Clustering XML Documents Using Frequent Edge-Sets

Zhiyuan Jin(✉), Le Wang, and Yanfen Chang

College of Computer Science, Dahongying University, Ningbo 315175, China
lanborokk@163.com

Abstract. Clustering of XML documents is a useful technique for knowledge discovery in XML databases. However, the process of clustering XML documents is always time-consuming due to the semi-structured characteristics of the documents. In this paper, we present an efficient clustering algorithm called Frequent Edge-based XML Clustering (FEXC) to cluster XML documents using *frequent edge sets*. First, we represent XML documents using edge sets, and then discover the frequent edge sets for each document employing a traditional frequent pattern mining approach. Second, for each frequent edge set, we find all the documents containing it, and then compute a measure called *entropy overlap*, which indicates the document relevance (overlap) with the ones containing all other frequent edge sets. Clustering is then performed using the entropy overlap measure. Third, we perform a merging process which removes redundant clusters, therefore reducing the number of clusters. Experimental results show that our proposed method outperforms the traditional distance-based XML clustering algorithm in terms of efficiency without compromising the quality of clustering.

Keywords: XML · Clustering · Frequent edge set · Semi-structured data

1 Introduction

In recent years, XML data have become ubiquitous with the rapid upsoaring in both number and scale of applications such as XML database systems, business transactions, XML middleware systems, and so on. Discovering knowledge from XML structural data has become more significant with the exponential growing of XML documents available on the Web. Among various approaches, XML document clustering is one of the useful mining approaches for knowledge discovery. The objective of XML document clustering is to group XML documents with similar characteristics together, which can be used in broad applications including web mining, information retrieval, querying, storage compression.

In this paper, we introduce a novel clustering approach, Frequent Edge-based XML Clustering (FEXC), which exploits discovery of edge set frequently occurring in XML documents in order to cluster documents. A frequent edge set is a set of edges occurring together and whose occurrence number is no less than a specified threshold in the XML document set. The intuition of our clustering criterion is that documents within same clusters share more frequent edge sets while documents belonging to different clusters share fewer frequent edge sets.

J. Abawajy et al. (eds.), *International Conference on Applications and Techniques in Cyber Security and Intelligence*, Advances in Intelligent Systems and Computing 580,
DOI 10.1007/978-3-319-67071-3_50

2 Clustering Algorithm

The cluster generation process using frequent edge sets and present the approach to disjointing clusters based on the concept of entropy overlap of clusters. We give the high level process of clustering algorithm including cluster construction and cluster merging. First, we mine frequent edge sets from the XML document sets. Then we generate clusters based on the discovered edge sets. Finally, to remove redundant clusters and reduce the number of clusters, we construct tree-like clusters and merge the similar clusters.

2.1 Constructing Disjoint Clusters

In Fig. 1 we present the algorithm for generating disjoint clusters adopting a method similar to the algorithm FTC. In each step, we select a frequent edge set with minimum entropy overlap until there not exist any frequent edge sets in the remaining sets or each document has been assigned to a cluster. Documents containing the selected edge set will constitute a new cluster. If an edge set has been chosen as a target cluster, we will remove all documents containing the selected edge set from the sets in the remaining sets. In case that two edge sets have the same overlap, we will consider the edge set with more edges as the selected one. Because more edges in the edge set means more information and we can describe the cluster more specifically.

```
Algorithm ConstructClusters (FES, doccount)
Input: Frequent Edge Sets
Output: Selected frequent edge sets
1       SelectedSets = Φ;
2       n = 0;
3       RemainFES = FES;
4       while RemainFES ≠ Φ and n < doccount do
5          for each ES ∈ RemainFES do
6             Calculate Entropy Cluster of Cluster for frequent edge set ES;
7          TargetES = the ES with minimum EOC and maximum |ES|;
8          SelectedSets = SelectedSets ∪ { TargetES };
9          RemainFES = RemainFES – {TargetES};
10         n = n + |doc(TargetES)|;
11         for each document d in doc(TargetES) do
12            for each ES ∈ RemainFES do
13               if d ∈ doc(ES)
14                  doc(ES) = doc(ES) – d;
15      return SelectedSets
```

Fig. 1. Clusters construction

2.2 Merging Clusters

In the previous section, we present the approach to forming non-overlapping clusters. However, it is likely to generate too many clusters using frequent edge sets. In some circumstances users may specify the number of final clusters which may be a small one. Therefore a merging process is demanded to produce less clusters through merging the similar clusters into a large one.

