from tuples. As an example, a set {note/to, note/from,note/heading,note/body } is the path set of Fig.3.

Definition 5 The longest common subsequence (LCS): it is to find the longest subsequence common to all sequences in a set of sequences. Suppose that S is a common subsequence of two sequences P1 and P2,then s has 2 condition: 1)S $\subset$ P1 and S $\subset$ P2 ; 2) S is the longest sequence which Satisfies the condition 1.For example :the subsequence bookstore/book is the longest common subsequence of two path bookstore/book/author and bookstore/book/price.

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>George</to>
  <from>John</from>
  <heading>Reminder</heading>
  <body>Don't forget the meeting!</body>
</note>
```

Figure 2. XML Document

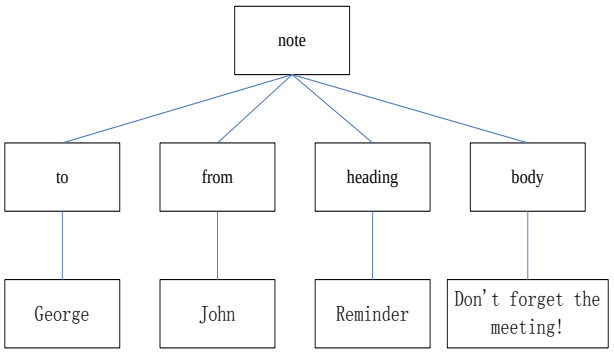

Figure 3. XML document tree

A. Pretreatment

XML document is composed of elements, attributes, XLink, comments and etc., Xlink is very important in data usage, but it is not the case in the impact of the document structure; in addition, comments are added in order to facilitate understanding of the document and information, they don't impact on document structure; therefore they are not considered in the calculation process of structural similarity[10].

B. Estimate structure similarity

In general, XML document should follow a given syntax, DTD or XML Schema, which define the corresponding XML elements appearing in this document, related properties, and rules of each element or attribute should be followed. However, XML documents are often found on the web no corresponding syntax documents, especially XML documents created based on valid HTML, so it is necessary to analyze and determine whether the XML document has the same structure.

The same nodes in the XML document have effect on retrieval efficiency, so the problem of duplicate nodes is solved based on define 1 firstly; in addition, As the XML document is compiled by different mechanisms. The same content is described by different elements; the fuzzy match is used by WordNet to solve the problem. In order to improve the efficiency and accuracy of the calculation, firstly, paths are extracted from tuple set; then the path similarity is computed; finally, structure similarity is computed by optimal matching.

1) Extract path

The xml document is modeled as a tuple set which may include the same path and different content. To decrease complexity, the two steps are taken for the tuple set of XML document. Firstly, the attribute node becomes the sub node of the element node, as some information of xml

document may be denoted using attribute or sub element, so all of the attribute information transform to sub element to compute simple. Secondly, the paths are extracted from the tuple set, then the tuple are deleted that have the same paths and the content are merge into the tuple with the same path; finally all of the paths are extract from the tuple set to compute structure similarity. Simplify tuple set may decrease complexity that is constructed using the following pseudo code.

Input : Tuple set TS

$TS = \{\langle p_1, c_1\rangle, \langle p_2, c_2\rangle \cdots \langle p_n, c_n\rangle\}$

Output：Simply tuple set

Extract path from every tuple called

$P = \{p_1, p_2 \cdots p_n\}$

For i=2 to P.length

For j=1 to i-1

If $p_i = p_j$ then

For k=j+1 to P.length

deletet tuple i

add the content in tuple i to the content set of tuple j

Next

P.length=P.length-1

end if

Next j

Next i

2) Path similarity

The path similarity is relate to not only these nodes in the path and levels of nodes in XML documents, and the root node is the most influential to the whole similarity, and then decreased, so the path similarity is defined as the sum of product between the matching degree of all the nodes in the path and the weights of the hierarchy. In that the matching degree of the nodes is computed by WordNet, and the weights of the hierarchy i is defined as $\frac{1}{2^i}$. The degree of node matching divide into two kinds of cases:

(1) If the similarity of semantic is larger than 0.7, then the degree of matching equals to the similarity of semantic;

(2) If the similarity of semantic is smaller than 0.7, then the degree of matching equals to 0.

The similarity of semantic is computed by the method in [].

The longest fuzzy common subsequence of two paths is determined based on the degree of node matching; the path similarity is computed by the longest fuzzy common subsequence. There are the common features between LFCS and LCS, and dynamic programming is an effective method for solving LCS, so LFCS may be solved by dynamic programming and compute similarity. The process is realized by the following pseudo code.

