A Prufer Sequence Based Approach to Measure Structural Similarity of XML Documents

Ramanathan Periakaruppan and Rethinaswamy Nadarajan

PSG College of Technology, Coimbatore, India
rm_pk@yahoo.com

Abstract. XML is a W3C standard for exchange of semi-structured data. For many applications it is necessary to extract information from semi-structured data which is a complex task. In this paper we address the problem of computing structural similarity of XML documents which play a crucial role in clustering process. Previous works on path based approach fail to capture the sibling relationship among the nodes and also ignore the similarity when the nodes in the paths to be matched, are not in the same order but still convey same semantics. Another weakness of this approach is in the case of the partial path match, is that the level information is not taken into account when the nodes to be compared appear in different hierarchical level. To address these issues, we describe a method based on Prufer Sequence for measuring the structural similarity of XML documents, in this paper. Benefit of Prufer sequence based representation is that, it stores the ancestor-descendant and sibling relation. XML trees are encoded based on Prufer sequence which establishes a one-to-one mapping between XML tree and sequence. Instead of extracting all paths only common nodes are extracted based on Prufer sequence code. We have devised an algorithm to compute similarity by exploring all relations among the common nodes namely parent-child, ancestor-descendant and sibling. The experimental results show that the proposed approach is effective.

Keywords: XML, Structural Similarity, Prufer Sequence.

1 Introduction

Due to the increasing usage of XML documents, efficient techniques are needed to manage high volume of XML data, so that it can be used in many web based applications. Semantic Web technologies RDF, RDF Schema and OWL are based on XML and study of the structural similarity of XML documents are easily extensible to ontology matching. Two important ontology matching methods are Linguistic Similarity Matching (LSM) and Structure Similarity Matching (SSM). Wang et al. [18] stress the importance of structural information of ontology in case of ontology mapping. In this paper, our focus is on matching similar XML documents based on structural semantic relations. Main objective of this study is to integrate heterogeneous XML documents so that the information is more relevant to the user. Our proposed algorithm is capable of finding structural relations among XML elements and hence we can use it

Y.T. Demey and H. Panetto (Eds.): OTM 2013 Workshops, LNCS 8186, pp. 639–648, 2013.

to find the structural relations among concepts in ontology mapping. Grouping similar XML document gains much attention in research communities [1]. Since XML documents are hierarchical in nature grouping /clustering of similar XML documents is a challenging task [2]. Challenges are mainly due to the heterogeneous nature of XML documents (i.e.) XML documents are from different sources and they do not follow a common DTD/Schema. The key factor is to find an efficient technique for computing similarity between XML documents.

XML document similarity, in general, can be classified as structure, content and semantic. The term semantic is applicable to content, element name and structure. In this paper our focus is on computing structural similarity by considering semantics. By structural similarity we mean that the nodes of XML documents to be compared have same hierarchical order. However, we observe that sometimes even though the nodes of the paths to be compared are in different hierarchical order and still the semantics are not affected. For example the paths ***actor→movie →title*** and ***movie → title → actor*** are semantically the same. Also two nodes that have parent-child relationship in one document may have ancestor-descendant or sibling relationship in another document and still they are semantically similar.

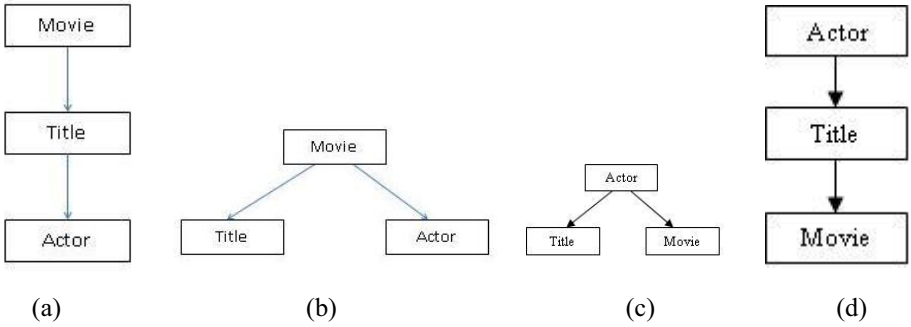
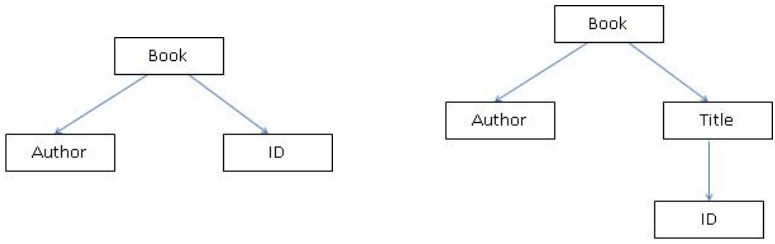


