View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Queensland University of Technology ePrints Archive

QUT Digital Repository:
http://eprints.qut.edu.au/

Kutty, Sangeetha and Tran, Tien and Nayak, Richi and Li, Yuefeng (2008) *Clustering XML Documents Using Closed Frequent Subtrees: A Structural Similarity Approach.* In: 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007 Dagstuhl Castle, December 17-19, 2007, Germany.

© **Copyright 2008 Springer**

Clustering XML Documents Using Closed Frequent Subtrees: A Structural Similarity Approach

Sangeetha Kutty, Tien Tran, Richi Nayak, and Yuefeng Li

Faculty of Information Technology,
Queensland University of Technology, Brisbane, Australia
{s.kutty,t4.tran,r.nayak,y2.li}@qut.edu.au

Abstract. This paper presents the experimental study conducted over the INEX 2007 Document Mining Challenge corpus employing a frequent subtree-based incremental clustering approach. Using the structural information of the XML documents, the closed frequent subtrees are generated. A matrix is then developed representing the closed frequent subtree distribution in documents. This matrix is used to progressively cluster the XML documents. In spite of the large number of documents in INEX 2007 Wikipedia dataset, the proposed frequent subtree-based incremental clustering approach was successful in clustering the documents.

Keywords: Clustering, XML document mining, Frequent Mining, Frequent subtrees, INEX, Structural mining.

1 Introduction

The rapid growth of XML (eXtensible Mark-up Language) after its standardization has marked its acceptance in a wide array of industries ranging from education to entertainment and business to government sectors. The major reason for its success can be attributed to its flexibility and self-describing nature in using structure to store its content. With the increasing number of XML documents there arise many issues concerning the efficient data management and retrieval. XML document clustering has been perceived as an effective solution to improve information retrieval, database indexing, data integration, improved query processing [8] and so on.

Clustering task on XML documents involves grouping XML documents based on their similarity without any prior knowledge on the taxonomy[10]. Clustering has been frequently applied to group text documents based on the similarity of its content. However, clustering XML documents presents a new challenge as it contains structural information with text data (or content). The structure of the XML documents is hierarchical in nature and it represents the relationship between the elements at various levels.

Clustering XML documents is a challenging task[10]. Majority of the existing algorithms utilize the tree-edit distance to compute the structural similarity between each pair of documents. This may lead to incorrect results as the calculated tree-edit distance can be large for very similar trees conforming to the same schema for different size

N. Fuhr et al. (Eds.): INEX 2007, LNCS 4862, pp. 183–194, 2008.
© Springer-Verlag Berlin Heidelberg 2008

trees [12]. A recent study showed that XML document clustering using tree summaries provide high accuracy for documents [3]. The structural summaries of the XML documents were extracted and used to compute the tree-edit distance. Due to the need of calculating the tree-edit distance between each pair of document structural summaries, this process becomes expensive for very large dataset such as INEX Wikipedia test collection that contains 48305 documents. This lays the ground to employ a clustering algorithm which does not utilise the expensive tree-edit distance computation.

In this paper, we propose CFSPC(Closed Frequent Structures-based Progressive Clustering) technique to cluster XML documents incrementally using the closed frequent subtrees. These closed frequent subtrees are called as the Pre-Cluster Form (PCF). Using the PCFs of the XML documents the global similarity between the XML documents is computed incrementally.

The assumption that we have made in this paper, based on the previous research [9] is that documents having a similar structure can be grouped together. For instance, the document from a publication domain will have a different structure than a document from a movie domain. Using this assumption we utilize only the hierarchical structure of the documents to group the XML documents. We have not included the content of the documents as it incurs a huge overhead in mining frequent trees and finding similarity between documents.

Rest of the paper is organized as follows: Section 2 provides the overview of the CFSPC method. Section 3 covers the pre-processing of XML documents for mining. Section 4 details the mining process which includes frequent mining and clustering. In Section 5, we present the experimental results and discussion. We conclude in Section 6 by presenting our future works in XML document mining.