Due to the monotonicity property, documents containing frequent edge sets also contain the frequent sub edge sets. According to our clusters generated principle, documents in the cluster represented by the super edge set can also be represented by the sub edge set. Given two non-overlapping clusters labeled with A and B respectively, where A and B are two frequent edge sets. If the edge set A is a subset of the edge set B, cluster A is consider as a super-cluster of cluster B and cluster B is a sub-cluster of cluster A. This relationship of clusters can be exploited to construct the clusters using a tree-like diagram. If a cluster has more than one super-clusters, then the cluster with the most number of edges will be selected as the parent node in the tree-like clusters. In Fig. 2 we show an instance of tree-like clusters. The label for each cluster represents a frequent edge set. The label NULL of the root node represents an empty edge set which is a subset of all edge set. Clusters A1, A2, A3 are 3 sub-clusters of the cluster A, while clusters B1, B2 are 2 sub-clusters of the cluster B.

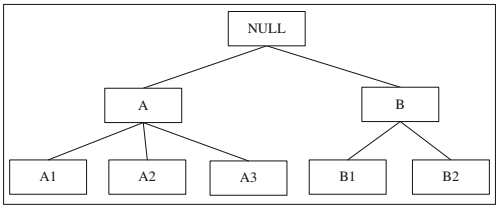

Fig. 2. Tree-like clusters

Clusters are merged into a new one through removing sub-clusters, i.e., documents in both super-clusters and sub-clusters are merged together. Now we discuss the criterion for merging similar clusters. Basically, we merge similar clusters by the goodness which describe the similarities between clusters. We define the cluster similarity as follows:

$$goodness(A, B) = \frac{|\{e \in (A \cap B)\}|}{|\{e \in (A \cup B)\}|}. \quad (1)$$

We measure the cluster similarity based on the number of the same edges in both of the frequent edge sets. That is, the more overlap in their frequent edges, the closer the two clusters.

A bottom-up sub-clusters pruning process is adopted to merge similar clusters. We remove the most redundant sub-clusters level by level from bottom to top. We present the algorithm for pruning sub-clusters in the tree-like clusters. First of all, we compute the goodness of each sub-cluster with its super-cluster at one level. Then we sort the

goodness of clusters in descending order and remove the clusters with larger goodness. We perform this process level by level until we reach the top of the tree or the count of the remaining clusters is equal to user's specified count.

3 Experiments

In this section, we present experimental results of our FEXC clustering algorithm compared to previous distance-based algorithms both on the clustering performance and clustering quality. We implemented both FEXC algorithm and tree distance-based algorithm in Java language (JDK1.5) and carried out all experiments on an Intel Xeon 2GHz computer with 2 GB of RAM running operating system RedHat Linux 9.0. For the tree distance-based XML clustering algorithm we adopt the computing method of tree distance similar to the algorithm presented by Dalamagas. In our frequent edge-based clustering algorithm, we adopt an idea the same as Eclat to generate the frequent edge sets.

Clustering Quality. We used the XML document generator to generate documents with varying document numbers from 400 to 20000. And for each DTD, we generate the same number of documents. The support threshold for mining frequent edge sets in FEXC algorithm is 20%. In Table 1(a) and (b) we present the number of generated clusters adopting two clustering algorithms for various document numbers and different parameters, where TDXC stands for the tree distance-based XML clustering algorithm and FEXC stands for frequent edge-based XML clustering algorithm. From Table 1, we can find that FEXC generates fewer clusters compared to TDXC, and XML documents are more centralized using FEXC algorithm. However, TDXC makes documents scatter in more clusters, which results in a poor clustering quality especially when employing cluster merging.

