XML Documents Clustering using Tensor Space Model-A Preliminary Study

Sangeetha Kutty,Richi Nayak, Yuefeng Li
School of Computer Science
Queensland University of Technology (QUT)
Brisbane, Qld
Email: {s.kutty,r.nayak,y2.li}@qut.edu.au

***Abstract*—A hierarchical structure is used to represent the content of the semi-structured documents such as XML and XHTML. The traditional Vector Space Model (VSM) is not sufficient to represent both the structure and the content of such web documents. Hence in this paper, we introduce a novel method of representing the XML documents in Tensor Space Model (TSM) and then utilize it for clustering. Empirical analysis shows that the proposed method is scalable for a real-life dataset as well as the factorized matrices produced from the proposed method helps to improve the quality of clusters due to the enriched document representation with both the structure and the content information.**

***Keywords*-XML documents; Clustering; Tensor; Structure and Content; Decomposition;**

I. Introduction

Rapid growth of web technologies has witnessed a sudden surge in the number of XML (eXtensible Markup Language) documents. For instance, English Wikipedia contains 3.1 million web documents in XML format; the ClueWeb dataset, used in Text Retrieval Conference (TREC) tracks, contains 503.9 million XML documents collected from the web in January and February 2009. The majority of existing XML document clustering methods utilize either the structure features [1] or the content features present in the documents. Clustering methods utilizing only the content features of the documents consider the documents as a "bag of words" or a Vector Space Model (VSM) and ignore the structure features [1]; clustering methods utilizing only the structure features of the documents represent each document as a set of paths (sequences) or trees.

However, these methods, with their single-feature focus, tend to falsely group documents that are similar in both features. To correctly identify similarity among documents, the clustering process should use both their structure and their content information.Approaches on clustering both the structure and the content features of the XML documents are limited. Approaches using the VSM often fail to scale for even small collections of a few hundred documents, and in some situations have resulted in poor accuracy [2]. VSM cannot model both structure and content features of XML documents effectively as the mapping between the structure and its corresponding content is lost. The content and structure features inherent in an XML document should be modeled in a way that the mapping between the content of the path or tree can be preserved and used in further analysis. The example shown in Fig. 1 the importance of using both the structural and content features for clustering. Fig. 1 shows the fragments of six XML documents from the publishing domain: the XML fragments shown in (a), (b), (c) and (f) share a similar structure and the fragments in (d) and (e) share a similar structure. It can be noted in Fig. 1 that though the fragments in (a) and (f) have a similar structure to (b) and (c) fragments, these two sets of fragments differ in their content. Utilizing a clustering method based only on the structural similarity of XML documents will result in two clusters about "Books" and "Conference Articles". However, it will fail to further distinguish the documents in the "Books" cluster. On the other hand, utilizing a clustering method based only on content similarity will fail to distinguish between "Books" and "Conference Articles". In order to derive meaningful clusters, these fragments should be analyzed in terms of both their structural and their content similarity. Clustering the XML documents by considering the structural and content features together will result in three clusters, namely "Books on Data Mining (DM)", "Books on Biology (Bio)" and "Conference articles on Data Mining". We could use these kinds of meaningful clusters for effective storage and retrieval of XML documents. Despite the advantage of using both structural and content features in clustering, there are several challenges in combining them effectively and utilizing the combination for clustering. This paper proposes a novel way to represent the common substructures of XML documents using frequent subtrees to represent the structural similarity among them. It also aims to utilize these frequent subtrees when representing content features. The running example in Fig. 1 shows that there are two common or frequent substructures: <Title>,<Author> among the fragments (a), (b), (c)and (f) and <ConfTitle> <ConfAuthor> <ConfLoc> among the fragments (d) and (e). This leads to the hypothesis that content corresponding to these tags only should be deemed important. Content corresponding to infrequent substructures such as <publisher>, <year>, <ConfName> and <ConfYear> can be ignored as these

<Book>
<Title> On the Origin of Species</Title>
<Author> <Name>Charles Darwin</Name></Author>
<Publisher><Name>John Murray </Name></Publisher>
</Book>

(a)