2 The CFSPC (Closed Frequent Structures-Based Progressive Clustering) Method: Overview

As illustrated in Fig.1 CFSPC involves two major phases Pre-processing and Mining. The pre-processing phase involves extraction of the structure of a given XML document to obtain a *document tree*. Each *document tree* contains nodes which represent the tag names. The mining phase includes application of *frequent subtree mining* and

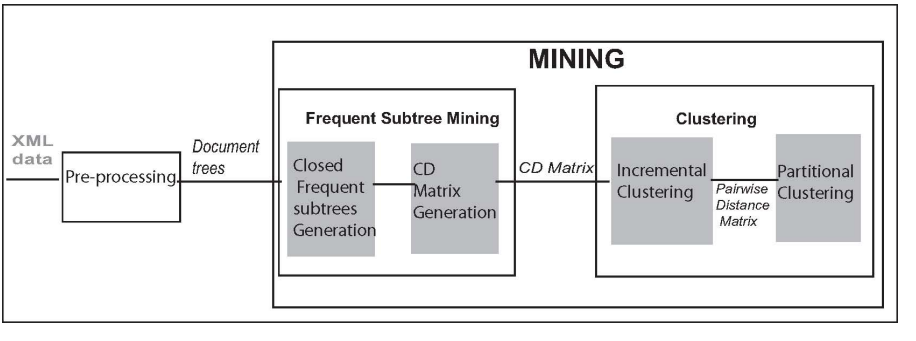

Fig. 1. The CFSPC Methodology

clustering. The frequent subtree mining determines the closed frequent subtrees from the *document trees* for a given support threshold. The closed frequent subtrees are condensed representations of the frequent subtrees. The distribution of the closed frequent subtrees in the corpus is modelled as a subtree-document matrix, $CD_{|CFS|\times|DT|}$, where *CFS* represents the closed frequent subtrees and *DT* represents the document trees in the given document tree collection. Each cell in the *CD* matrix represents the presence or absence of a given closed frequent subtree in the document tree. This matrix is used in calculating the similarity between documents.

As discussed earlier, the generation of distance matrix between each pair of documents is expensive for the INEX Wikipedia corpus due to its high dimension. Hence in the second phase of mining, the incremental clustering method is used to progressively cluster the documents in the corpus by comparing each document tree to the existing clusters. The similarity is measured by computing the Common SubTree coefficient (Ω) using the *CD* matrix based on the number of common closed frequent subtree between the document tree and existing clusters. Based on Ω, the document tree is grouped into an existing cluster with which it has the maximum Ω and greater than the user-defined cluster threshold otherwise the *document tree* is assigned to a new cluster.

As incremental clustering avoids the expensive pair-wise computation, it can cluster large data sets such as INEX 2007 Wikipedia dataset. However, this process results in undefined number of clusters according to the similarity measure used. In order to obtain the user-defined number of clusters, we utilize the pair-wise partitioning clustering algorithm [5]. The similarity between each pair of clusters is computed using Ω. Due to the reduced size of clusters, it is now computationally feasible to generate the pair-wise similarity matrix. This similarity matrix becomes the input to the partitional clustering algorithm. This algorithm generates the required number of clusters.

By combining the incremental and pair-wise clustering method, the CFSPC method is able to produce the clustering solution for the large data sets.

3 CFSPC Phase 1: Pre-processing

In the pre-processing phase, the XML document is decomposed into a tree structure with nodes representing only the tag names. The tag names are then mapped to unique integers for ease of computation. The semantic and syntactic meanings of tags are ignored. The Wikipedia documents conform to the same schema using the same tag set. Additionally previous research has shown that the semantic variations of tags do not provide any significant contribution in the clustering process [9, 10]. Other node information such as data types and constraints are also ignored.

There are several research works on clustering that use paths extracted from XML documents as a document representation and form the basis of calculating similarity between the documents[1, 10]. We have chosen to use the tree format to represent the XML documents. The tree format includes the sibling information of the nodes which is not included when an XML document is represented as a series of paths.

