ISSN No. 0976-5697

Volume 4, No. 10, Sep-Oct 2013

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Mining XML Document Based on Structure

Mehdi G. Duaimi*& YasirAbdAlhamed
Computer Science Department
University of Baghdad
Baghdad, Iraq
mehdi_duaimi@ymail.com*, yasir_sh79@yahoo.com

***Abstract*:**With the growing number of XML documents on the Web it becomes essential to effectively organize these XML documents in order to retrieve useful information from them. A possible solution is to apply clustering on the XML documents to discover knowledge that promotes effective data management, information retrieval and query processing. This paper presents a framework for clustering XML documents by structure. Modelling the XML documents as rooted ordered labelled trees, we study the usage of structural distance metrics in hierarchical clustering algorithms to detect groups of structurally similar XML documents. We suggest the usage of structural summaries for trees to improve the performance of the distance calculation and at the same time to maintain or even improve its quality.

Keywords: XML, Tree Similarity Measure, structural summary, clustering, DTDs

I. INTRODUCTION

XML documents are becoming ubiquitous because of their rich and flexible format that can be used for a variety of applications ranging from scientific literature and technical documents to handling news summaries utilize XML in information representation and exchange.

More than 50 domain-specific languages have been developed based on XML [1], Such as MovieXML for encoding movie scripts, GraphML for exchanging graph structured data, Geography Markup Language (GML) for expressing geographical features and interchanging them over the Internet, Twitter Markup Language (TML) for structuring the twitter streams, Chemical Markup Language, Mathematics Markup Language (MathML) and many others [2].

XML has also been used to represent the web-based free-content encyclopedia known as Wikipedia, which has more than 3.4 million XML documents, in the last four years, the INEX (Initiative for the Evaluation of XML retrieval) has focused on clustering large collections of Documents using representations of structure documents.

The increased popularity of XML has raised many issues regarding the methods of how to effectively manage the XML data and retrieve these XML documents in large collections. A possible solution to the problem of handling large XML collections is to group similar XML documents. This taskof grouping in data mining is referred to as clustering. Clustering task group unknown data into smaller groups according to the data commonality without having any prior knowledge about the dataset. The clustering of similar XML documents has been perceived as potentially being one of the more effective solutions to improve document handling by facilitating better information retrieval, data indexing, data integration and query processing [3]. In spite of its potential, there are several challenges in clustering XML documents. Unlike the clustering of text documents or flat data, clustering of XML documents is an intricate process and consequently the most commonly used clustering methods for text clustering cannot be used for clustering these documents. This is due to the fact that XML documents are semi-structured in nature and have a flexible structure as well as their content showing the semantics. The semi-structured nature of XML data requires the computation of similarity by including their structural similarity [4].

II. RELATED WORKS

a. SangeethaKutty MCIS (*Faculty of Science and Technology at Queensland University of Technology Brisbane, Queensland, Australia*) 2011, introduces the structural similarity in the form of frequent subtrees and then uses these frequent subtrees to represent the constrained content of the XML documents in order to determine the content similarity[4].

b. Joe Tekli et al (*University of Bourgogne*) 2007, introducing the notion of structural commonality between subtrees, putting forward an algorithm for its discovery) an efficient algorithm was introduced for computing tree-based edit operations costs able to consider, via the sub-tree commonality notion, XML sub-tree structural similarities) a prototype was developed to evaluate and validate our approach[5].

c. Lian et al. (*Faculty of Information Technology Queensland University of Technology*) 2004, represents the XML document as graph-based and measures the common set of nodes and edges appearing between the documents. To retain the structure information from the XML documents [6].

d. Jeong&Keun, Leung et al., Jeong&Keun 2008,use the sequential pattern mining to extract the frequent paths from XML documents and then use them for clustering[7].

e. Shen and Wang (*University of Wisconsin – Madison, WI, U.S.A.*) 2003, breaks the XML documents into a number of macro-path sequences where each macro-path contains the properties of an element such as its name, attributes, data types and textual content. A matrix similarity of the XML documents is then generated based on the macro-path similarity technique [8].

f. Nierman&Jagadish, Dalamagas et al.(*University of Michigan*) 2004 have been proposed to represent the XML documents as tree-based and use the tree edit distance to measure the similarity between the documents using the document structure [9].
g. Lee et al. 2002, introduces a complex computational technique to map the element similarity between the schemas by considering the semantics, immediate descendent and leaf-context information. Its purpose is to be used as the pre-processing stage for applications such as data integration [10].
h. Cobéna et al. [2002] proposed XyDiff, an algorithm for detecting changes in XML documents. The algorithm first computes a signature (i.e., hash value) and a weight (i.e., subtree size) for every node in both documents in a bottom-up fashion (the root nodes of the two documents end up with the largest weights). Nextstarting with the root nodes of the two documents XyDiff compares the signatures of the two nodes. If they are equal, the two nodes are matched; otherwise, their child nodes will be inserted into a priority queue in which the subtrees with the largest weights are always compared first [11].