<Book>
<Title> Data Mining concepts and Techniques</Title>
<Author> Jiawei Han</Author>
<Author> Micheline Kamber</Author>
<Publisher>Morgan Kaufmann</Publisher>
<Year> 2006</Year>
</Book>

(b)

<Book>
<Title> Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)</Title>
<Author>Eibe Frank</Author>
<Author>Ian Witten</Author>
</Book>

(c)

<Conference>
<ConfTitle>Survey of Clustering Techniques </ConfTitle>
<ConfAuthor>John Smith</ConfAuthor>
<ConfName> SIAM International Conference</ConfName>
<ConfYear> 2007 </ConfYear>
<ConfLoc>Las Vegas </ConfLoc>
</Conference>

(d)

<Conference>
<ConfTitle>An exploratory study on Association rules mining </ConfTitle>
<ConfAuthor>Michael Bonchi</ConfAuthor>
<ConfName> AusDM Conference</ConfName>
<ConfLoc> Melbourne, Australia</ConfLoc>
</Conference>

(e)

<Book>
<Title> Classification of Plants</Title>
<Author> <Name>John Brown </Name></Author>
<Publisher><Name>Springer Publications</Name></Publisher>
</Book>

(f)

Figure 1. A Sample XML Dataset

do not occur in most of the documents. By eliminating the content corresponding to infrequent substructures the dimensionality of the input (content) data is reduced. Our proposed method applies clustering on this reduced input data and produces the three desired clusters - differentiating between conference articles and books on DM and Bio. Our method is able to group the documents based on the content and structural similarity by containing only the relevant content constrained by the common substructure of the documents.

In this paper we propose a novel method that represents the XML documents in a Tensor Space Model (TSM) and uses the TSM for clustering. In the TSM, storing the content corresponding to its structure helps to analyze the relationship between structure and content. Unlike the VSM, which uses a vector to model, TSM is based on the multi-linear algebraic character level high-order tensors (generalization of matrices) [3]. Decomposition algorithms are used to analyze the relationships between various tensor dimensions. Using these decomposition algorithms we cluster the XML documents. In this paper we propose a clustering method, XML document Clustering with TSM (XCT), that utilizes a tensor model to efficiently combine the content and structure features of XML documents.

II. RELATED WORK

XML document mining using both the structure and the content has received significant attention in the last decade. XML data mining track in Initiative for Evaluation of XML retrieval (INEX) is a major contest with a special focus on classification and clustering of XML documents conducted since 2005 [4], [5]. However, most of the XML document clustering methods, with some exceptions [6], [7], utilize either the structure or the content features of XML documents in order to cluster them [1], [8]. Clustering methods using both the features fail to scale for even small collections [2] and in some situations they have resulted in poor accuracy [2], [6]. This could be due to ineffective combination of structure and content such as capturing entire structure and content in the XML documents including irrelevant and redundant information. A recent structure-based clustering method, XProj [1] clusters the XML documents by utilizing frequent substructures. XProj is disadvantaged when the clustering depends not only on the structure features but

on the content features or both. There has been limited research on clustering using multi-dimensional aspects of the documents due to the complexity and the explosion of data produced as a result of combining these multi-dimensional features. The complexity becomes worse when dealing with large-sized datasets. There has been an attempt [7] to use a BitCube representation to cluster and query the XML documents. Paths, words and documents represent the three dimensions of the BitCube where each entry in the cube presents either the presence or absence of a given word in the path of a document. However, this approach suffers from the typical disadvantages inherent in Boolean representation models, such as the lack of partial matching criteria and natural measures of document ranking. Also, it is evident from the experimental results that it is an expensive task to project a document into small bitcubes based on their paths. This calls for a multi-dimensional representation of XML documents using not all the content features but more significant content features. Defining what is significant is a critical problem. A previous research [9] has shown that using the content constrained within the common frequent subtrees is sufficient for clustering. Motivated by this, in this research we consider the content features as significant if they appear within the common subtrees in the documents.