As shown in Fig. 2, the pre-processing of XML documents involves three subphases. They are namely:

1. Parsing
2. Representation
3. Duplicate branches removal.

3.1 Parsing

The XML data model is a graph structure comprising of atomic and complex objects. It can be modelled as a *tree*. Each XML document in INEX Wikipedia corpus is parsed and modelled as a rooted labeled ordered *document tree*. The document tree is *rooted* and *labeled* as there always exists a root node in the document tree and all the nodes are labeled using the tag names. The left-to-right *ordering* is preserved among the child nodes of a given parent in the document tree and therefore they are ordered.

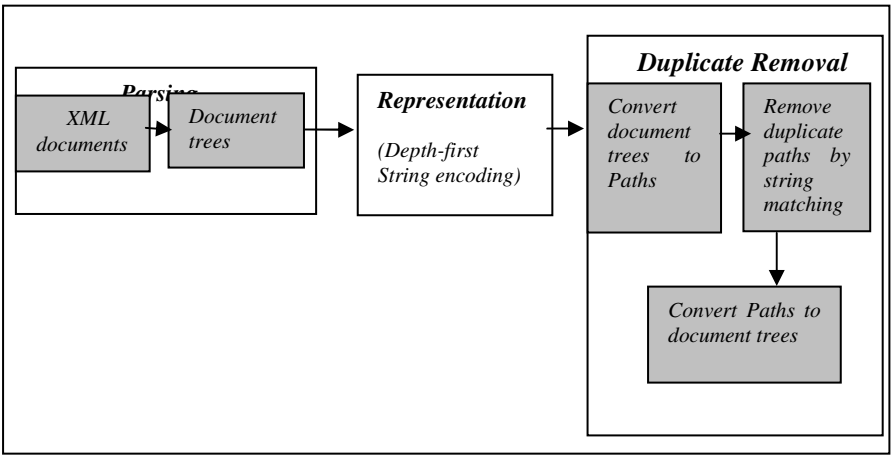

Fig. 2. The Pre-processing phase

3.2 Representation

The document trees need to be represented in a way that is suitable for mining in the next phase. A popular representation for trees, the depth-first string format[2], is used to represent the document trees. The *depth-first string encoding* traverses a tree in the *depth-first order*. It represents the *depth-first traversal* of a given document tree in a string like format where every node has a "–1" to represent backtracking and "#" to represent the end of the string encoding. For a document tree T with only one node r, the depth-first string of T is $S(T) = l_r\#$ where l is the label of the root node r. For a document tree T with multiple nodes, where r is the root node and the children nodes of r are $r_1,...,r_k$ preserving left to right ordering, the depth-first string of T is $S(T)= l_r\ l_{r_1}\text{-}1\ l_{r_2}\text{-}1 \ldots l_{r_k}\text{-}1\#$.

3.3 Duplicate Branches Removal

An analysis of the INEX Wikipedia dataset reveals that a large number of *document trees* contain duplicate branches. These duplicate branches are redundant information

and hence they could cause additional overhead in the mining process. In order to remove the duplicate branches, the document tree is converted to a series of paths. The duplicate paths of the document trees are identified using string matching and removed. The remaining paths are combined together to create the document trees without having duplicate branches.

4 CFSPC Phase 2: Mining

The mining phase includes two phases namely incremental clustering and pair-wise clustering. We first explain the generation of closed frequent subtrees from document trees. We then explain the process of clustering with the use of closed frequent subtrees.

4.1 Incremental Clustering

Frequent Subtree mining is first applied on the XML documents to identify closed frequent subtrees for a given user-specified support threshold. Closed frequent subtrees are condensed representations of frequent subtrees without any information loss[7]. Frequent subtree mining on XML documents can be formally defined as follows:

Problem Definition for the Frequent Subtree Mining on XML Documents
Given a collection of XML documents $D = \{D_1, D_2, D_3, \ldots, D_n\}$ modelled as document trees $DT = \{DT_1, DT_2, DT_3, \ldots, DT_n\}$ where n represents the number of XML documents or document trees. There exists a subtree $DT' \subseteq DT_k$ that preserves the parent-child relationship among the nodes as that of the document tree DT_k.

