International Journal of Computer and Electrical Engineering, Vol. 2, No. 1, February, 2010
1793-8163

Tag Name Structure-based Clustering of XML Documents

Mohamad Alishahi, Mahmoud Naghibzadeh and Baharak Shakeri Aski

***Abstract*—The concern of this paper is to extract knowledge from XML documents. The motivation is the existence of large amount of XML documents and its exploding trend and the need to summarize this information into a more abstract and usable documentations. Data mining is one of the most effective ways of knowledge extraction which has received fresh attentions. Many algorithms have been developed for the clustering of XML documents. We have focused on one of the most promising algorithm called XCLS and have directed our efforts on improving its clustering quality and performance. An improved adaptation of the algorithm called XCLS+ is devised. Both algorithms are implemented and evaluated It is shown that the performance of the new algorithm is enhanced in comparison to the previous one.**

***Index Terms*—Clustering, Data mining, Level similarity, Level structure.**

I. INTRODUCTION

Nowadays the World Wide Web is a huge platform of data and this data is created, stored and transferred incrementally. XML (eXtensible Markup Language) documents are one of the best tools for representing and transferring data because of their flexibility and self description, and many of the information resources use them. Since XML tags describe the structural and semantic concepts of information in texts, these documents are known to be semi-structured. This semi structure and irregularity causes problems for knowledge extraction from this kind of documents. So we need data mining techniques such as clustering, classification and association rules to extract knowledge. One of the most important advantages of data mining techniques is improving the speed and accuracy of the XML based search engines. We can refer to IEEE paper collection as an example and so searching a specific term in these papers has better results [1] – [4].

XML data exists in two forms: XML documents and XML schemas. XML schema defines structure while XML document defines data [5]. We can say that XML documents are instances of a XML schema and this schema includes acceptable elements, attributes, number of element occurrence and other constraints. We have to pay attention that among large number of XML documents only the ones that correspond to correct schemas are valid. There are many languages to describe the structure and content of these documents such as DTD (Document Type Definition) and XSD (XML Schema Definition). XSD has more capabilities than DTD [2]. For example, we have produced a DTD and its corresponding XML document for simple bookstore information in Fig. 1. Representing and modeling XML documents is one more subject to be considered. There are three methods for this aspect: tree structure, graph and time series. Among these, the tree structure is most common as XML documents have hierarchical organization defined by the provider and this organization more naturally corresponds to tree structure.

Figure.1 An instance of DTD schema

Fig. 2 shows an XML document and its corresponding tree structure. In this paper we focus on methods that use this structure. Section 2 is related works for XML clustering and in section 3 we briefly describe how XCLS algorithm that incrementally clusters XML documents, works. In section 4 we represent our algorithm and evaluate and compare it with XCLS in section 5. Section 6 is conclusion and future works.

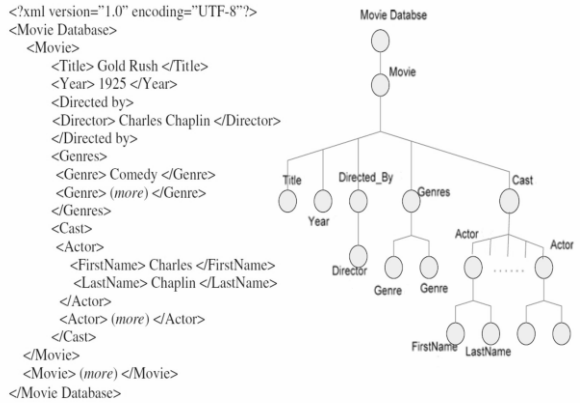


Figure.2 A sample XML document with its tree

Manuscript received June 6, 2009.

M. Alishahi is the member of Young Researchers Club-Islamic Azad University, Mashhad Branch, Mashhad, Iran (Corresponding author, Mobile: +989153171983, email: alishahi@ymail.com).

M. Naghibzadeh is with the Department of Computer Engineering, Ferdowsi University of Mashhad, Mashhad, Iran (e-mail: naghibzade@um.ac.ir).

B. Shakeriaski is with the Department of Computer Engineering, Islamic Azad University-Ramsar, Mashhad, Iran (e-mail: shakeriaski.b@gmail.com).

II. Related Works

Clustering XML data is more complicated than common text data as XML allows inserting structural and conceptual aspects into document content. A XML document includes tags and data. Tags describing names of elements contain concepts as text data. Besides document, structure tags also show the relationship between elements [1]. There are two types of algorithms for specifying XML documents similarities. First are structure-based only algorithms and second are structure and content algorithms. In [4] there is a classification of the methods that have been used in XML document clustering so far. Current methods are usually implemented using models based on neural networks, vector, and similarity. The foundation of neural network based models is recursive neural networks and self organizing map. Vector based models use converting XML document format to vector and similarity based models define a similarity criterion between documents to cluster them [4]. Using similarity based model is more common because of its generality. So [2] demonstrates this model in two ways: structure level and element level.