Fig. 1. XML document Structural Similarity example

For example in Fig 1, the elements ***title*** and ***actor*** have parent-child relation in document (a), sibling relation in document (b) and inverse parent-child relation in both document (c) and document (d) (i.e. when compared with document (a)) and in all the cases they have similarity based on semantics.

Another issue in structural similarity is the partial match. For example in Fig 2 the path ***Book→title→id*** in document (b) is partially mapped with the path ***Book→id*** in document (a). Here the ***id*** element occurs at different levels in both documents and to compute the similarity accurately, level weight should be considered.



(a) (b)

Fig. 2. Partial Match example

From the above examples we observe that the following factors are to be considered to compute XML document structural similarity using path based approach.

1. Sibling relation.
2. Ancestor–descendant relation.
3. Nodes in two paths are the same but in different hierarchical order.
4. Level weights should be given to nodes in case of partial match.
5. Extraction of common paths.
6. To capture all relations between nodes an efficient XML representation is needed.

Existing path based approaches handle some of the above issues but not all. A comparison is provided in Table 1.

Table 1. Comparison of various path based approaches

Approach	Extraction of paths	Sibling Match	Ancestor-Descendant Match with Level weight	Nodes in different hierarchical order	XML Representation
Rafiei et al. [3]	All paths including subpaths	No	No	No	Path Set
Buttler et al. [4]	Reduction of paths based on Shingle	No	No	No	Reduced path sets in terms of hash values
Joshi et al.[5]	All Xpaths	Partial	No	No	Xpath Set
Zhang et al.[6]	Paths as closed frequent association tag sequence	Yes	Yes, without level weight	No	Frequent Association tag sequence (FATS) set

2 Related Work

Earlier approaches for finding structural similarity between XML documents are based on tree edit distances. In this approach XML documents are represented as ordered labeled trees and they aim to compute cheapest edit sequence operations that can transform one tree to another. Tai [7] was first to introduce non exponential algorithm for finding minimum edit distance between trees. Shasha et al. [8] provides an edit distance algorithm in which they allow insertion, deletion and relabeling of single nodes anywhere in the tree, however insertion and deletion of entire sub trees were not considered. Later Chawathe [9] framed an algorithm with time complexity $O(n^2)$ which considers the edit operations, insertion and deletion at the leaf node level and allows replacement of node anywhere in the tree but disallows the move operation. Identification of sub tree similarities was considered by Nierman et al. [10], however the overall complexity of the algorithm is $O(n^2)$. Combined methods of tree edit distance and semantic similarity based on information retrieval was proposed by Tekli et al. [11]. In general XML document similarity based on tree edit distance needs to compute pair wise similarity and hence there is huge computational complexity and has scalability issue.

Various algorithms were proposed to measure XML document based on path [3,4,5]. Buttler [4] shows that the similarity comparison can be reduced to constant time complexity by using path Shingle technique. Rafiei et al. [3] shows that two XML documents are similar if they have more common paths in their path sets and they have a linear time complexity. Joshi et. al [5] uses XPaths which captures the sibling information and the experimental results shows improved clustering results when compared with normal path based approaches. However in all these approaches semantic structural similarity is not considered. Vinson et. al [12] developed an algorithm PathSim which uses edit distance algorithm to compute similarity and does not consider the structural similarity with semantics. Choi et. al [13] proposed a clustering method called PSim based on path similarities of XML data. The clustering process was more efficient and scalable than the previous approaches, however the structural similarity with semantics is not considered. Zhang et. al [6] proposed a method for clustering XML documents by computing similarity measure based on mining frequent association tags. Here, all types of relation among the nodes are considered, the process, however fails to detect when the hierarchical order of node changes and also the similarity measure doesn't consider level weight of nodes.