Tensor Space Modeling (TSM) has been successfully used in representing and analyzing multi-dimensional data in signal processing, web mining and many other fields [3]. Tensor clustering is a multi-way data analysis task which is currently gaining importance in the data mining community. The simplest tensor clustering scenario, co-clustering or bi-clustering, in which two dimensions are simultaneously clustered, is well established [10]. Authors in [11] proposed a method for multi-way clustering on tensors by extending the co-clustering technique from matrices to tensors using relational graphs. Another recently proposed approximation based Combination Tensor Clustering algorithm [12] clusters along each of the dimensions and then represents the cluster centers in the tensor. These co-clustering techniques capture only the 2-way relationships among the features and ignore the dependence of multiple dimensions in clustering: this may result in loss of information while grouping the objects. Selee et al. [13] studied the problem of applying tensor clustering techniques in grouping bibliographic data with multiple similarity values. They utilised six different similarity values such as similarity among words in abstracts, between names of authors co-citations etc. These similarities are pre-defined which is in contrast to our work, where we apply tensor clustering on XML documents without any prior assumption as it should be in the case of clustering a large number of documents with diverse nature.

Several decomposition algorithms, such as Higher Order SVD (HOSVD); CP, a higher-order analogue of Singular Value Decomposition (SVD) or Principal Component Analysis (PCA);Tucker and Multi-Slice Projection, have been reviewed in detail in [14]. PARAFAC is a higher-order analogue of Singular Value Decomposition (SVD) or Principal Component Analysis (PCA). The PARAFAC solutions are not unique due to its heavy dependence on initial guess but HOSVD and Tucker tend to provide unique solutions. Incremental Tensor Analysis (ITA) methods [15] have been proposed recently to detail with large datasets for efficiently decomposing sparse tensors ($density \leq 0.001\%$). MET [16],a memory-efficient implementation of Tucker proposed to avoid the intermediate blow-up in tensor factorization, is shown in our results shows not to scale to our medium-sized and large-sized datasets. In MACH [3], a recently proposed random decomposition technique suitable for large dense datasets, the number of entries in the tensor is randomly reduced using Achlioptas-McSherry's technique [17] to decrease the density of the dataset. However, as discussed in section 5, MACH often ignores smaller length documents and tends to group most of the smaller length documents in a single cluster in spite of differences in their structure and content. In this paper we unfold a tensor into a matrix and apply SVD on the generated matrix.

III. The Proposed XCT Method

A. Problem Definition and preliminaries

Let there be a collection of XML documents $D = \{D_1, D_2, ..., D_n\}$, where D_i is an XML document containing tags and data enclosed within those tags. The structure of D_i can be defined as a list of tags showing the hierarchical relationships between them. The structure of D_i is modeled as a rooted, ordered and node-labeled document tree, $DT_i = (N, n0, E, f)$, where (1) N is the set of nodes that correspond to tags in D_i, with the node labels corresponding to tag names; (2) $n0$ is the root node which does not have any edges entering in it; (3) E is the set of edges in DT_i; and (4) f is a mapping function $f : E \rightarrow N \times N$. Previous research has shown that, in a dataset, only the content constrained within the concise common or frequent subtrees (Closed Frequent Induced - CFI) can be used to group the documents, rather than the entire content of the XML documents [9]. Therefore the proposed XCT method generates these CFI subtrees to represent the common subtrees in the dataset and uses these CFI subtrees to extract the content of the documents corresponding to them. The process begins by identifying the subtrees that belongs to a document tree. A subtree $CFI_j \in CFI$ is present in document tree DT_i, if CFI_j preserves the same parent-child relationship as that of DT_i. The document content(or *structure-constrained content*) contained within the CFI_j subtree in DT_i, noted as $C(D_i, CFI_j)$, is retrieved from the XML document D_i, a collection of node values or terms. The node value of a node (or tag) of a CFI_j, $C(N_i)$ in D_i is a vector of terms, $\{t_1, ..., t_k\}$ that the node contains. The term t is obtained after stop-word removal and stemming.