Table 1. Cluster count

Document count	TDXC cluster count	FEXC cluster count
(a) MaxRepeats = 4, NumberLevels = 7		
400	15	7
2000	20	10
4000	24	9
8000	31	9
12000	29	8
16000	43	9
20000	52	12
(b) MaxRepeats = 7, NumberLevels = 10		
400	18	8
2000	25	6
4000	31	9
8000	44	9
12000	37	7
16000	51	10
20000	66	15

Cluster Merging Quality. In some circumstances, the count of generated clusters is more than the user's expected result. A cluster merging process is demanded to reduce the cluster count. In our experiments, documents are generated from four DTDs. As a result, we specify 4 as the final cluster count. In Table 2 we show the precisions of the two algorithms respectively, which are computed as follows:

$$precision = \frac{\sum_{j=1}^{n} entry(j,j)}{\sum_{j=1}^{n}\sum_{k=1}^{n} entry(j,k)}. \tag{2}$$

where j and k represent the type of clusters, entry(j, k) denotes that the document belonging to the cluster k is assigned to cluster j. The equality between j and k means the right cluster result of the document. A better cluster merging quality can be seen in Table 2 employing FEXC. From Table 1, we know that TDXC makes documents more decentralized, and thus it is more likely to merge clusters by mistake. On the contrary, fewer clusters are produced using FEXC. Therefore, when merging similar clusters using the tree-like clusters, a higher precision can be obtained. In some cases, it even can reach 100%.

Table 2. Cluster precision

Document count	TDXC cluster precision (%)	FEXC cluster precision (%)
(a) Cluster precision for Table 1(a)		
400	99.33	100
2000	98.71	100
4000	98.41	100
8000	98.27	99.01
12000	98.58	99.33
16000	96.32	98.51
20000	93.23	98.12
(b) Cluster precision for Table 1(b)		
400	99.13	100
2000	99.21	100
4000	98.51	99.42
8000	97.39	100
12000	98.04	100
16000	94.25	99.23
20000	92.67	98.21

3.1 Clustering Performance

In Fig. 3 we present the performance of the two algorithms with various sizes of datasets from 400 to 20000 on different parameters for document generation. From the figure we can find the algorithm FEXC is faster than TDXC for clustering XML documents, and

the improvement of performance using FEXC is more obviously for the dataset with large size. Since the time consumed by the algorithm FEXC mainly depends on the mining process, and many existed efficient mining algorithms relate to the dataset size linearly. While in the algorithm TDXC, computations of tree distance play the most important part and spend the most of time. For a dataset with n documents, TDXC needs $n * (n - 1)/2$ times of comparisons between documents, which lead to an inefficient clustering process and a low scalability. When the dataset is on increase, the running time spent on TDXC increase drastically.

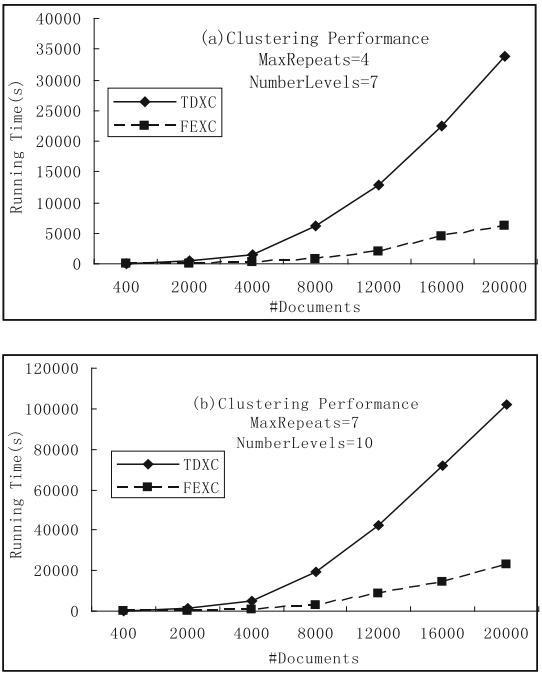

Fig. 3. Performance comparison

3.2 Clustering Scalability

In Fig. 4, the experiments show that our algorithm present good scalability. When the dataset is on increase, the running time doesn't rise fast. As mentioned earlier, the time consumed by our algorithm is determined by the mining process which is proportional to the size of dataset. Therefore, good scalability can be obtained using FEXC compared to TDXC.