3 XML Document Representation Based on Prufer Sequence

In this section we introduce some basic definitions and concepts to model XML document tree as Prufer Sequence, benefits of Prufer Sequence based representation and the method of generating prufer sequence.

3.1 Basic Definitions

Definition 3.1 [XML Document Tree]

An XML document is an rooted ordered labeled directed Tree $T=(R_T, N_T, E_T, L_T)$ where R_T is the root node of the tree T, N_T is finite set of nodes, $E_T \subseteq [N_T]^2$ is set of edges and each node $n \in N_T$ has a label from L_T.

Definition 3.2 [Parent – Child]

Let n_i , $n_j \in N_T$. Then ni ,nj have Parent-Child relation if $(n_i,n_j) \in E_T$.

Definition 3.3 [Path]

A Path from node n_1 to n_k is a sequence of nodes $<n_1,n_2,\ldots n_k>$ such that n_i is the Parent of n_{i+1} for $1 \leq i < k$.

Definition 3.4 [Ancestor-Descendant]

If there is a Path from the node n_i to n_j then the node pair (n_i , n_j) is said to have Ancestor-Descendant relation.

Definition 3.5 [Sibling]

Let n_1 , $n_2 \in N_T$. Then n_1 and n_2 are said to have Sibling Relation if $(n_1,n_p) \in E_T$ and $(n_2,n_p) \in E_T$ where n_p is parent of both n1 and n2. If n_2 has greater order number than n_1, then n_2 is said to be Right-Sibling of n_1.

Definition 3.6 [Inverse Parent-Child]

Let T_1 and T_2 be two trees. Two nodes ni and nj are said to have Inverse Parent-Child relation if $(n_i,n_j) \in E_{T1}$ and $(n_j,n_i) \in E_{T2}$.

3.2 Benefits of Prufer Sequence Based Representation

Many techniques are available to represent structural information of XML documents and some of them are Level Structure, Level Edge and Path Set. The main drawback of these techniques is all structural relations cannot be represented. Level Structure method fails to capture parent/child or sibling relations, Level Edge fails to capture partial relations and Path set method fails to capture sibling relations. But prufer sequence based representation overcome these limitations by capturing all structural relations namely parent-child, ancestor-descendant and sibling.

3.3 Construction of Prufer Sequence

In order to capture structural relation among nodes, XML documents are represented based on Prufer Sequence code. Prufer [14] proposed an algorithm that describes the one-to-one mapping between tree and sequence by deleting one node from the tree at a time until two nodes is left. To construct Prufer sequence, traversal methods like pre-order, post-order can be used such that every node is assigned a unique number between 1 and n where n is the total number of nodes. A modified Prufer sequence was proposed by Rao et al. [15] in which a XML document tree T_1 is modified by adding a dummy node as the child for every leaf in T_1. Using Post Order traversal, nodes in T_1 are numbered from 1 to n called Post Order Traversal number (POTN). Then T_1's Prufer sequence is constructed by deleting leaf node with the smallest number in T_1 in the increasing order of post-order number and the number of its

parent is recorded. In the resulting tree this process is repeated until only the root node is left. The resulting sequence is called Numbered Prufer Sequence (NPS) and the associated node label is called Labeled Prufer Sequence(LPS). Thus an XML document is represented using NPS and LPS. Ontology matching based on prufer sequence was described by Algergawy et al. [19], which construct prufer sequence of ontologies as explained above and the structural similarity was computed by considering three types of node contexts.

3.4 Example for Prufer Sequence

Consider the XML document tree T_1 describing actor information which is shown in Fig 3. Post order traversal is done and each node is associated with Post Order Traversal Number (POTN) which is shown as circle. Dummy nodes are inserted for all leaf nodes which are shown as dashed lines. Prufer sequence is generated as explained above and the corresponding LPS and NPS are shown in Table 2.