Similarity in structure level measures and describes three set of data:

1) Structure and level similarity between documents
2) Structural similarity between documents and schemas
3) Structure and level similarity between schemas

In similarity checking we study how similar is a document to other documents and schemas. But in element level we check the similarity between the elements of the two schemas. Element level uses XML document schemas for clustering. Generally the main difference between these two methods is that element level specifies the similarity between tree elements (particularly the similarity in the name structure) whereas structure level studies the similarity of the whole tree and ignores the elements details [2].

In this paper we divide XML document clustering algorithms into two groups:

4) Pair wise methods
5) Incremental methods

Pair wise based algorithms are more common which first create a similarity matrix for each pair of documents. This matrix is initialized by a criterion for measuring similarity between two documents. Finally after completing the matrix we can use a general clustering algorithm such as K-means to locate a document in its proper cluster. Vice versa we specify the similarity to each existing cluster for each entering document and if the similarity is more than a threshold then it is placed in the corresponding cluster. Else a new cluster is created and the document is placed in it. Generally the main differences between pair wise and incremental algorithms are time complexity and application. In the best case, pair wise algorithms are from order of two but incremental algorithms work in linear time and are useful for online environments. In the next subsection we briefly introduce pair wise clustering algorithms and in next section we describe an incremental algorithm with details because our algorithm presented in this paper is an improvement for this algorithm.

A. Pair Wise Algorithms for Clustering XML Documents

It is common for pair wise algorithms to use tree edit distance to calculate the similarity between two XML documents. The problem is to calculate the minimum distance between two trees T1 and T2 while using insert, delete and edit operations for each node of the tree. We consider a cost for each of these operations that are computed depending on the node tags. So in order to specify the similarity between two XML documents, it suffices to calculate the minimum cost of converting T1 tree to T2 tree by a sequence of operations [2]. In other words, we calculate how many operations are required to convert T1 to T2. This cost is the similarity between T1 and T2.

The technique we mentioned above is the basis for indicating the similarity between XML documents. But we must pay attention to the time complexity and high cost of this method when we have to cluster many XML documents as we must compute the distance between each pair of documents.

In order to solve the problem above, many algorithms are designed that try to decrease the cost. Some of them are mentioned below:

1) In [7] a method is represented to summarize the XML tree. So by decreasing the number of repetition and nested elements and defining a new criterion for structural distance, a better performance is achieved.
2) In [8] using S-graph theory to represent XML documents and suggesting a new criterion for distance an improvement clustering on clustering is accomplished. The reason of this claim is encrypting S-graph to a bit string which is simple to use clustering.

III. XCLS Incremental Algorithm

XCLS (XML Clustering by Level Structure) algorithm tries to cluster XML documents by considering their structures. Of course this algorithm represents XML structure in a new way called level structure. Level structure only uses the elements or tags of the XML document and ignores their contents and attributes. Using a global criterion for computing similarity is a considerable point in incremental methods. Comparing the entering document with existing clusters necessitates having a global criterion. This global criterion uses level structure. Level structure acquired from tree representation shows the existing nodes of each level [1].

Fig. 3 shows the level structure of a XML document. In level structure we re-label nodes to integers for ease.

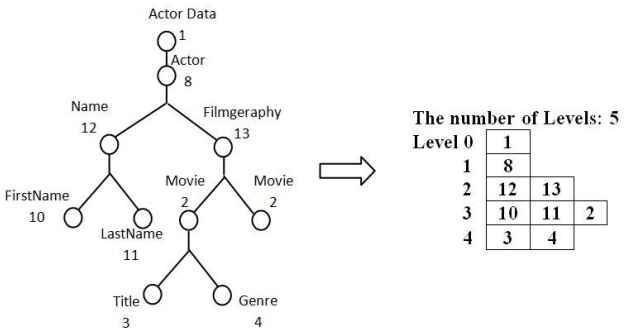


Figure.3 A sample XML document with its level structure

The formula (1) is used to specify the similarity between two objects (an object can either be a document or a cluster) focuses on common nodes on each level of two objects.

$$LevelSim_{1->2} = \frac{0.5\times\sum_{i=0}^{L-1} CN_1^i \times (r)^{L-i-1} + 0.5\times\sum_{j=0}^{L-1} CN_2^j \times (r)^{L-j-1}}{\left(\sum_{k=0}^{L-1} N^k \times (r)^{L-k-1}\right)\times Z} \quad (1)$$

The parameters used in formula (1) are:

1) Z Size of the cluster, i.e., the number of documents within the cluster.
2) CN_1^i Sum of occurrences of every common element in the level i of the object 1.
3) CN_2^j Sum of occurrences of every common element in the level j of the object 2.
4) N^k Number of elements in level k of the document.
5) r Base Weight: the increasing factor of weight. This is usually larger than 1 to indicate that the higher level elements have more importance than the lower level elements.
6) L Number of levels in the document.

In other words, the *LevelSim* mentioned in (1) is based on common nodes in different leveles of objects. In next subsections we explain clustering by comparing objects according to level similarity.