Support(DT') (or frequency(DT')) is defined as the number of document trees in DT where DT' is a subtree. A subtree DT' is frequent if its support is not less than a user-defined minimum support threshold. In other words, DT' is a frequent subtree in the document trees in DT such that,

$$frequency\ (DT')/|DT| \geq min\_supp \quad (1)$$

where min_supp is the user-given support threshold and $|DT|$ is the number of document trees in the document tree dataset DT.

Due to the large number of frequent subtrees generated at lower support thresholds, recent researchers have focused on using condensed representation without any information loss [6]. The popular condensed representation is the closed frequent subtrees which is defined as follows.

Problem Definition for Closed Subtree
For a given document tree dataset, $DT = \{DT_1, DT_2, DT_3, \ldots, DT_n\}$, if there exists two frequent subtrees DT' and DT'', the frequent subtree DT' is closed of DT'' iff for every $DT' \supseteq DT''$, $supp(DT') = supp(DT'')$ and there exists no superset for DT' having the same support as that of DT'. This property is called as *closure*.

In order to generate the closed frequent subtrees from the pre-processed document trees, the CMTreeMiner[2] is utilized. This algorithm adopts the apriori-based approach of generate-and-test to determine closed frequent subtrees. Having generated the closed frequent subtrees, their distribution in the corpus is modelled as a Boolean

subtree-document matrix, $CD_{|CFS|\times|DT|}$, where *CFS* represents the closed frequent subtrees and *DT* represents the document trees in the given document tree collection. Each cell in the *CD* matrix represents the presence or absence of a given closed frequent subtree $\{cfs_1, cfs_2,\ldots,cfs_l\}$ in the document tree $\{DT_1, DT_2, DT_3,\ldots,DT_n\}$. Fig. 3 shows a $CD_{|CFS|\times|DT|}$ with closed frequent subtree $\{cfs_1, cfs_2, cfs_3\}$ in the document trees $DT = \{DT_1, DT_2, DT_3, DT_4\}$.

This matrix is used to compute the similarity between the document trees for clustering. The column of *CD* matrix for each document tree is referred as Pre-cluster Form (PCF).

	DT_1	DT_2	DT_3	DT_4
cfs_1	**1**	**0**	**1**	**1**
cfs_2	**0**	**1**	**0**	**1**
cfs_3	**1**	**1**	**1**	**0**

Fig. 3. *CD* matrix

The computation of structural similarity between documents and clusters in the incremental clustering process is given below.

Structural Similarity Computation
Using *CD* matrix, we compute the structural similarity between

1. *two document trees*
2. *a tree and a cluster*

Tree-to-Tree Similarity
To begin with, there exists no cluster. Firstly, the two trees are used to compute the pair-wise similarity to form a cluster. It is measured by first finding the common closed frequent subtrees between the two document trees using the *CD* matrix.

Problem Definition for Tree-to-Tree Similarity
Let there be two document trees DT_x and DT_y and their pre-cluster forms (PCFs), d_x and d_y respectively in the given *CD* matrix. For a given *CD* matrix, let $CFS = \{cfs_1,\ldots, cfs_n\}$ be a set of closed frequent subtrees representing the rows and let $DT = \{DT_1, DT_2, DT_3,\ldots,DT_n\}$ be the document trees representing the columns, the PCF of a document tree DT_x *is* $d_x = \{x_1, x_2,\ldots, x_n\}$ where $x_1 \ldots x_n \in \{0,1\}$ and $n = |CFS|$.

To compute the tree-to–tree similarity using the PCFs d_x and d_y in the *CD* matrix, the common closed frequent subtrees ($\zeta_i(d_x, d_y)$), between the two document trees DT_x and DT_y are computed for a given *i-th* closed frequent subtree using,

$$\zeta_i(d_x, d_y) = (d_x(i) \,\&\, d_y(i)=1) \;?\; 1 : 0 \tag{2}$$

Using the PCFs d_x and d_y in the *CD* matrix, the possible *i-th* closed frequent subtrees ($\alpha_i(d_x, d_y)$) is calculated between the two document trees DT_x and DT_y using,

$$\alpha_i(d_x, d_y) = (d_x(i) \,|\, d_y(i)=1) \;?\; 1 : 0 \tag{3}$$

The degree of similarity ($\Omega_{dx,\,dy}$) between the two document trees using their PCFs, d_x and d_y is finally computed by combining the equations (2) and (3). The degree of similarity between the two document trees is the probability of the occurrence of a common closed frequent subtree in the possible closed frequent subtree space. It is defined as the ratio of sum of the common closed frequent subtrees over the total number of the possible closed frequent subtrees between a pair of document trees.