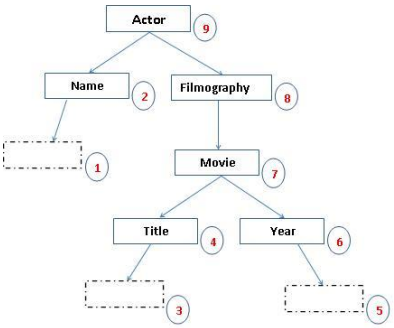


Fig. 3. XML Document Tree T_1 with POTN

Table 2. Prufer Sequence of XML document Tree T_1 with LPS and NPS

LPS	Name	Actor	Title	Movie	Year	Movie	Filmography	Actor
NPS	2	9	4	7	6	7	8	9

3.5 Algorithm for XML Document Structural Similarity

Given two XML document trees T_1 and T_2, based on section 3.3 prufer sequence is generated. Then the algorithm for computing structural similarity between T_1 and T_2 is given below.

Algorithm StructSim(T_1,T_2)
Input: T_1,T_2 // T_1,T_2 are XML Document Trees $T_1$and T_2
Output: Structural Similarity between T_1 and T_2
1 $LPS_1[1..m]$ ← Labeled Prufer Sequence of T_1 with 'm' nodes
2 $NPS_1[1..m]$ ← Numbered Prufer Sequence of T_1
3 $LPS_2[1..n]$ ← Labeled Prufer Sequence of T2 with 'n' nodes

4 NPS_2 [1..n] ← Numbered Prufer Sequence of T_2

5 CN[1..k]← $LPS_1 \cap LPS_2$ /* CN – Common nodes between T_1 and T_2, k is the number of common nodes*/

6 **if** k>1 then go to step 7
else
$Sim(T_1, T_2) \leftarrow 0$
go to step 17
end if

7 CN_1[1..k] ← Sort(CN) /*Sort Common nodes of Tree T_1 based on NPS_1 */

8 CN_2[1..k] ← Sort(CN) /* Sort Common nodes of Tree T_2 based on NPS_2 */

9 **for** index i varying from 1 to k-1 do
begin
Form the Relation Set $\mathfrak{R}_1$ from the set of common nodes CN_1 as follows
$\mathfrak{R}_1$={(CN_1[i], CN_1[j]) / CN_1[i]$\mathfrak{R}$$CN_1$[j], 1<j<=k and j > i)
Where $\mathfrak{R}$ is Parent-Child Relation
If Parent-Child Relation is not found then find immediate Ancestor-Child Relation and all Right-Sibling Relation based on NPS1
end for

10 Similarly form the Relationt $\mathfrak{R}_2$ from the set CN_2 based on NPS_2 as follows
$\mathfrak{R}_2$={(CN_2[i], CN_2[j]) / CN_2[i]$\mathfrak{R}$$CN_2$[j], 1<j<=k and j > i)

11 $\forall$(CN_2[i],CN_2[j])$\in\mathfrak{R}_1$ find exact match relations in $\mathfrak{R}_2$.
$\mathbf{E_{sim}}$ ← number of exact matches

12 Eliminate all exact match relations in both $\mathfrak{R}_1$ **and** $\mathfrak{R}_2$

13 $\forall$ (CN[i],CN[j])$\in\mathfrak{R}_1$ **and** $\mathfrak{R}_2$,
find the weighted similarity using the formula
$\mathbf{W_{Sim}} = \sum_{n=1}^{p} W_n$, where p is the number of remaining pair of relations in $\mathfrak{R}_1$ **and** $\mathfrak{R}_2$ and W_n is given by

$$Wn = \frac{LT_1(CN[i]).LT_2(CN[i]) + LT_1(CN[j]).LT_2(CN[j])}{[max(LT_1(CN[i], LT_2(CN[i])]^2 + [max(LT_1(CN[j], LT_2(CN[j])]^2}$$

Where
LT_i(CN[i]) - Level weight of CN[i] in tree T_i
Level weight of a node n in a tree T_i is found as $LT_i(n) \leftarrow$ level(n)/ height(T_i)

14 Root similarity is computed as
$\mathbf{R_{Sim}}$ = RSim1 + RSim2, where RSim1 and RSim2 is 0 if root is not matched and 1 if root is matched

15 Common node similarity is computed by the formula

$$\mathbf{C_{Sim}} = \frac{E_{Sim} + W_{Sim} + R_{Sim}}{E_n + P_n + 2}$$

E_n number of exact match, P_n number of partial match and 2 is added for root similarity match

16 Overall similarity is computed by the formula

$$\mathbf{Sim}(\mathbf{T_1}, \mathbf{T_2}) = C_{Sim} \cdot \frac{|LPS_1 \cap LPS_2|}{|LPS_1 \cup LPS_2|}$$

17 End

4 Experimental Results and Analysis

Experiments were performed to prove the effectiveness of our algorithm. Evaluation metrics namely Precision, Recall and F measures are used to compare our algorithm with other competitive algorithms. We have used two real time datasets. The first dataset is from Niagara project experimental data set namely movies and actors which we have downloaded from the web site **http://www.cs.wisc.edu/niagara/data**. The second data set is from ACM SIGMOD Record and DBLP. The dataset information is provided in Table 3.