A. The Steps of Matching the Elements of Two Objects

1) Start with searching for common elements in the first level of both objects. If at least one common element is found, mark the number of common elements with the level number in object 1 (CN_1^0) and the number of common elements with the level number in object 2 (CN_2^0), then go to step 2. Otherwise, go to step 3.
2) Move both objects to next level (level i++, level j++) and search for common elements in these new levels; if at least one common element is found, mark the number of common elements with the level number in object 1 (CN_1^i) and the number of common elements with the level number in object 2 (CN_2^j), then go to step 2. Otherwise, go to step 3.
3) Only move object 2 to next level (level j), then search for common elements in the original level (i) of object 1 and the new level (j) of object 2. If at least one common element is found, mark the number of common elements with the level number in object 1 (CN_1^i) and the number of common elements with the level number in object 2 (CN_2^j), then go to step 2. Otherwise, go to step 3.
4) Repeat the process until all levels in either object have been matched.

We can calculate the similarity between two objects by above stages and then cluster accordingly. Fig. 4 shows a sample example for calculating the level similarity between two documents. The calculated measure by the given formula is always between zero and one. If there is no common element between two objects in the current level then it proceeds to the next level of the second object. So the level similarity $LevelSim_{1\to2}$ differs from $LevelSim_{2\to1}$. We must calculate both cases and consider the greater value as the similarity between two objects.

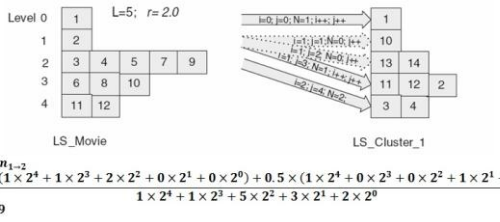


Figure.4 An example for calculating the similarity

B. Clustering by Level Similarity

The general procedure for clustering is that the first document enters and forms the first cluster. The next document enters and the similarity between the two existing documents is computed. If the calculated number is greater than the user provided threshold then the entering document is placed in the existing cluster. Pay attention that in this case two documents with possibly different structures are located in the same cluster. So we should merge their structures to keep the algorithm incremental. To merge the documents we congregate the elements of the same levels into a new level structure called cluster level structure. In fact the level structure of a cluster is the representative of the existing documents in the cluster. But if the computed similarity is less than the user provided threshold then the entering document forms a new cluster. This procedure is repeated and for each entering document we compare it with existing clusters. If the maximum calculated similarity is greater than the user provided threshold then the document is placed in the most similar cluster, otherwise it forms a new cluster. Finally, this procedure is performed to the last document when the clustering task is finished.

Two of the advantages of this algorithm is its linear time and its quality of clustering. But one of its disadvantages is specifying the threshold by user as different threshold may cause different results. Also the sequence of entering the documents is the other problem in this algorithm that may cause different clustering results. To obviate the latter problem a reassignment phase is performed at the end of the algorithm. In this phase some of the documents are selected randomly and are assigned to the clusters, again. This task is repeated until there is no improvement in placing documents in two consecutive iterations. Fig. 5 shows the clustering algorithm [1] with the reassignment phase.

XCLS ALGORITHM

Input:
XML Documents
LevelSim_Threshold
Output: A set of clusters.
Method:
/* Phase 1 – Allocation */
For all XML documents to be clustered
(1) Read the next document (represented as the level structure);
(2) Compute the levelSim between the document and each existing cluster;
(3) Assign the document to an existing cluster if maximum of *LevelSim(s)* is found between

two objects and *LevelSim* > *LevelSim_Threshold;*

(4) Otherwise, form a new cluster containing the document.

/* **Phase 2 – Reassignment** */

For all XML documents

(1) Read the randomly selected document)i.e. level structure);

(2) Compare the levelSim between the document and each existing cluster;

(3) Reassign the document to an existing cluster if maximum of *LevelSim(s)* is found between two objects and *LevelSim* > *LevelSim_Threshold;*

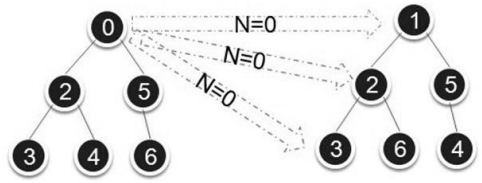
(4) Otherwise, form a new cluster containing the document.

/* **Stop** if there is no improvement in two iteration */

Figure.5 XCLS clustering algorithm

IV. XCLS+

As we study the XCLS algorithm in more detail we observed some problems that are XCLS's deficiency. Fig. 6 shows an example for which the process of matching the elements of two objects by XCLS algorithm is unable to find all common elements of the two objects; consequently the real similarity cannot be obtained. In this example there are six elements in each object with five of them being common but the similarity that is calculated by XCLS is equal to zero. This is not reasonable.



$$LevelSim_{1\to 2}=\frac{0.5\times(0\times 2^2+0\times 2^1+0\times 2^0)+0.5\times(0\times 2^2+0\times 2^1+0\times 2^0)}{1\times 2^2+2\times 2^2+3\times 2^2}=0$$

Figure.6 An example for the problem of the XCLS algorithm

Thus in this paper we try to suggest a new algorithm named XCLS+ and this algorithm uses a new method for matching the common elements between two objects. As we described XCLS we see that XCLS orders the elements by level and in each level tries to find the common elements and fill the CN_1^i and CN_2^j parameters, but in XCLS+ we try to order the elements by tag names and according to algorithm 1 in Fig. 7 we are able to find all common elements.