$$\Omega_{dx,\,dy} = \frac{\sum_{i=1}^{j} \zeta_i(d_x, d_y)}{\sum_{i=1}^{j} \alpha_i(d_x, d_y)} \quad \text{where } j = |CFS| \tag{4}$$

If the tree-to-tree similarity value ($\Omega_{dx,\,dy}$) between the PCFs, d_x and d_y of DT_x and DT_y respectively is higher than the user-defined minimum cluster threshold *(μ)*, then, d_x and d_y are grouped into the same cluster otherwise they are assigned to two separate clusters. If they are grouped into the same cluster then the two PCFs are merged by union operation.

$$d_{clust}(i) = (d_x(i) \,|\, d_y(i)=1) \;?\; 1 : 0 \tag{5}$$

Tree to Cluster Similarity

Once a cluster is formed, the similarity between the incoming document tree and the existing cluster is computed using their PCFs given by d_x and d_{clust} respectively. It is computed using the Equation (2) given by

$$\zeta_i(d_x, d_{clust}) = (d_x(i) \,\&\, d_{clust}(i) = 1) \;?\; 1 : 0 \tag{6}$$

Similar to Equation (3), the possible closed frequent subtrees between a document tree and a cluster is computed as follows,

$$\alpha_i(d_x, d_{clust}) = (d_x(i) \,|\, d_{clust}(i) = 1) \;?\; 1 : 0 \tag{7}$$

Using equations (6) and (7), the degree of similarity between a document tree and a cluster is computed. The degree of similarity between the document tree and a cluster is the probability of the occurrence of a common closed frequent subtree in the possible closed frequent subtree space. It is defined as the ratio of the sum of common closed frequent subtrees over the total number of possible closed frequent subtrees between a document tree and its cluster.

$$\Omega_{\text{dx, clust}} = \sum_{i=1}^{j} \frac{\zeta_i(d_x, d_{clust})}{\alpha_i(d_x, d_{clust})} \quad \text{where } j = |CFS| \tag{8}$$

If the tree-to-cluster similarity value ($\Omega_{\text{dx, clust}}$) between PCFs d_x and d_{clust} of DT_x and DT_{clust} is higher than the user-defined minimum cluster threshold (μ), then, d_x and d_{clust} are grouped into the DT_{clust} cluster otherwise d_x is assigned to a separate cluster. In situations where d_x is grouped into the DT_{clust} cluster then the two clusters are merged by union operation.

$$d_{clust}(i) = (d_x(i) | d_{clust}(i) = 1) \; ? \; 1 : 0 \tag{9}$$

CFSPC is a progressive clustering algorithm. The clusters are formed in an incremental fashion. The process starts without any cluster. When a new tree arrives, it is assigned to a new cluster. A cluster is represented as the PCF of the document tree if it has a single member. A cluster with multiple member document trees is represented by the union of their PCFs. When the next tree arrives, the similarity between the current document tree and the document tree in the cluster is computed using the tree to tree similarity method. If the similarity value is greater than the user-defined cluster threshold (μ) then the incoming document tree is grouped into the cluster otherwise it is assigned to a new cluster. If there exists new PCF information with respect to the closed frequent subtrees in the recently clustered document tree, then the additional information is merged with the clustering information.

The incremental clustering results in a large number of clusters. This is due to allowing the documents to form a separate cluster when an appropriate cluster is not found for them. In order to control the number of clusters, the clusters are further merged using pair-wise clustering.