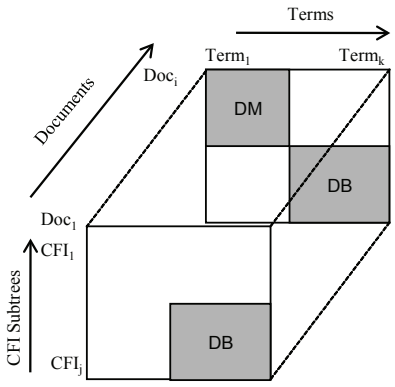

Figure 2. Visualisation of a third-order tensor for the XML document dataset

An induced subtree dt_i preserves the parent-child relationship among the nodes of the document tree, DT_i. dt_i with node set n' and edge set e' is an induced subtree of DT_i with node set N and edge set E *iff* (1) $n' \in N$; (2) $e' \in E$; (3) the labeling of nodes of n' in dt_i is preserved in DT_i; (4) $(n1, n2) \in e'$ where $n1$ is the parent node of $n2$ in dt_i *iff* $n1$ is the parent node of $n2$ in DT_i; and (5) for $n1, n2 \in n', preorder(n1) < preorder(n2)$ in dt_i *iff* $preorder(n1) < preorder(n2)$ in DT_i [9]. The frequent induced (FI) subtrees in DT are the subtrees that have a support that is equal to or more than the user-defined minimum threshold (min_supp). Given two frequent induced subtrees dt_i and dt_j in DT, the frequent induced subtree dt_i is closed of dt_j *iff* (1) dt_j is an induced subtree of $dt_i, supp(dt_i) = supp(dt_j)$; (2) there exists no supertree for dt_i having the same support as that of dt_i [9]. These subtrees are called CFI subtrees.

The next step involves modeling the derived structure and content features in a tensor model. Firstly,the tensor notations and conventions used in this paper are akin to the notations used by previous works [14], [13], [3]. Let $\mathcal{T} \in R^{M_1 \times M_2 \times M_3 \times ... M_n}$ be a tensor of n dimensions where M_i is a dimension (or mode or way). The order of a tensor is the number of dimensions. In this work, we focus on the third-order tensor, $\mathcal{T} \in R^{M_1 \times M_2 \times M_3}$. Entries of a tensor are shown using a_{ijk} and the subscript (i, j, k) range from I, J, K in each dimension. Each element (or entry) of a tensor needs n indices to represent or reference its precise position in a tensor. For example, the element a_{ijk} is an entry value at the i, j and k dimensions. Given the documents set D, its corresponding set of CFI subtrees and the set of terms for each CFI subtree, the collection of XML documents is now represented as a third-order tensor $\mathcal{T} \in R^{D \times CFI \times Terms}$. The tensor is populated with the number of occurrences of the structure-constrained term $Terms_i$ that corresponds to the CFI_j for document D_k. An optimization technique is applied on the CFI dimension to reduce the size of the tensor. Figure 2 visualizes a 3-order tensor for the document dataset.

Input: *Document Dataset;* ***D****, Dcoument Tree Dataset;* ***DT****, Mininum Support:* ***min_supp****, Constraint Length:* ***const****, NumCluster:* ***c***;

Output: *Clusters:* $\{Clust_1 ... Clust_c\}$

1: Compute $CFI = \{CFI_1, ..., CFI_p\}$ for DT using the PCITMinerConst algorithm for the given *min_supp* and *len*.
2: Form clusters of similar CFI subtrees, $CFISC = \{(CFI_1, ..., CFI_q)(CFI_r, ..., CFI_s)(CFI_t, ..., CFI_u)\}$, where $CFISC = \{CFISC_1, ..., CFISC_h\}, k \ll p$ using large itemset algorithm.
3: **for** every document $D_i \in D$ **do**
4: Identify the $CFISC$ existing in $DT_i, \delta(DT_i) = \{CFISC_l, ..., CFISC_h\}$.
5: For every $CFISC_j$ in $\delta(DT_i)$ retrieve the structure-constrained content in D_i, $C(D_i, CFISC_j) = C(N_1), ..., C(N_m)$. The set $C(N_m) = t_1, ..., t_k \in Terms$, where *Terms* is the term list in *D*.
6: **end for**
7: Form a tensor $\mathcal{T} \in R^{DXCFISCXTerms}$,where each tensor element is the number of times a term t_k occurs in $CFISC_j$ for a given document D_i.
8: $\mathbf{T}_{(D)}$ = Unfold $\mathcal{T}$ along its $Document(D)$ mode
9: Compute SVD on $\mathbf{T}_{(D)}$
10: Set U'_D to be the r_d leading left singular matrix of U_D
11: Apply K-means clustering to U'_D to generate the c number of clusters.