ALGORITHM 1 The Steps of Matching the Elements in XCLS+

Input:

O1, O2: Two Novel Level Structures represented **as Table** which **Ordered by** *TagName, Level*

OL1: The number of rows of *O1* in {$OL1_1, OL1_2, \ldots, OL1_w$ }

OL2: The number of rows of *O2* in {$OL2_1, OL2_2, \ldots, OL2_z$ }

Output: *CN1* [1.. w] **as number**

CN2 [1.. z] **as number**

Method:

(1) **Repeat**

(2) Compare each row *i* of *O1* with each row *j* of *O2*

(3) **if** (*O1.TagName = O2.TagName)*

{

CN1 [$OL1_i$] ++;

CN2 [$OL2_j$] ++;

Go to the next rows of both objects *O1, O2;*

}

(4) **if** (*O1.Tagname* > *O2.TagName*)

Go to the next row of just *O2*

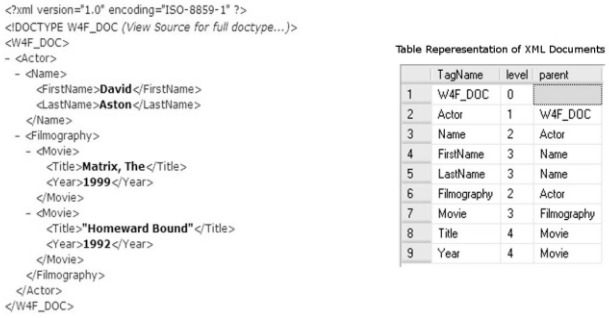
(5) **else**

Go to the next row of just *O1*

(6) **until** all rows of both objects checked

Figure.7 The XCLS+ matching process algorithm

Note that we store all the XML documents as data base tables and we can easily order them by tag name or level. Fig. 8 is an example of a XML document that is stored by data base tables. By storing the tag names, their levels, and their parents we can easily retrieve the original documents.



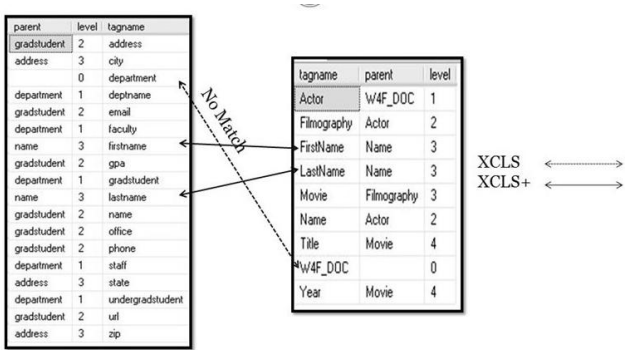
```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!DOCTYPE W4F_DOC (View Source for full doctype...)>
- <W4F_DOC>
  - <Actor>
    - <Name>
        <FirstName>David</FirstName>
        <LastName>Aston</LastName>
      </Name>
    - <Filmography>
      - <Movie>
          <Title>Matrix, The</Title>
          <Year>1999</Year>
        </Movie>
      - <Movie>
          <Title>"Homeward Bound"</Title>
          <Year>1992</Year>
        </Movie>
      </Filmography>
    </Actor>
</W4F_DOC>
```

Table Reperesentation of XML Documents

	TagName	level	parent
1	W4F_DOC	0	
2	Actor	1	W4F_DOC
3	Name	2	Actor
4	FirstName	3	Name
5	LastName	3	Name
6	Filmography	2	Actor
7	Movie	3	Filmography
8	Title	4	Movie
9	Year	4	Movie

Figure.8 XML document and its corresponding table

The description of algorithm 1 is as follow: we order the tables by tag names then we try to find the common tag names disrespectful of levels of matched tags. For each pair of matched tags found, by using the level attribute of the tags the CN_1^i and CN_2^j parameters are computed. Since we first find all possible matched tags and then consider the levels of the matched tags to compute CN_1^i and CN_2^j, all possible pairs of equal tag names of the two objects are found. However, based on XCLS only matched tags of matched levels are found. Fig. 9 shows the difference between XCLS and XCLS+.



parent	level	tagname
gradstudent	2	address
address	3	city
	0	department
department	1	deptname
gradstudent	2	email
department	1	faculty
name	3	firstname
gradstudent	2	gpa
department	1	gradstudent
name	3	lastname
gradstudent	2	name
gradstudent	2	office
gradstudent	2	phone
department	1	staff
address	3	state
department	1	undergradstudent
gradstudent	2	url
address	3	zip