4.2 Partitional Clustering

A similarity matrix is generated by computing the degree of similarity between each pair of PCFs representing the clusters using the following equations,

$$\alpha_i(d_{clust_1}, d_{clust_2}) = (d_{clust_1}(i) | d_{clust_2}(i) = 1) \; ? \; 1 : 0 \tag{10}$$

$$\Omega_{clust_1, clust_2} = \sum_{i=1}^{j} \frac{\zeta_i(d_{clust_1}, d_{clust_2})}{\alpha_i(d_{clust_1}, d_{clust_2})} \quad \text{where } j = |CFS| \tag{11}$$

where $\Omega_{clust_1, clust_2}$ is the cluster-to-cluster similarity value. The similarity matrix is fed to a partitional clustering algorithm such as the k-way clustering solution[5]. The k-way clustering algorithm groups the documents to the required number of clusters. The k-way clustering solution computes cluster by performing a sequence of *k-1* repeated bisections. In this approach, the matrix is first clustered into two groups, and then one of these groups is chosen and bisected further. This process of bisection continues until the desired number of bisections is reached. During each step of bisection,

the cluster is bisected so that the resulting 2-way clustering solution locally optimizes a particular criterion function [5].

5 Experiments and Discussion

We implemented the CFSPC algorithm using Microsoft Visual C++ 2005 and conducted experiments on the Wikipedia corpus from the INEX XML Mining Challenge 2007. The required numbers of clusters for INEX result submission were 21 and 10 clusters. The incremental clustering technique for a given clustering threshold often generates a large number of clusters. Hence, the k-way clustering algorithm option in CLUTO[5] is used to group the intermediate clusters to the required number of clusters (21 and 10 clusters).

We submitted 2 results, one with 21 clusters and the other with 10 clusters using the cluster threshold of 0.4. The following table summarizes the results based on Micro F1 and Macro F1 measure evaluation metrics for 10 and 21 clusters with the clustering threshold of 0.4.

Table 1. Submitted clustering results for INEX Wikipedia XML Mining Track 2007

Clustering Threshold	Number of Clusters using incremental clustering	Number of Clusters	Micro F1	Macro F1
0.4	2396	21	0.251	0.251
		10	0.251	0.250

We conducted several more experiments with varying support threshold and clustering threshold. The experimental results for varying clustering threshold are shown in Table 2.

Table 2. Results from INEX Wikipedia XML Mining Track 2007 with varying clustering threshold

Clustering Threshold	Number of Clusters using incremental clustering	Number of Clusters	Micro F1	Macro F1
0.5	3735	21	0.252	0.248
		10	0.251	0.249
0.3	1682	21	0.253	0.249
		10	0.251	0.247
0.2	1217	21	0.252	0.261
		10	0.251	0.249
0.1	857	21	0.251	0.258
		10	0.251	0.263

As indicated in the Tables 1 and 2, the number of clusters using incremental clustering increases with the clustering threshold. The partitional clustering could provide the required number of clusters. It can be seen from the Table 2 that there is not much improvement in the Micro F1 average; however, there is an improvement for Macro F1 average for lower clustering threshold. The results on the Wikipedia dataset clearly indicates that there is not any significant improvement in performance for varying clustering threshold using structural only information in clustering.

To analyse whether the number of closed frequent subtrees is an influential factor in final clustering results, experiments are conducted with the higher support threshold than the previous set of experiments. We ran the experiments with varying clustering thresholds setting the 10% support threshold to generate the frequent trees.

Table 3. Results from INEX Wikipedia XML Mining Track 2007 for 10% Support threshold and various clustering threshold

Support Threshold	No. of Closed Frequent Subtrees	Clustering Threshold	No. of Clusters using Inc. Clustering	No. of Clusters from Part. clustering	Micro average (F1)	Macro average (F1)
10%	387	0.4	1118	21	0.253	0.269
				10	0.252	0.245
		0.5	1633	21	0.253	0.256
				10	0.251	0.247
		0.6	2510	21	0.252	0.248
				10	0.251	0.243

Also, we wanted to analyse whether the number of clusters plays a significant role. The above Table 3 summarizes the results on various numbers of clusters at 0.5 clustering threshold with 10% support threshold. The results from Table 3 show that the clustering performance does not vary much with the change of various parameters.