Figure 3. High level definition of XCT approach

Figure 3 provides an overview of the XCT method. It begins with mining the CFI subtrees using the PCITMinerConst algorithm and then identifying the constrained content within those CFI subtrees for a given document. Once the structure and content features are obtained for each document, the documents are represented in the TSM along with their structure and content features. The next task is to decompose the created TSM to obtain factorized matrices. Lastly, the K-means algorithm is applied to one of the factorized matrices representing the left singular matrix for the "Document" dimension U_D and the clusters of documents are obtained.

B. Generation of Structure Features for TSM

The Prefix-based Closed Induced Tree Miner (PCITMiner) algorithm [18] is modified to generate the length-constrained CFI subtrees from the document tree dataset DT. The length constrained CFI subtrees are used in this method for the following reasons: (1) Extracting all the CFI subtrees is computationally expensive for datasets with a high branching factor; (2) All CFI subtrees are not required while utilizing them in retrieving the content. In fact the long sized CFI subtrees become more specific and result in retrieving distinct terms associated only with this tree. This may result in a higher number of clusters with uneven sizes. We call the modified algorithm the PCITMinerConst

algorithm. Figure 3 illustrates the computationally expensive operation of checking whether the mined CFI exists in a given document tree due to the graph isomorphism problem. This step can be optimized by grouping similar subtrees based on their similarity and then retrieving the content corresponding only to the group of similar CFI. A large itemset algorithm for clustering transactional data has been modified to include subtrees, rather than items, to conduct the grouping of the CFI trees based on the similarity of the subtrees. The clusters of CFI subtrees, called Closed Frequent Induced Subtree Cluster ($CFISC$), become a tensor dimension for representing and analyzing XML documents. Let $CFISC$ be a set of CFI subtrees given by $\{(CFI_1, ..., CFI_q)(CFI_r, ..., CFI_s)(CFI_t, ..., CFI_u)\}$

C. Generation of Content Features for TSM

$CFISC$ is used to retrieve the structure-constrained content from the XML documents. We now define the coverage of a $CFISC_j$ and its constrained content for the given document D_i. Compared with the content features of an XML document, the structure-constrained content features include the node values corresponding only to the node labels of the set of CFI subtrees in $CFISC_j$.

Definition 1: Structure-Constrained content features. These features of a given $CFISC_j, C(D_i, CFISC_j)$ of an XML document D_i, are a collection of node values corresponding to the node labels in the $CFISC_j$ where $CFISC_j$ is a cluster of CFI subtrees corresponding to DT_i. The node value of a node (or tag) of a $CFISC_j \in CFISC, C(N_i)$, in D_i is a vector of terms, $\{t_1, ..., t_k\}$ that the node contains. The term t is obtained after using pre-processing techniques such as stop-word removal and stemming. Firstly, the CFI subtrees corresponding to the $CFISC_j = \{CFI_r, ..., CFI_s\}$ for a given document D_i are flattened into their nodes $\{N_1, ..., N_m\} \in N$, where *N* is the list of nodes in *DT*. Then the node values of $\{N_1, ..., N_m\}$ are accumulated and their occurrences for a document D_i are recorded.

D. The TSM Representation and Decomposition

For the given documents set D, its corresponding set of CFI subtrees and the set of terms for each CFI subtree, the collection of XML documents is now represented as a third-order tensor $\mathcal{T} \in R^{D \times CFI \times Terms}$. The tensor $\mathcal{T}$ is populated with the number of occurrences of the structure-constrained term $Terms_i$ that corresponds to the CFI_j for document D_k. Given the tensor $\mathcal{T}$, the next task is to find the hidden relationships between the dimensions. The tensor decomposition algorithms enable an overview of the relationships that can be further used in clustering. The tensors are built and unfolded or matricized incrementally. Figure 4 shows the process of matricization or unfolding along the mode-1 of $\mathcal{T}$which results in a matrix $\mathrm{T}_{(1)}$. This means that the mode-1 fibers (higher order analogue of rows and columns) are aligned to form a matrix. Essentially this means that the mode-1 fibers of $\mathcal{T}$ are mapped to the rows of matrix $\mathrm{T}_{(1)}$ and the modes-2 and -3 are mapped to the columns of this matrix. Then, we matricize the tensor and decompose the matrix using Singular Value Decomposition (SVD).