No Match

tagname	parent	level
Actor	W4F_DOC	1
Filmography	Actor	2
FirstName	Name	3
LastName	Name	3
Movie	Filmography	3
Name	Actor	2
Title	Movie	4
W4F_DOC		0
Year	Movie	4

XCLS ⟷
XCLS+ ⟷

Figure.9 An example for the steps of XCLS+ algorithm

The other problem of XCLS is considering no structural information (parent child relation) between the elements of consecutive levels. The XML documents could be group into two categories. One is homogeneous XML documents and the other is heterogeneous [9]. Heterogeneous XML documents are derived from different Document Type Definitions (DTDs) and represent semantically different information. Such XML documents, in most cases, do not share the same node tags, as they belong to different semantic categories. However, in some cases, such as in a movie DTD and in an actor DTD, the XML documents may share some node tags but can have different parent/child relationships. On the other hand, homogeneous XML documents are usually derived from sub-DTDs of the same DTD. These documents usually have the same set of nodes in similar levels and their differences lie in the relation between nodes of consecutive levels. XCLS method does not pay attention to these relations. Thus clustering homogeneous XML documents by XCLS does not lead to reliable results. In Fig. 10 we provide an example that shows this problem.

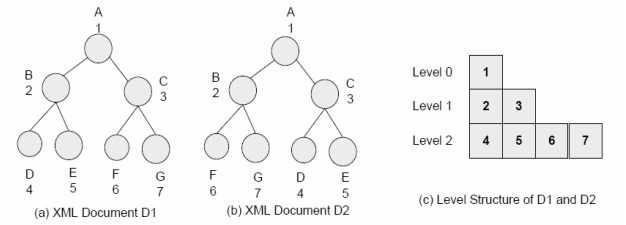


Figure.10 Equal XCLS similarity of any object to (a) and (b)

As it is observed from Fig. 10, there are two objects that have exactly the same elements in each level. Thus if we compare them using XCLS algorithm, we will get the highest possible similarity array. However, these two objects are different. For example, in the first object element B is the parent of D but in the second object its parent is C.

In XCLS+ we suggest a new level structure that reflects both element and structural information. With this suggestion, level similarity formulas have to be modified. This is described in the next subsections.

A. *Suggested level structure in XCLS+*

The new level structure in XCLS+ is based on idea that in addition to the elements name we should consider the information about their parents, too. Each level of the new structure will store element names and their parents. The new level structure for an example document is created in Fig. 11. As it is seen in the figure, the new structure denotes, for example, that the parent of element 11 in level three is element 13 in level two.

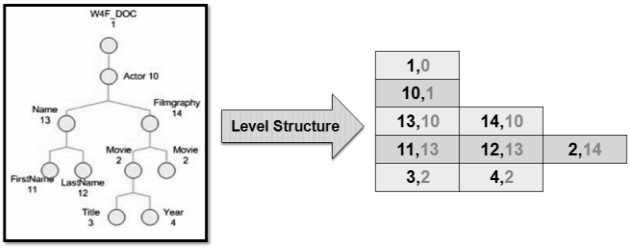


Figure.11 The new level structure in XCLS+

As we suggested a new level structure in XCLS+ the level structure of the clusters will also be changed. According to incremental algorithms we know that after each document's cluster is recognized, we must combine the level structure of the document and its corresponding cluster. This will update the cluster's level structure to be used for new coming objects. In Fig. 12 we present the combination of two level structures in XCLS+ algorithm. The level structure of a cluster is the union of elements of the two objects at each level. Note that, in XCLS+, there are two items of information (element's name and parent's name) in each box of the level structure. In case of two similar elements with different parents, both boxes have to be included in the combined level structure. For example, in Fig. 12 at level 2 of both level structures we have element 14 with different parents thus in level structure of the cluster we use element 14 twice, each time with a different parent.

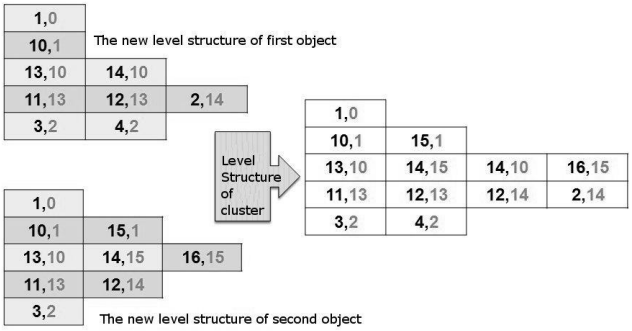


Figure.12 The new level structure for clusters in XCLS+

B. *New level similarity formula*

Due to structural information that we added to XCLS+ it is necessary to enter this information in level similarity formula. With some changes in (1) we provide (2) as a new level similarity formula.