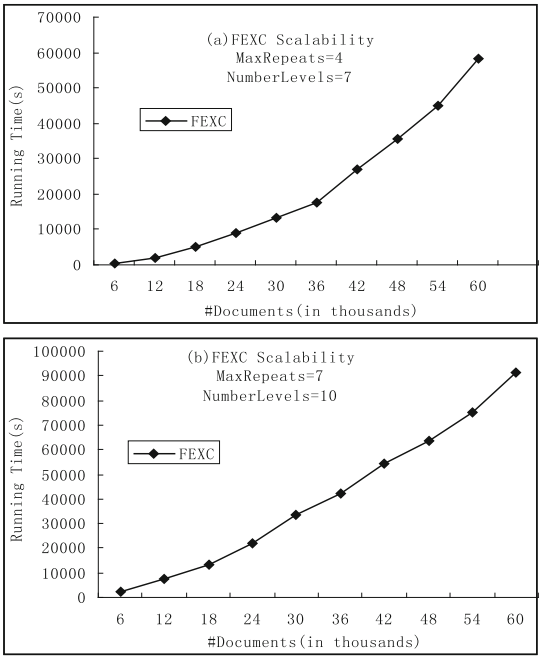

Fig. 4. Scalability of FEXC

3.3 Cluster Description

FEXC can automatically produce a description for each cluster. In Table 3 we show the cluster result of 4000 documents adopting FEXC. Descriptions of cluster are shown in the left of the table and the names of DTDs are presented in the right of the table. The description of each cluster (cluster label) contains all the edges in the frequent edge sets which represent the cluster.

Table 3. Cluster description

Cluster label	DTD
HomePage → volumes	HomePage.dtd
OrdinaryIssuePage → notes OrdinaryIssuePage → sections note → otherSources	OrdinaryIssuePage.dtd
ProceedingsPage → date ProceedingsPage → confYear article → title	ProceedingsPage.dtd
SigmodRecord → issues	SigmodRecord.dtd

4 Conclusion

In this paper, we present an efficient clustering algorithm called FEXC to cluster XML documents using frequent edge sets. The intuition of our clustering criterion is that documents within same clusters share common frequent edge sets while documents belonging to different clusters share few frequent edge sets. We discover frequent edge sets from the large set of documents and measure entropy overlap of each frequent edge set in order to construct clusters. To remove redundant clusters and reduce the count of clusters a cluster merging process is employed. Our experimental results show that FEXC outperforms tree distance-based clustering algorithm in terms of efficiency, but still presents the good quality of clustering.

References

1. Thakur, R.S., Jain, R.C., Pardasani, K.R.: Mining level-crossing association rules from large databases. J. Comput. Sci. **2**(1) (2006)
2. Koltsidas, H., Müller, H., Viglas, S.D.: Sorting hierarchical data in external memory. Proc. Vldb Endow. **1**(1), 1205–1216 (2008)
3. Beil, F., Ester, M., Xu X.W.: Frequent term-based text clustering. In: KDD, pp. 436–442 (2002)
4. Wong, K.F., Yu, J.X., Tang, N.: Answering XML queries using path-based indexes: a survey. World Wide Web **9**(3), 277–299 (2006)
5. Costa, G., Manco, G., Ortale, R., Tagarelli, A.: A tree-based approach to clustering XML documents by structure. In: PKDD, pp. 137–148 (2004)
6. Dalamagas, T., Cheng, T., Winkel, K.J., Sellis, T.K.: Clustering XML documents by structure. In: SETN, pp. 112–121 (2004)
7. Nayak, R., Iryadi, W.: XML schema clustering with semantic and hierarchical similarity measures. Knowl. Based Syst. **20**(4), 336–349 (2007)
8. Lee, M.L., Yang, L.H., Hsu, W., Yang, X.: XClust: clustering XML schemas for effective integration. In: CIKM, pp. 292–299 (2002)
9. Leung, H., Chung, K.F.L, Chan, S.C., Luk, R.W.P: XML document clustering using common XPath. In: WIRI, pp. 91–96 (2005)
10. Nierman, A., Jagadish, H.V.: Evaluating structural similarity in XML documents. In: WebDB, pp. 61–66 (2002)
11. Tagarelli, A., Greco, S.: Toward semantic XML clustering. In: SDM, pp. 188–199 (2006)
12. Wang, L., Cheung, D.W., Mamoulis, N., Yiu, S.M.: An efficient and scalable algorithm for clustering XML documents by structure. IEEE Trans. Knowl. Data Eng. **16**(1), 82–96 (2004)