Input two paths $p_i = \{m_1, m_2 \cdots m_k \cdots\}$ and $p_j = \{n_1, n_2 \cdots n_l \cdots\}$

Output path similarity s_{ij}

$s_{ij} = 0$

For k=1 to $|p_i|$

C[r,0]=0

Next k

For l=1 to $|p_j|$

C[0,t]=0

Next l

For k=$|p_i|$ to 1

For l=$|p_j|$ to 1

Compute Similarity s of m_k and n_l

If $s \geq 0.7$ then //fuzzy match

If $k <= l$ then

$s_{ij} = s_{ij} + s \times \frac{1}{2^l}$

Else

$s_{ij} = s_{ij} + s \times \frac{1}{2^k}$

End if

Elseif $n_k > n_l$ then

k=k-1

Else

l=l-1

End if

Next l

Next k

Return s_{ij}

3) Structure similarity

Structure similarity of documents is the Optimal matching problem between paths, so it is similarity with the task allocation, Hungarian algorithm is one of the most effective algorithms to solve the linear assignment problem that can solve the problem in polynomial time [11-14]. The Hungarian algorithm is used to calculate the optimal path matching of two documents in this paper, and then the average value is calculated as the structural similarity. For the purposes of our similarity measure, it is desired that any such unmatched paths contribute also to the overall similarity between the two path sets. To this end, we add some virtual paths to the smaller set, so that both sets have the same size, and for each such virtual element, its similarity to each path is defined following 2 rules to satisfy the definition of similarity. The original m × n matrix is M where the number of paths in query document is m, and the number of paths in target document is n.

If m > n, the resulting matrix M′ will be m × m and have m − n additional rows filled with0.5.

If m < n, the resulting matrix M′ will be m × m and have m − n additional columns filled with 0.5.

The structural similarity is defined as Hungarian(M′) / m.

C. Estimate Content similarity

Extract the content from the tuple, 3 steps are used to compute content similarity, that are pretreatment, set similarity and content similarity.

Pretreatment: a set of distinct terms{t1,t2…tm},denoted by T, is extracted from the

content in tuple. A term,ti,is a key word that appears in textual content after punctuation mark and stop-word removal and stemming. The query document and target documents are described by the same method.

Set similarity: A matrix is constructed to compute set similarity where a term q of content in query tuple is as a row of matrix, a term t of content in target tuple is as a column of matrix, and the element in the matrix is semantic similarity.

$$SSim(Q, T_i) = \begin{bmatrix} sim(q_1, t_{i1}) & \cdots & sim(q_1, t_{in}) \\ \vdots & \ddots & \vdots \\ sim(q_m, t_{i1}) & \cdots & sim(q_m, t_{in}) \end{bmatrix} \quad (1)$$

$$SSim(Q, T_i) = \frac{\sum_{j=1}^{m} \max_{k=1}^{n}(sim(q_j, t_{ik}))}{m} \quad (2)$$

Content similarity: the content similarity is similar to the set similarity, so the method in set similarity is used to compute the content similarity.

D. Estimate Document similarity

After the structure and content similarity are computed, the document similarity is computed using formula (1). Where $DSim(Q, D)$ is the document similarity of query Q and Document D, $CSim(Q, D)$ is the content similarity of query Q and Document D and $SSim(Q, D)$ is the structure similarity of query Q and Document D, α and β is weight, and their value depends on the importance of the content and structure in information retrieve. If α =0, then the similarity is only considered structure similarity and no content similarity; If α =1, then the similarity is only considered content similarity and no structure similarity. Otherwise, it considers not only structure and also content.

IV. EXPERIMENT

A. Two experiments are done to show the accuracy of SC-Simialrity.

1) Experiment 1 structure similarity

Given the XML document trees from [15]in that Q is query tree, and the others are XML document trees. The similarity are computed only considering structure in the paper to compare the method in the paper to the method in [17]. The result is displayed in in table 1. As their structures are same between Q and R5, the similarity of Q and R5 should be 1,but the method in [15] only is 0.75.,and so on. Our approach is more accuracy.

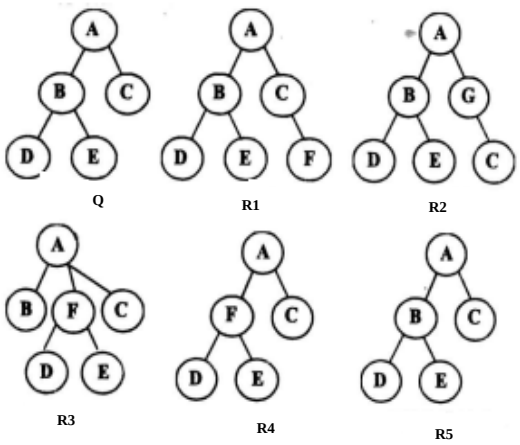

Figure 4. XML document trees

TABLE I. THE SIMILARITY RESULTS BETWEEN Q AND THE OTHER DOCUMENTS

sim	our approach	the method in [15]
sim(Q,R1)	0.916666667	0.75
sim(Q,R2)	0.875	0.6875
sim(Q,R3)	0.71875	0.5
sim(Q,R4)	0.75	0.3
sim(Q,R5)	1	0.75

2) Experiment 2 information Retrieval

The method to compute similarity is used in information retrieval, and precision is look as measure factor. The real data sets and synthetic data sets are used in the experiments , the real data set are quoted from ACM SIGMOD [16], the synthetic data sets are from 3 DTDs[17], 50 XML documents are generated for every DTD by XML Generator automatically , the query contents are constructed using the document in ACM SIGMOD and artificial generation, and α is set to 0.8. The accuracy of information retrieve is shown in Fig .5.

