Finding Syntactic Similarities Between XML Documents

Davood Rafiei
Dept. of Computing Science
University of Alberta, Canada
drafiei@cs.ualberta.ca

Daniel Moise
Dept. of Computing Science
University of Alberta, Canada
moise@cs.ualberta.ca

Dabo Sun
Dept. of Computing Science
University of Alberta, Canada
dabo@cs.ualberta.ca

ABSTRACT

We present a concise and accurate structural summary of XML documents and show that this summary can be used to effectively cluster documents that belong to a structurally similar class. We present efficient formulations of similarity between structural summaries that leads to a better detection of documents that conform to the same DTD. Our formulation is based on the intuition that two documents are likely to be generated by the same DTD if a large fraction of paths in the two documents are the same or similar. Our experimental evaluation shows that this method does an excellent job of grouping documents generated by the same DTD, outperforming some of the previously proposed solutions based on a tree comparison.

Categories and Subject Descriptors

H.3.1 [**Information Systems**]: Information storage and retrieval—*content analysis and indexing*

1. INTRODUCTION

There is a large and still growing number of applications that use the eXtensible Markup Language (XML) for data exchange. Storing documents generated by two or more applications in a database can be challenging as data may not conform to a non-trivial unifying DTD or the unifying DTD may be too complicated. A possible solution is to group documents that conform to the same or similar DTDs together before before storing them.

Grouping similar XML files together can lead to a better storage mapping and indexing [5, 9]. Without a proper grouping, elements under structurally similar paths can be scattered at different locations in the storage device, thus making the retrievals inefficient. Furthermore, more specific and accurate DTDs can be constructed for documents in a cluster, which in turn can be useful in query evaluation and can limit the access to only the relevant portions of data.

If the structure of an XML document is described as an ordered tree, there is considerable past [10, 2, 11] and more recent [7] work on finding the edit distance between ordered trees which are applicable. These algorithms often use dynamic programming to find the edit distance between two ordered trees and may vary in the set of edit operations or their weightings. In the presence of a training dataset, Zaki and Aggarwal propose a rule-based classifier that relates frequent ordered tree structures in an XML document to class labels [15].

There is past work that adopt a data centric view of an XML document, i.e. element ordering is not relevant. A hierarchical clustering algorithm based on common parent-child tags between documents is proposed by Lian et al. [6]. Theobald et al. [13] use parent-child tags, pairs of tags and content terms, twigs and the semantic relationships between terms (defined by Wordnet [14]) to classify each XML document into one of a few known classes.

In this paper, we take a data-centric view of an XML document and transform the structural description of a document to a concise set of paths and the frequency of each path. The transformation in general is lossy but avoids a tree matching problem which is often costly. When the structure of a document is summarized in the form of a set, standard set comparison techniques can be applied.

Our experiments on both real and synthetic data show that our method, when used for clustering XML documents generated by the same DTD, results in fewer misclusterings and is much faster than tree-based approaches.

2. STRUCTURAL SUMMARY

An XML document can be modeled as a node-labeled directed tree [1] where each node in the tree represents either an element or an attribute in the corresponding XML document. When a node represents an element, the node is labeled with the tag name of the element and when a node represents an attribute, it is labeled with the name of the attribute. Each edge of the tree represents a hierarchical inclusion relationship between either two elements or an element and an attribute. Since we are only interested in the structure of an XML document, we ignore other possible node types such as data values, comments and processing instructions. We refer to this tree description of an XML document as a *structure tree* [2].

DEFINITION 1. *A structure tree for an XML document d*

ACM Conference on Hypertext and Hypermedia 2005

[1] Two special attributes ID and IDREF, when present, may not be properly represented in a tree.
[2] This is an adaptation of *dataguide* [3] for XML documents.

is a tree t such that for every path in d there is a corresponding path in t and vice versa.

As an example, Figure 1 shows an structure tree for the following XML document.

```
<people><person><name>Tom</name><address>UofA
</address></person> <person><name>Cat</name>
<age>18</age></person></people>
```

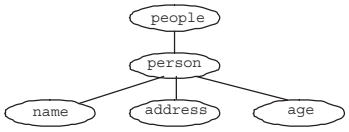

Figure 1: A structure tree

3. STRUCTURAL SIMILARITY

Given two XML documents, we want to find out if the two documents are structurally similar irrespective of the order in which the elements appear in each document. A solution is to compare the respective structure trees of the two XML documents. However, an XML document can have more than one structure trees and it is not clear which one should be used for a comparison. Furthermore, the problem of finding the edit distance between two unordered trees is NP-complete [16].