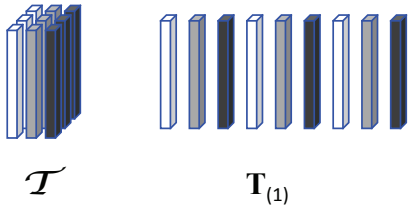

Figure 4. Mode-1 matricization of a 3- order tensor

E. The TSM Clustering

Huang et al. [10] have theoretically proved that higher order SVD (HOSVD) simultaneously reduces the subspace and groups the values in each dimension. Since the PTCD algorithm is a progressive analogue of HOSVD we utilize the same theorem. The 3-order tensor given $\mathcal{T} = \{\mathcal{T}_{ijk}\}^{D_{i=1} CFISC_{j=1} Terms_{k=1}}$. The rank-1 and PTCD, treats every index uniformly as given by

$$\min_{U_1, U_2, U_3, \mathcal{Y}} J_1 = ||\mathcal{T} - U_1 \times_1 U_2 \times_2 U_3 \times_3 \mathcal{Y}||^2 \quad (1)$$

where U_1, U_2, U_3 are leading left singular matrix and $\mathcal{Y}$ is a 3-order core tensor. U_2 and U_3 include the subspaces after projection and matrix U_1 provides the clustering results on the data index direction and hence they are the cluster indicators for the documents. Consequently, we apply the K-means clustering algorithm on the U_1 matrix [15].

IV. EXPERIMENTS AND DISCUSSIONS

Experiments are conducted to evaluate the accuracy and scalability performance of XCT on a real-life dataset.

A. Dataset Description

The ACM SIGMOD dataset contains 140 XML documents corresponding to two DTDs, IndexTermsPage.dtd and OrdinaryIssuePage.dtd (with about 70 XML documents for each DTD), similar to the setup in XProj [1]. Previous research for XML documents clustering [1] has used the ACM to cluster the documents into two groups according to their structural similarity. To compare our work with this earlier research, we conducted our experiments not only with two cluster categories according to structural similarity but also on 5 categories using expert knowledge considering both the structure and the content features of XML documents. We have made this dataset available[1].

[1]http://sky.fit.qut.edu.au/ nayak/datasets

Table I
DETAILS OF THE ACM SIGMOD DATASET

Attributes	ACM SIGMOD dataset
No. of Docs	140
No. of tags	38
No. of internal nodes	2070
Max length of a document	45
No. of distinct terms	7135
Total No. of words	38141
Size of the collection	1 MB
Presence of formatting tags	No
Presence of Schema	Yes
Number of Categories	2 & 5

B. Experimental design

Experiments were conducted on the High Performance Computing system, with a RedHat Linux operating system, 16GB of RAM and a 3.4GHz 64bit Intel Xeon processor core. Experiments were conducted to evaluate the accuracy of clustering results of XCT over other clustering techniques, decomposition techniques and representation.

Following are the representation and the other existing algorithms used for comparing the outputs of the proposed XCT method.

Structure Only (SO) Representation: An input matrix $D\times$ CFI is generated where D represents the documents and CFI represents the list of closed induced frequent document subtrees has been used. Each document is represented by the CFI subtrees that are present in it.

Content Only (CO) Representation: The content of XML documents is represented in a matrix $D\times Terms$ with Terms obtained after pre-processing using techniques such as stop-word removal, stemming and integer removal. Each entry in the matrix contains the term frequency of terms in D.

Clustering using CP and Tucker: The left singular matrix resulting from applying CP or Tucker decomposition on the tensor is used as an input for k-means clustering.

Clustering using MACH: The MACH decomposition technique has been applied on the original tensor without random indexing. MACH randomly projects the original tensor to a reduced tensor with smaller percentage of entries (10% from the original tensor as specified in [3]) and then uses Tucker decomposition to decompose the reduced tensor. To compare with XCT, we apply k-means clustering on the left singular matrix to group the documents. Moreover, since INEX data have been used by other researchers; we provide the results cited by other researchers as well in our empirical analysis.