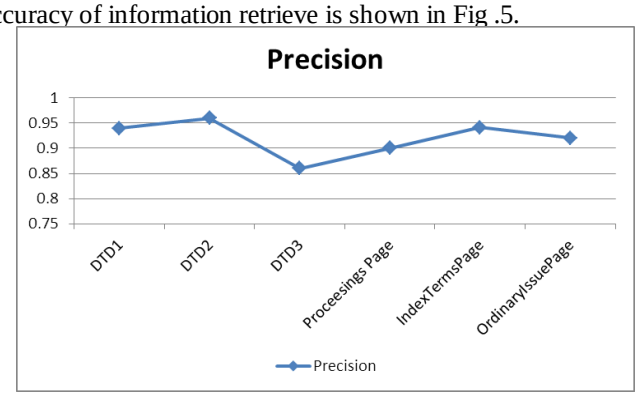

Figure 5. the accuracy of information retrieve

V. SUMMARY

The similarity of XML documents is the key to information retrieval. A similarity method is proposed which compute similarity considering structure and content of XML document. The XML document is represented as tuple, the structural similarity is computed by the longest fuzzy common subsequence, and the content

similarity is computed by set similarity. Two experiments are done by the method in the paper and results show that it is effective.

REFERENCES

[1] Yao, J.T., Varde, A., Rundensteiner, E., Fahrenholz, S.: XML Based Markup Languages for Specific Domains. In: Web-based Support Systems. Advanced Information and Knowledge Processing, Springer, 2010：215－238.

[2] Alsayed Algergawy,Marco Mesiti.XML data clustering:An overview[J].ACM,2011:14

[3] Ren xueli,Dai Yubiao.Research on XML Classification Based on Fuzzy Path Matching[J].computer Application and Software,2015.10:113-116

[4] Andrew Nierman,H.V. Jagadish.Evaluating Structual Similarity in XML Document[EB/OL]. http://db.ucsd.edu/webdb2002/papers/44.pdf.2013.12

[5] K.C.Tai.Tree to tree editing problem[J].ACM,1979:422-423

[6] Nierman, A., Jagadish, H.V.: Evaluating Structural Similarity in XML Documents. In: Mary, F., Fernandez, Y.P. (eds.) WebDB 2002, Madison, Wisconsin, USA, pp. 61–66 (2002)

[7] Ghosh S, Mitra P. Combining Content and Structure Similarity for XML Document Classification using Composite SVM Kernels[C]. The 19th International Conference on Pattern Recognition, Tampa, FL, 2008: 1-4.

[8] Wu J, Tang J. A bottom-up approach for XML documents classification[C]. The 2008 International Symposium on Database engineering and applications, ACM, 2008: 131-137.

[9] S. Flesca,F. Furfaro,S. Greco. Querying and Repairing Inconsistent XML Data[C]. Web Information Systems Engineering – WISE 2005, 175-188

[10] Zhang Na,Zhang Dongzhan.An improved method for classifying XML documents based on structure and content[C].ISCSC10,2010.8:427-429

[11] WordNet[EB/OL].http://wordnet.princeton.edu/wordnet/download/current-version/#win,2014.1

[12] Integer programming. http://www.baike.com/wiki/%E6%95%B4%E6%95%B0%E8%A7%84%E5%88%92.2014.10

[13] Hungarian Algorithm[EB/OL]. http://blog.csdn.net/pi9nc/article/details/11848327.2015.

[14] Hungarian Algorithm[EB/OL]. http://baike.baidu.com/link?url=A5lAHhEx2K6K_2UpVrE5qWOXyzpmUeuziVyTuZfT8qI1zAihQG9vWOC4BCN-an_gQHXlaIk9zqsdw9F3fFbrkK.2015.5

[15] Lu Yuan，Wan Changxuan. The Structural Similarity Computation for XML Information Retrieval [J]. Journal of Information，2007：110-111.

[16] Sigmod XML data sets[EB/OL]. http://www.sigmod.org/publications/sigmod-record/1312,2014.2

[17] Joe Tekli,Richard Chbeir,Kokou Yetongnon. A Hybrid Approach for XML Similarity[EB/OL]. http://www.researchgate.net,2014.1