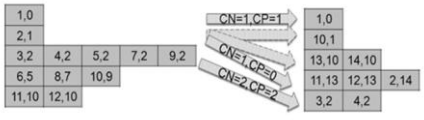
$$LevelSim_{+1,2} = \frac{0.5\times\sum_{i=0}^{L-1}\left(CN_1^i + CP_1^i\right)\times(r)^{L-i-1} + 0.5\times\sum_{j=0}^{L-1}\left(CN_2^j + CP_2^j\right)\times(r)^{L-j-1}}{\left(\sum_{k=0}^{L-1}N^k\times(r)^{L-k-1}\right) + 0.5\times\left(\sum_{i=0}^{L-1}CP_1^i\times r^{L-i-1} + \sum_{j=0}^{L-1}CP_2^j\times r^{L-j-1}\right)} \quad (2)$$

Most of the terms used in Formula (2) are exactly the same as the corresponding terms in Formula (1). However, there are some new terms which are described in the following:

1) CP_1^i Number of occurrences of all common elements in level *i* of the object 1 which have the same parent.

2) CP_2^j Number of occurrences of all common elements in level *j* of the object 2 which have the same parent.

With respect to Relation (2), it is clear that the number of common elements which have the same parents will play an important role in calculating the similarity measure, especially when are clustering homogeneous XML documents. To compare the new similarity formula with the old one, we bring an example that calculates the similarity with both formulas in Fig. 13. As we used the parent-child information we achieve a more reliable similarity measure. By investigating the documents of Fig. 13, we see that in level 2 of the first object and level 4 of the second object we have two elements (3, 4) that have the same parent too, so these objects similarity measure should be higher.



$LevelSim_{1\to2} = 0.4259$

$$LevelSim_{+1\to2} = \frac{0.5\times(2\times2^4+1\times2^3+4\times2^2+0\times2^1+0\times2^0)+0.5\times(2\times2^4+0\times2^3+0\times2^2+1\times2^1+4\times2^0)}{1\times2^4+1\times2^3+5\times2^2+3\times2^1+2\times2^0+0.5\times((1\times2^4+2\times2^2)+(1\times2^4+2\times2^0))} = 0.6483$$

Figure.13 Similarity measure of two objects in XCLS+

V. EXPERIMENTAL EVALUATIONS

In order to show the improvement in XCLS+ we try to compare it with XCLS algorithm under similar circumstances. All the experiments for both algorithms were done on the same data sets which are consisting of

heterogeneous XML documents. These data sets are standard and are usually used for XML clustering algorithm's evaluation. We will compare both algorithms for the aspects of quality of clustering and running time.

Both XCLS and XCLS+ algorithms and also the evaluation criteria were implemented with Microsoft visual studio 2008 platform using C# language and all documents of data sets are stored as tables in Microsoft SQL server 2005 data base. All the experiments were done in a machine with 3.2GHZ Intel Celeron CPU and 1GB of RAM. In the next two subsections we will describe the data sets and the evaluation criteria which measure the quality of clustering and finally the results will be discussed.

A. Data set

All the experiments in this paper were done on two sets of data. First, is the heterogeneous XML documents obtained from [10], [11] and second, is homogeneous XML documents obtained from [12].

The heterogeneous XML Files data set contains 460 XML documents. The documents are from various domains that are shown in TABLE I. The number of tags varies from 10 to 100 in these sources. The nesting level varies from 2 to 15. Majority of these domains contains a number of different documents that have structural and semantic differences. Hence, even though documents are from the same domain, they might not be considered similar enough to be grouped into the same clusters [1].

TABLE.IDATA SET OF VARIOUS DOMAINS WITH AMOUNT OF 460 XML FILES

class	No.
Movie	74
University	22
Automobile	207
Bibliography	16
Company	38
Hospitality message	24
Travel	10
Order	10
Auction	4
Appointment	2
Document page	15
Bookstore	2
Play	20
Club	12
Medical	2
Nutrition	2

For homogeneous XML documents, we used the dblp DTD to create 4 sub DTDs. We created those sub-DTDs in such a way that the derived synthetic XML documents would share the same node tags in most of the levels, but also contain some different relationships in some of the levels. More precisely, the first sub DTD contained only the <*proceedings*> node in the first level of the dblp DTD and the corresponding nodes in the sublevels, the second sub-DTD contained only the <*article*> node, the third sub-DTD contained only the <*book*> node and the last sub DTD contained the <*masterthesis*> node. We created a total number of 164 homogeneous XML documents: 20 derived from the first sub-DTD, 22 derived from the second sub-DTD, 32 derived from the third sub-DTD, and 90 derived from the fourth sub-DTD [9].

B. Evaluation Criteria

The evaluation criteria for XML documents clustering are categorized in two groups: the first one is internal cluster quality evaluation criteria and the second one is external cluster evaluation criteria. Internal evaluation criteria have no information about the data sets and try to determine how similar the documents are. But here the problem is that the similarity between the documents is measured by the level similarity in XCLS and by a novel level similarity in XCLS+. So as we used two different measures it may not be reasonable to compare these two algorithms by internal evaluation criteria. Consequently, in this article we use the external evaluation criteria which are based on the comparison of clusters' classes to known external classes. Entropy, purity, and FScore are used as external evaluation criteria in this article, which are taken from [1], [13] described as follow.