Table 4. Comparison of our approach against other structure-only approaches on INEX Wikipedia dataset

Approaches	Number of clusters	Micro F1	Macro F1
Hagenbuchner et.al[4]	10	0.251	0.257
	21	0.264	0.269
Hagenbuchner[4]	10	0.252	0.267
	21	0.258	0.252
Tien et. Al[10]	10	0.251	0.252
	21	0.251	0.253
Our approach	10	0.251	0.263
	21	0.253	0.269

Table 4 lists the comparison between our approach and other approaches using structure-only on INEX 2007 Wikipedia dataset. There were two other participants using structure-only and their results are presented in Table 4. It is evident from Table 4 that there is no significant difference between our approach and other approaches using only the structure of XML documents. Based on our experiments and the comparison with other approaches[4, 11] using structure-only in the INEX 2007 Document Mining challenge, it can be concluded that clustering using structural similarity between documents is not suitable for the INEX 2007 Wikipedia data set. As the INEX 2007 Wikipedia dataset is a homogeneous collection with most of the documents having only one schema and hence the structure of the XML document plays a less important role than the content.

6 Conclusions and Future Direction

In this paper, we have proposed and presented the results of our progressive clustering algorithm for mining only the structure of XML documents in INEX 2007 Wikipedia dataset. The main aim of this study is to explore and understand the importance of structure of the XML documents over the content of XML for clustering task. In order to cluster the XML documents, we have used a frequent subtree – document matrix generated from closed frequent subtrees. Using the matrix, we have computed the similarity between XML documents and incrementally clustered them based on their similarity values. From the experimental results, it is evident that the structure plays a minor role in determining the similarity between the INEX documents.

This is the first study conducted on INEX dataset using common subtrees and hence in the future, we will aim in devising efficient similarity computation techniques to effectively cluster the XML documents. Also, as a future work, we will be focusing on including the content of XML documents to provide more meaningful cluster.

References

1. Aggarwal, C.C., et al.: 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, San Jose (2007)
2. Chi, Y., et al.: Frequent Subtree Mining- An Overview. In: Fundamenta Informaticae, pp. 161–198. IOS Press, Amsterdam (2005)
3. Dalamagas, T., et al.: A methodology for clustering XML documents by structure. Inf. Syst. 31(3), 187–228 (2006)
4. Hagenbuchner, M., et al.: Efficient clustering of structured documents using Graph Self-Organizing Maps. In: Pre-proceedings of the Sixth Workshop of Initiative for the Evaluation of XML Retrieval, Dagstuhl, Germany (2007)
5. Karypis, G.: CLUTO - Software for Clustering High-Dimensional Datasets Karypis Lab, May 25 (2007)
6. Kutty, S., Nayak, R., Li, Y.: PCITMiner- Prefix-based Closed Induced Tree Miner for finding closed induced frequent subtrees. In: Sixth Australasian Data Mining Conference (AusDM 2007), ACS, Gold Coast (2007)

7. Kutty, S., Nayak, R., Li, Y.: XML Data Mining: Process and Applications. In: Song, M., Wu, Y.-F. (eds.) Handbook of Research on Text and Web Mining Technologies. Idea Group Inc., USA (2008)
8. Nayak, R., Witt, R., Tonev, A.: Data Mining and XML Documents. In: International Conference on Internet Computing (2002)
9. Nayak, R.: Investigating Semantic Measures in XML Clustering. In: Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence, pp. 1042–1045. IEEE Computer Society Press, Los Alamitos (2006)
10. Tran, T., Nayak, R.: Evaluating the Performance of XML Document Clustering by Structure Only. In: Comparative Evaluation of XML Information Retrieval Systems, pp. 473–484 (2007)
11. Tran, T., Nayak, R.: Document Clustering using Incremental and Pairwise Approaches. In: Pre-proceedings of the Sixth Workshop of Initiative for the Evaluation of XML Retrieval, Dagstuhl, Germany (2007)
12. Xing, G., Xia, Z., Guo, J.: Clustering XML Documents Based on Structural Similarity. In: Advances in Databases: Concepts, Systems and Applications, pp. 905–911 (2007)