Table 3. Real time XML Document datasets

Data Sets	Number of XML Documents
Movie Database #1	490
Actor Database #1	477
DBLP #2	3104
ACM SIGMOD #2	6150

Clustering is done by agglomerative hierarchical clustering with complete linkage which groups clusters at different levels. The principle behind agglomerative hierarchical clustering is that, it begins with as many clusters as objects and clusters are successively merged until one cluster remains. The clustering results were evaluated based on average precision, average recall and average F measure. First experiment is based on Movie and Actor Database. We have chosen this dataset because the movie and actor XML documents have semantic structural relationship. The comparison is done with XCLUST [17] and XEDGE [16] and the results are depicted in Fig 4.

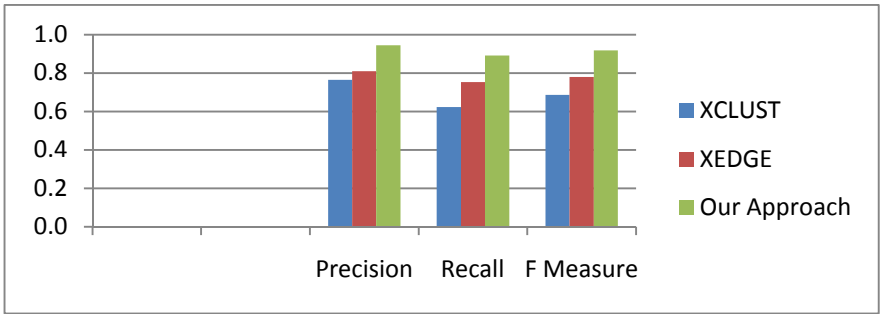


Fig. 4. Precision, Recall and F measure comparison for Actor and Movie dataset

From Fig.4 we observe that our approach reports better precision and recall when compared with XCLUST and XEDGE clustering techniques. We achieved 94% average precision, 89% average recall and 91% and average F measure when compared with XEDGE which reports 80% average precision, 75% average recall and 77% average F measure. The XCLUST algorithm reports still reduced values than XEDGE. This is mainly because XCLUST does not consider hierarchical relations. The drawback of XEDGE is that it does not consider the paths in different hierarchical order. Hence our algorithm outperforms both XEDGE and XCLUST.

Our next set of experiment is based on second dataset namely DBLP and ACM SIGMOD record as mentioned earlier. We compared our algorithm with XCLUST [17], XEDGE [16] and FATS [6]. As in previous experiment we clustered the documents based on agglomerative hierarchical clustering and the results are depicted in Fig 5.

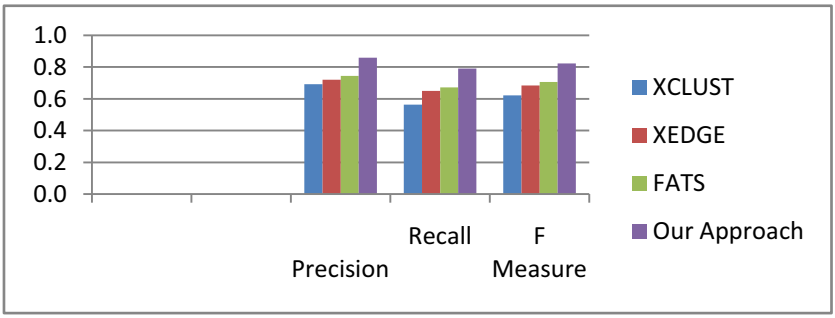


Fig. 5. Precision, Recall and F measure comparison for DBLP and Sigmod Dataset

From Fig 5 we observe that our approach outperforms three algorithms namely XCLUST, XEDGE and FATS. We further observe that the precision and recall are much less in XCLUST and XEDGE when compared with our algorithm. This is because sibling relations are not considered in both algorithms. FATS algorithm reports 74% precision and 67% recall, which is less than our algorithm. This is due to the fact that FATS algorithm doesn't consider the hierarchical weight in case of partial match.