The entropy measure looks at how the various classes of documents are distributed within each cluster. Given a particular cluster C_i of size n_i, the entropy of a cluster is defined as:

$$E(C_i) = \frac{1}{\log k} \sum_{r=1}^{k} \frac{n_i^r}{n_i} \log \frac{n_i^r}{n_i}$$

Where k is the number of classes in the dataset, and n_i^r is the number of documents of the rth class that are assigned to the ith cluster. The entire clustering solution's entropy is the sum of the individual cluster entropies weighted according to the cluster size. The formula is given here:

$$Entropy = \sum_{i=1}^{k} \frac{n_i}{N} E(C_i)$$

A perfect clustering solution will be the one that leads to clusters that contain documents from only a single class, in which case the entropy will be zero. In general, smaller the entropy value, the better the clustering solution is.

Purity measures the extent to which each cluster contains documents primarily from one class. The formula of purity of a cluster is:

$$P(C_i) = \frac{1}{n_i} \max(n_i^r)$$

The purity of entire clustering solution is obtained as a weighted sum of the individual cluster purity:

$$Purity = \sum_{i=1}^{k} \frac{n_i}{N} P(C_i)$$

In general, larger the value of purity, the better the clustering solution is.

FScore is a combination of precision and recall. Precision defines the rate of correct matches in the generated solution, and Recall defines the rate of correct matches in the model solution.

Given an XML document category Z_r with the n^r number of similar XML documents, and a cluster C_i with the n_i number of similar XML documents categorized by the clustering algorithm. Let n_i^r be the number of documents in cluster C_i belonging to Z_r.

Precision (correctness) is defined as: $p(Z_r, C_i) = n_i^r / n_i$

Recall (accuracy) is defined as: $r(Z_r, C_i) = n_i^r / n_r$

The *FScore* combining precision and recall with equal weights is defined as:

$$F(Z_r, C_i) = \frac{p(Z_r, C_i) \times r(Z_r, C_i)}{p(Z_r, C_i) + r(Z_r, C_i)} = \frac{2n_i^r}{n_i + n_r}$$

The FScore value of a category Z_r is the maximum FScore value attained in any cluster of the clustering solution. Hence, the FScore of the overall clustering solution is defined as the sum of the individual class FScores weighted differently according to the number of documents in the class:

$$FScore = \frac{\sum_{r=1}^{k} n_r F(Z_r, C_i)}{N}$$

Where k is the number of clusters. A good clustering solution has the FScore value close to one.

C. Quality Evaluation

TABLE II shows all the experiments and assessments about the quality of clustering on heterogeneous XML files. As shown in table, we tried to compare the algorithms under different circumstances and each time we run the algorithms with different parameters and record the results. The results declare that XCLS+ clusters the documents with better quality or at least the same.

TABLE.II THE EXPERIMENTS AND ASSESSMENTS ABOUT THE QUALITY OF CLUSTERING FOR HETEROGENEOUS XML DOCUMENTS

ALGORITHM	R FACTOR	THRESHOLD	ENTROPY	PURITY	FSCORE
XCLS	2	0.75	0.06	0.87	0.89
	2	0.6	0.07	0.83	0.84
	1.5	0.75	0.08	0.85	0.86
	1.5	0.6	0.08	0.86	0.86
XCLS+	2	0.75	0	1	1
	2	0.6	0.03	0.92	0.93
	1.5	0.75	0.04	0.90	0.91
	1.5	0.6	0.08	0.86	0.86

According to this issue that XCLS+ calculate the similarity more precisely in homogenous XML documents the result will be better because in homogenous XML files the relations play an important role so XCLS+ is capable of proper clustering. If we concentrate on TABLE III that evaluate both XCLS and XCLS+ on homogeneous XML files in different situation we will observe that the quality of clustering is much better as we use XCLS+. Note that as we decrease threshold and r factor the clustering quality decrease too. If the threshold level is reduced, the measured similarity, of a document against a cluster, by the two methods, XCLS and XCLS+, may pass the threshold level. However, if the threshold level is high XCLS and XCLS+ may act differently and will not produce similar results. For a given document and a given cluster, one may decide to assign the document to the cluster while the other may decide not to assign the document to the cluster. In such cases, the quality of the XCLS+ clustering is more accurate. In addition, if r is low, clustering quality slightly decreases. In such cases, we can say that the effect of r surpasses the effect of parent-child relation.

TABLE.III THE EXPERIMENTS AND ASSESSMENTS ABOUT THE QUALITY OF CLUSTERING FOR HOMOGENEOUS XML DOCUMENTS

M	FACTOR	LD	Y		
XCLS	2	0.75	0.26	0.49	0.49
	2	0.6	0.18	0.56	0.57
	1.5	0.75	0.25	0.50	0.51
	1.5	0.6	0.18	0.56	0.58
XCLS+	2	0.75	0.08	0.87	0.89
	2	0.6	0.09	0.84	0.85
	1.5	0.75	0.10	0.81	0.81
	1.5	0.6	0.12	0.79	0.80

D. Running time Evaluation

The time complexity for both algorithms XCLS and XCLS+ is equal because the basis of both algorithms is equal and the difference is the way they use to find the common elements between two documents. Thus as the XCLS algorithm assert time complexity is linear but the running time is different because as explained previously XCLS+ algorithm is able to find all common elements just by one iteration but XCLS needs two iterations. According to recent explanation our algorithm must have better running time than the XCLS. After calculating the running time for both heterogeneous and homogenous XML documents of our data sets Fig. 14 and 15 prove our claim.