C. Evaluation measures

The standard criterion of purity is used to determine the quality of clusters by measuring the extent to which each cluster contains documents primarily from one class. The macro and micro purity of entire clustering solution is obtained as a weighted sum of the individual cluster purity. In general, larger the value of purity, better the clustering solution is.

Table II
RESULTS OF PURITY ON ACM SIGMOD DATASET

# Clusters	Methods	Micro-purity	Macro-purity
5	XCT	0.91	0.91
	S+C using VSM	0.75	0.79
	Clustering using MACH	0.70	0.75
	Clustering using Tucker	0.59	0.87
	Clustering using CP	0.84	0.93
	CO	0.73	0.78
	SO	0.64	0.72
2	XCT	1	1
	S+C using VSM	0.98	0.98
	Clustering using MACH	0.97	0.93
	Clustering using Tucker	0.56	0.48
	Clustering using CP	0.89	0.93
	CO	0.97	0.94
	SO	1	1

$$Purity = \frac{\#\ Docs\ with\ the\ majority\ label\ in\ cluster\ k}{\#\ Docs\ in\ cluster\ k} \quad (2)$$

$$Micro - Purity = \frac{\sum_{k=0}^{n} Purity(k) * \#\ Docs\ Found\ By\ Class(k)}{\sum_{k=0}^{n} \#\ Docs\ Found\ By\ Class(k)} \quad (3)$$

$$Macro - Purity = \frac{\sum_{k=0}^{n} Purity(k))}{Total\ Number\ of\ Categories} \quad (4)$$

D. Empirical Analysis

Accuracy of Clustering: Table II, provide the purity results of clustering on the datasets using XCT, other representations and other decomposition algorithms.

As it can be seen from this table that the proposed XCT method outperforms the other methods in terms of accuracy of their clustering solution. It should be noted that algorithms such as CP, Tucker,MACH shows a poor performance over XCT. We conduct the time complexity of XCT in the following subsection to understand its potential in the following subsection.

Time complexity analysis: Now we analyze the time complexity of XCT .The time complexity of XCT is composed of three major components namely frequent mining for CFI subtrees, clustering of CFI subtrees and decomposition. It is given by $O(d*s*m)+O(dcp)+\sum r\prod n+O(dkn)$ where d represents the number of documents, s is the number of 1-Length CFI subtrees,m is the number of PCITMinerConst iterations, c is the number of structure-based clusters, p is the number of similarity computation iterations, n is the number of terms, k is the number of non-zero values in the sparse matrix and r is the number of CFISC. The time complexity for decomposition is $\sum r\prod N_i+O((dkn)$ which includes the cost of matricization along mode m-1 and sparse Singular Value decomposition respectively.

Scalability Test: The scalability analysis was conducted with min_supp at 10%, ***constraint length*** at 5 and the number of clusters at 5. The execution time for PCITMinerConst is less than a few 10 milliseconds and hence it has not been

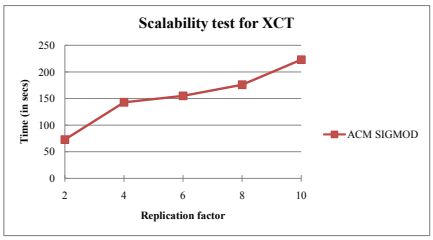

Figure 5. Scalability of XCT

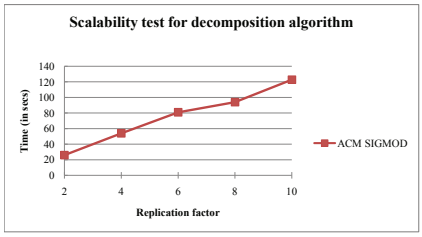

Figure 6. Scalability of the decomposition algorithm

reported as it is not of much significance. We can see from Figures 5 and 6 that XCT and the decomposition algorithm scale nearly linearly with the dataset size.

V. Conclusion

In this paper, we have proposed a clustering method, XCT, for effectively combining both the structure and the content features in XML documents. By utilizing the frequent subtrees in clustering the content, we developed a clustering method that can efficiently work for real-life XML dataset. The experimental results clearly ascertain that XCT outperforms other existing approaches in improving accuracy. Our future work will focus on applying the proposed clustering method on very large datasets and various types of other types of semi-structured documents.