5 Conclusion

In this paper we have proposed an effective algorithm for computing structural similarity of XML documents using Prufer Sequence. Main advantage of our algorithm is computing similarity by considering all structural relationships including structural semantics. Through experiments we have demonstrated that our algorithm is more effective than some of the previous works. In future we have planned to extend our work by computing both structural and content semantic similarity.

References

1. Tekli, J., Chbeir, R., Yetongnon, K.: An overview on XML similarity: background, current trends and future directions. Computer Science Review 3(3), 151–173 (2009)
2. Aggarwal, C.C., Ta, N., Wang, J., Feng, J., Zaki, M.: Xproj: a framework for projected structural clustering of xml documents. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 46–55. ACM (2007)
3. Rafiei, D., Moise, D.L., Sun, D.: Finding syntactic similarities between xml documents. In: 17th International Workshop on Database and Expert Systems Applications, DEXA 2006, pp. 512–516. IEEE (2006)
4. Buttler, D.: A short survey of document structure similarity algorithms. United States. Department of Energy (2004)
5. Joshi, S., Agrawal, N., Krishnapuram, R., Negi, S.: A Bag of Paths Model for Measuring Structural Similarity in Web Documents. In: Proceedings of the ACM SIGKKD Conference on Knowledge Discovery and Data Mining, USA, pp. 577–582 (2003)
6. Zhang, L., Li, Z., Chen, Q., Li, X., Li, N., Lou, Y.: Mining frequent association tag sequences for clustering XML documents. In: Sheng, Q.Z., Wang, G., Jensen, C.S., Xu, G. (eds.) APWeb 2012. LNCS, vol. 7235, pp. 85–96. Springer, Heidelberg (2012)
7. Tai, K.: The tree-to-tree correction problem. Journal of the ACM, JACM (1979)
8. Shasha, D., Zhang, K.: Approximate tree pattern matching, Pattern Matching in Strings, Trees and Arrays, ch. 14. Oxford Univ. Press (1995)
9. Chawathe, S.: Comparing hierarchical data in external memory. In: Proceedings of the International Conference on Very Large Databases, pp. 90–101 (1999)
10. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in XML documents. In: Proc. of ACM SIGMOD WebDB, pp. 61–66 (2002)
11. Tekli, J., Chbeir, R.: A novel XML document structure comparison framework based-on sub-tree commonalities and label semantics. J. Web Sem. 11, 14–40 (2012)
12. Vinson, A.R., Heuser, C.A., da Silva, A.S., De Moura, S.: An Approach to XML Path Matching. In: The 9th Annual ACM International Workshop on Web Information and Data Management, pp. 17–24 (2007)
13. Choi, I., Moon, B., Kim, H.-J.: A clustering method based on path similarities of XML data. Data & Knowledge Engineering 60(2) (2007)
14. Prufer, H.: Neuer beweis eines satzes uber permutationen. Archiv für Mathematik und Physik 27, 142–144 (1918)
15. Rao, P., Moon, B.: PRIX:Indexing and querying XML using Prufer Sequences. In: Proceedings of ICDE (2004)
16. Antonellis, P., Makris, C., Tsirakis, N.: XEdge: clustering homogeneous and heterogeneous XML documents using edge summaries. In: Proceedings of the 2008 ACM Symposium on Applied Computing, pp. 1081–1088. ACM (2008)
17. Nayak, R.: Fast and effective clustering of XML data using structural information. Knowledge and Information Systems 14(2), 197–215 (2008)
18. Wang, Y., Liu, W., Bell, D.A.: A structure-based similarity spreading approach for ontology matching. In: Deshpande, A., Hunter, A. (eds.) SUM 2010. LNCS, vol. 6379, pp. 361–374. Springer, Heidelberg (2010)
19. Algergawy, A., Schallehn, E., Saake, G.: A sequence-based ontology matching approach. In: Proceedings of 18th European Conference on Artificial Intelligence Workshops (2008)