Our proposed solution is to describe the structure of an XML document as a set of paths and to avoid a tree matching problem. Even though an XML document can have more than one structure trees, they all have the same set of paths in common with the document (this is directly inferred from the definition of a structure tree). For each XML document, we extract from its structure tree every path that starts from the root and ends at a leaf. These paths here are referred to as the *root paths*.

To count for similar but not identical paths between two XML documents, we extract for each XML document in addition to the root paths all subpaths of the root paths (i.e. a consecutive sequence of tag names); we refer to the union of root paths and subpaths as *the path set* of and XML document. The path set of our working example is: { *people/person/name*, *people/ person/address*, *people/person/age*, *people/person*, *person/name*, *person/address*, *person/age*, *people*, *person*, *name*, *address*, *age* }. Furthermore, to count for paths that may appear more frequently, we further extract from the original XML document the frequency of each path.

Given the features of an XML document as a set of *(path, frequency)*, we can use standard set comparison techniques to find out if two documents are similar; the equality operation between two paths is the standard case-insensitive comparison of two strings. Informally, we call two XML documents similar if a large fraction of the paths in their path sets are the same. There are a number of ways of computing such an overlap including the Jaccard Coefficient or its extensions, the Dice Coefficient and the well-known *Cosine* measure (see an information retrieval text such as [8] for more details). Given the pair-wise similarity between all documents in the collection, a clustering algorithm can be applied to group similar XML documents into clusters, where every cluster should ideally represent a set of XML documents that share the same DTD.

4. EXPERIMENTS

To evaluate the effectiveness of our approach we ran experiments using both real and synthetic data. Our real data, referred to here as *RE*, was the online XML version of the ACM Sigmod Record from March 1999 [12]. This collection contained XML files shared among four DTDs: *ProceedingsPage*, *IndexTermsPage*, *OrdinaryIssuePage* and *SigmodRecord*. The collection had 989 XML files with a total size of 3.35 MB.

For synthetic data, we selected all DTDs reported by Nierman and Jagadish [7] except one called *HealthProduct.dtd* which we couldn't obtain. This set of DTDs here are referred to as *DTD set A*. To further test our method, we also used an extended set of DTDs which included *DTD set A* and 5 additional DTDs. We refer to this set of DTDs as *DTD set B*[3]. Using the IBM XML generator [4], we generated 100 XML files for each DTD and each setting of the parameters M (the maximum repeat of an element marked with a '+' or '*') and P (the probability that an optional attribute appears). We grouped the files into eight data sets:

```
DS1: M=4,P=0.75,DTD set A  DS2: M=4,P=1,DTD set A
DS3: M=8,P=0.75,DTD set A  DS4: M=8,P=1,DTD set A
DS5: M=4,P=0.75,DTD set B  DS6: M=4,P=1,DTD set B
DS7: M=8,P=0.75,DTD set B  DS8: M=8,P=1,DTD set B
```

Our dataset was deliberately chosen as a superset of the datasets used by Nierman and Jagadish [7] so that we could compare our results not only with those of Nierman and Jagadish and also with the results of the algorithms suggested by Chawathe [1] and Shasha et al. [11] without implementing these algorithms. Note that our synthetic data is generated randomly but using the same parameters as in [7], so the two datasets may not be identical.

4.1 Evaluation

Ideally, we want to have all documents conforming to the same DTD be clustered together, but in practice this may not be the case. To measure the effectiveness of each method, we use the same notion of mis-clustering introduced by Nierman and Jagadish, i.e. the minimum number of the documents that can be moved in order to obtain all documents conforming to the same DTD in the same cluster. The hierarchical agglomerative clustering algorithm in the R project for statistical computing[4] was used to cluster our data sets. The result of the clustering is a dendrogram, showing the clusters that collapse in each step. We use the number of mis-clusterings reported by Nierman and Jagadish over data sets DS1-DS4 and the Sigmod collection as a base line to compare the performance of our approach against theirs and those of Chawathe [1], Shasha [11] and Tag Frequency [7].

Table 2 shows the number of mis-clusterings obtained using our approach when each document is represented as a binary vector (BV), a frequency vector (FV) and a normalized frequency vector (NFV) of the path occurrences. The *Cosine* measure used to final the similarity between two documents. Using the path frequencies improves the clustering

[3]DTDs are available at www.cs.ualberta.ca/~drafiei/dtds.html.
[4]www.r-project.org

	RE	DS1	DS2	DS3	DS4
Nierman	0	10	2	11	9
Chawathe	3	16	8	30	25
Shasha	3	16	9	32	39
Tag Frequency	3	22	21	35	40

Table 1: Number of mis-clusterings reported by Nierman and Jagadish

	RE	DS1-DS4	DS5	DS6	DS7	DS8
BV	0	0	33	30	25	29
FV	0	0	0	0	0	0
NFV	0	0	0	0	0	0

Table 2: Number of mis-clusterings using our approach

accuracy. We also did similar experiments using both the Jaccard and the Dice Coefficients. The results were either comparable or worse than the Cosine measure, thus it was not reported.