References

[1] C. C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. Zaki, "Xproj: a framework for projected structural clustering of xml documents," in *KDD '07*. USA: ACM, 2007, pp. 46–55.

[2] A.-M. Vercoustre, M. Fegas, S. Gul, and Y. Lechevallier, "A flexible structured-based representation for xml document mining," in *Advances in XML Information Retrieval and Evaluation*, 2006, pp. 443–457.

[3] C. E. Tsourakakis, "MACH: Fast Randomized Tensor Decompositions," in *SDM 2010*, Columbus,Ohio, USA, 2010, pp. 689–700.

[4] L. Denoyer and P. Gallinari, "Report on the xml mining track at inex 2005 and inex 2006: categorization and clustering of xml documents," *SIGIR Forum*, vol. 41, no. 1, pp. 79–90, 2007.

[5] R. Nayak, C. deVries, S. Kutty, and S. Geva, "Report on the XML Mining Track Clustering Task at INEX 2009," in *Focused Retrieval and Evaluation*, S. Geva, J. Kamps, and A. Trotman, Eds. Springer, 2010.

[6] A. Tagarelli and S. Greco, "Semantic clustering of xml documents," *ACM Trans. Inf. Syst.*, vol. 28, no. 1, pp. 1–56.

[7] J. P. Yoon, V. Raghavan, V. Chakilam, and L. Kerschberg, "Bitcube: A three-dimensional bitmap indexing for xml documents," *Journal of Intelligent Information Systems*, vol. 17, no. 2, pp. 241–254, 2001.

[8] H.-p. Leung, F.-l. Chung, S. Chan, and R. A. Luk, "Xml document clustering using common xpath," in *WIRI '05.*, F.-l. Chung, Ed., 2005, pp. 91–96.

[9] S. Kutty, R. Nayak, and Y. Li, "HCX: an efficient hybrid clustering approach for XML documents," in *DocEng '09*. NY, USA: ACM, 2009, pp. 94–97.

[10] H. Huang, C. Ding, D. Luo, and T. Li, "Simultaneous tensor subspace selection and clustering: the equivalence of high order svd and k-means clustering," in *KDD '08*. New York, NY, USA: ACM, 2008, pp. 327–335.

[11] A. Banerjee, I. Dhillon, J. Ghosh, S. Merugu, and D. S. Modha, "A generalized maximum entropy approach to bregman co-clustering and matrix approximation," in *KDD '04*. NY, USA: ACM, 2004, pp. 509–514.

[12] S. Jegelka, S. Sra, and A. Banerjee, "Approximation algorithms for tensor clustering," in *Algorithmic Learning Theory*, 2009, pp. 368–383.

[13] T. M. Selee, T. G. Kolda, W. P. Kegelmeyer, and J. D. Griffin, "Extracting clusters from large datasets with multiple similarity measures using IMSCAND," in *CSRI Summer Proceedings 2007*, M. L. Parks and S. S. Collis, Eds., December 2007, pp. 87–103. [Online]. Available: http://www.cs.sandia.gov/CSRI/Proceedings/

[14] A. Evrim, B, and Y. lent, "Unsupervised multiway data analysis: A literature survey," *IEEE Trans. on Knowl. and Data Eng.*, vol. 21, no. 1, pp. 6–20, 2009.

[15] S. Jimeng, T. Dacheng, P. Spiros, S. Y. Philip, and F. Christos, "Incremental tensor analysis: Theory and applications," *ACM Trans. Knowl. Discov. Data*, vol. 2, no. 3, pp. 1–37, 2008.

[16] T. G. Kolda and J. Sun, "Scalable tensor decompositions for multi-aspect data mining," in *ICDM 2008*, December 2008, pp. 363–372.

[17] D. Achlioptas and F. Mcsherry, "Fast computation of low-rank matrix approximations," *J. ACM*, vol. 54, no. 2, p. 9, 2007.

[18] S. Kutty, R. Nayak, and Y. Li, "PCITMiner: prefix-based closed induced tree miner for finding closed induced frequent subtrees," in *AusDM '07*. Australia: Australian Computer Society, Inc., 2007, pp. 151–160.