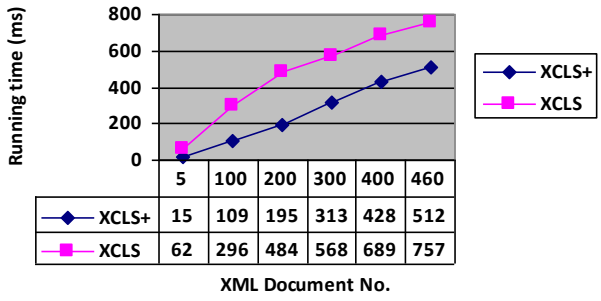


Figure.14 Comparing running time of XCLS and XCLS+ for heterogeneous data set

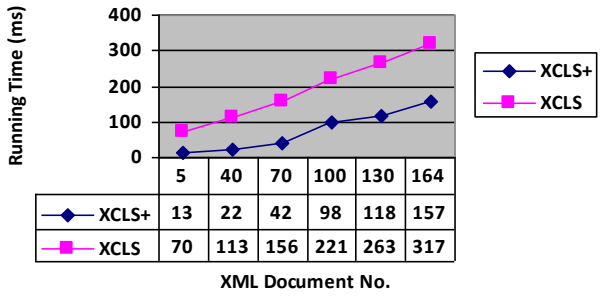


Figure.15 Similarity measure of two objects in XCLS+ for homogeneous data set

VI. CONCLUSION AND FUTURE WORKS

We presented an algorithm for clustering XML documents based on the XCLS algorithm. We compared our algorithm with the XCLS algorithm and perceived that it works better than XCLS or in the worst cases it performs like XCLS. Although this algorithm solves some of the problems of the XCLS algorithm but one of its remained problem is specifying the threshold that may affect its performance, remarkably. Future works for this paper can be: specifying the threshold by the algorithm automatically, considering the

concepts of elements for clustering, designing a two phases algorithm for clustering as in one phase using pair wise algorithms and in another using incremental algorithms. Finally, as the role of XML documents becomes more important, we hope our algorithm would be useful for clustering XML documents especially in search engines that search huge resources of the World Wide Web.

REFERENCES

[1] G. R. Nayak, "Fast and effective clustering of XML data using structural information," Knowl. Inf. Syst. 14(2), pp. 197-215, 2008.
[2] R. Nayak, "XML Data Mining: Process and Applications," in Song, Min and Wu, Yi-Fang, Eds, Idea Group Inc. /IGI Global, 2008.
[3] R. Nayak, T. Tran, "A Progressive Clustering Algorithm to Group the XML Data by Structural and Semantic Similarity," IJPRAI 21(4), pp. 723-743, 2007.
[4] L. Denoyer, P. Gallinari, "Report on the XML mining track at INEX 2005 and INEX 2006: categorization and clustering of XML documents," SIGIR Forum 41(1), pp. 79-90, 2007.
[5] S. Abiteboul, P. Buneman and D. Suciu, "Data on the Web: From Relations to Semistructured Data and XML," Morgan Kaumann, CA, 2000.
[6] S. Flesca, G. Manco, E. Masciari, L. Pontieri and A. Pugliese, "Fast detection of XML structural similarities," IEEE Trans. Knowl. Data Engin. 7(2), pp. 160–175, 2005.
[7] T. Dalamagas, T. Cheng, K. Winkel, T. K. Sellis, "A methodology for clustering XML documents by structure," Inf. Syst. 31(3), pp. 187-228, 2006.
[8] W. Lian, D. W. Cheung, N. Mamoulis, S. Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," IEEE Trans. Knowl. Data Eng. 16(1), pp. 82-96, 2004.
[9] P. Antonellis, C. Makris, N. Tsirakis, "XEdge: clustering homogeneous and heterogeneous XML documents using edge summaries," SAC 2008, pp. 1081-1088, 2008.
[10] The Wisconisn's XML data bank. Accessed from: http://www.cs.wisc.edu/hiagara/data.html. Cited sept 2004.
[11] The XML data repository. Accessed from: http://www.cs.washington.edu/research/xmldatasets/. Cited Sept 2004.
[12] http://www.informatik.uni-trier.de/~ley/db/about/dblp.dtd
[13] Y. Zhao, G. Karypis, "Criterion functions for document clustering: Experiments and analysis," Department of Computer Science, University of Minnesota, Minneapolis, 2001.

Author Biography

Mohamad Alishahi received master science of computer engineering in 2008 from Islamic Azad University of Mashad, Iran. His current research area is data mining and specially clustering; now he is working on a project about the role of data mining in power distribution companies.