One question is if we really need to extract paths and if we can obtain similar results by only using the tag frequencies and perhaps with some variations of the similarity measure. Since the Cosine measure performs the best in our experiments, we choose this measure for tag frequencies. We also pick two additional distance functions, namely the city-block distance because it is previously used [7] as reported in Table 1 and the Euclidean distance. Table 3 shows that the tag frequency is not enough to compare the structural similarity between XML documents.

We found some inconsistencies between our results for the tag frequency and the results reported earlier using the city-block distance [7]. One possible explanation is that we maybe counting the number of mis-clusterings differently. In our case, the number of mis-clusterings are counted manually, but we are not sure how this is done in the earlier reporting. Since our counts of the mis-clusterings are higher than those reported by Nierman and Jagadish, we feel justified to say that we are over-estimating the number of mis-clusterings. Thus, our approach outperforms previously-proposed methods.

5. DISCUSSIONS

We have proposed a simple and yet efficient approach to find the structural similarity between XML documents. We have also evaluated our approach with various data sets. Given a document with n root paths, each of length l, there are $nl(l+1)/2$ possible subpaths. Thus, the time complexity is expected to be $O(nl^2)$. On a modest hardware (Pentium 4, 2.8GHz CPU and 1GB RAM), the path extraction (in Java) for the Sigmod collection with 989 documents took 20 seconds. Computing the pair-wise distances (in C++) between all these documents took only 98 seconds.

A limitation of our approach is when we want to detect a similarity between documents with the same structures but different tag names. We expect this to be less problem with an increasing use of the namespaces and also the same tag names to refer to the same concepts. However, one solution is to allow users to specify some relabeling rules. For instance, if the tag names in one document are in French and the tag names in another document are in English, a possible relabeling can be a word-to-word translation of the tag names.

	RE	DS1	DS2	DS3	DS4
City block	24	208	200	211	240
Euclidean	24	0	62	0	0
Cosine	68	38	33	39	35

Table 3: Number of mis-clusterings using the tag frequency

6. REFERENCES

[1] S. S. Chawathe. Comparing hierarchical data in external memory. In *Proceedings of the VLDB Conference*, pages 90–101, Edinburgh, 1999.

[2] S. S. Chawathe, A. Rajaraman, H. Garcia-Molina, and J. Widom. Change detection in hierarchically structured information. In *Proceedings of the SIGMOD Conference*, pages 493–504. ACM Press, 1996.

[3] R. Goldman and J. Widom. Dataguides: enabling query formulation and optimization in semistructured databases. In *Proceedings of the VLDB Conference*, pages 436–445, 1997.

[4] IBM XML Generator. www.alphaworks.ibm.com/tech/xmlgenerator.

[5] R. Kaushik, P. Shenoy, P. Bohannon, and E. Gudes. Exploiting local similarity for indexing paths in graph-structured data. In *Proceedings of the ICDE Conference*, pages 129–140, 2002.

[6] W. Lian, D. W. Cheung, N. Mamoulis, and S. M. Yiu. An efficient and scalable algorithm for clustering xml documents by structure. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):82–96, 2004.

[7] A. Nierman and H. V. Jagadish. Evaluating structural similarity in XML documents. In *Proceedings of the WebDB Workshop*, Madison, June 2002.

[8] G. Salton. *Introduction to Modern Information Retrieval*. MacGraw Hill, 1983.

[9] H. Schöning. Tamino - a DBMS designed for XML. In *Proceedings of the ICDE Conference*, pages 149–154, 2001.

[10] S. M. Selkow. The tree-to-tree editing problem. In *In Information Processing Letters, 6(6)*, pages 184–186, 1977.

[11] D. Shasha and K. Zhang. Approximate tree pattern matching. In *Pattern Matching Algorithms*, pages 341–371. Oxford University Press, 1997.

[12] Sigmod Record in XML. www.acm.org/sigmod/record/xml/index.html.

[13] M. Theobald, R. Schenkel, and G. Weikum. Exploiting structure, annotation and ontological knowledge for automatic classification of xml data. In *Proceedings of the WebDB Workshop*, 2003.

[14] Wordnet:. a lexical database for the english language. www.cogsci.princeton.edu/ wn.

[15] M. J. Zaki and C. C. Aggarwal. Xrules: an effective structural classifier for xml data. In *Proceedings of the KDD Conference*, pages 316–325, 2003.

[16] K. Zhang, R. Statman, and D. Shasha. On the editing distance between unordered labeled trees. *Information Processing Letters*, 42(3):133–139, 1992.