III. XML

Extensible Markup Language (XML) is an abbreviated version of Standard Generalized Markup Language (SGML), for the exchange of structured documents over the Internet. Unlike HTML, XML readily enables the definition,transmission, validation, and interpretation of data between differing computing platforms and applications. XML permits people in a specialized field, such as chemistry,finance,or environmental data collection, to develop XML schema that define the markup language for the exchange of specialized data unique to their fields[12]. XML schema is the primary data format supported for data exchange by the State/EPA Environmental Information Exchange Network (Exchange Network).

XML is extensible, meaning a developer can *extend* the language by devising new tags to describe and share data in any specialized way desired as long as the new tags follow the XML syntax defined by the W3C XML specification. XML is very useful for organizations that do not share but need to develop a common data exchange format. Its extensibility provides flexibility in developing exchange formats in XML schema, provided all partners agree on the data format and definitions of the data it contains

IV. CLUSTERING

Is the task of grouping a set of objects in such a way that objects in the same group (called cluster) are more similar to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics. Cluster itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem. The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results.

V. CLUSTERING OF XML DOCUMENTS

After establishing a motivation to cluster XML documents,we turn our attention to the development of an effective clustering algorithm. In this section, we define a method to summarize XML documents such that a simple and efficient
Similarity metric can be applied. Then, we show how this metric can be used in combination with a clustering algorithm to divide a large collection of XML documents into groups according to their structural characteristics.
Although our definitions and methodology assume a database of XML documents, they can be seamlessly applied for any collection of semi structured data

VI. DOCUMENT REPRESENTATION

XML documents can be represented as labelled trees. In trees representing documents, internal nodes are labelled by element/attribute names and leaves are labelled by textual content. In the tree representation, attributes are not distinguished from elements, both are mapped to the tag name set; thus, attributes are handled as elements. Attribute nodes appear as children of the element they refer to and, for what concerns the order, they are sorted by attribute name, and appear before all sub-elements “siblings”.

XML document elements may actually refers to, that is, contain links to, other elements. Including these links in the model gives rise to a graph rather than a tree. Even if such links can contain important semantic information that can be exploited in evaluating similarity, most approaches disregard them and simply model documents as trees.

VII. STRUCTURAL SUMMARIES

In order to gain in performance, Structural summaries are produced using a dedicated repetition/nesting reduction process. The structural summary of an XML tree comes down to a modified tree in which the redundancies due to nested repeated and repeated XML nodes are eliminated ,The tree is traversed using pre-order traversal. For the current node, check if there is an ancestor with the same label. If there is no such ancestor, go on to the next node. If there is such ancestor, then move all current nodes’ subtrees to that ancestor. The subtreesare added at the end of the ancestor’s child list so that we will traverse these nodes later. Nothing will be moved if the current node is a leaf, as shown in***code list (1)***

Pseudo code list (1) Reduce Nesting.

Input

TreeNode : as TreeNode

Output

Tout : as TreeNode

Procedure

1. *Let first node as parent node*
2. *If have children then*
   For (first-child To last-child)
   Let child node as parent node
   End for
3. *For (first-child End if*
4. *If have children then*
   For (first-child To last-child)
   If has nesting then
   Delete all nesting
   else
   take other child as parent node
   end if
   End for
   Else
   End if
5. *End*

The algorithm start by taking the root node as a parent node and check, if it has a child, then, taking the first child as new parent and check if the new parent has child if it has, then check, if it has nesting, then delete nesting if does not go to second child until reaching the last node.

The aim of Repetition Reduction is to reduce the repeated nodes in the original tree. The tree is traversed using pre-order traversal. At each node, check whether the path from the root to the node already exists or not by looking it up in a hash table keeping the paths. If there is no such a path, store this node in the hash table, with its path being the index. If there is already one such path in the hash table, then this node is a repeated node, and in that case:

a) move all its subtrees to the destination node that we find in the hash table by using the path as index,
b) add the subtrees at the end of the destination node's child list to traverse these subtrees later, and
c) Delete the current node and start to traverse the subtreeswhich have been moved to the destination node.

After traversing all the nodes that have been moved, we go on to traverse the right siblingOf the node which is deleted.If there is no such node the traversal ends.

Repetitions reduction request only a pre-order traversal on the original tree. And Pseudo code list (3.3).

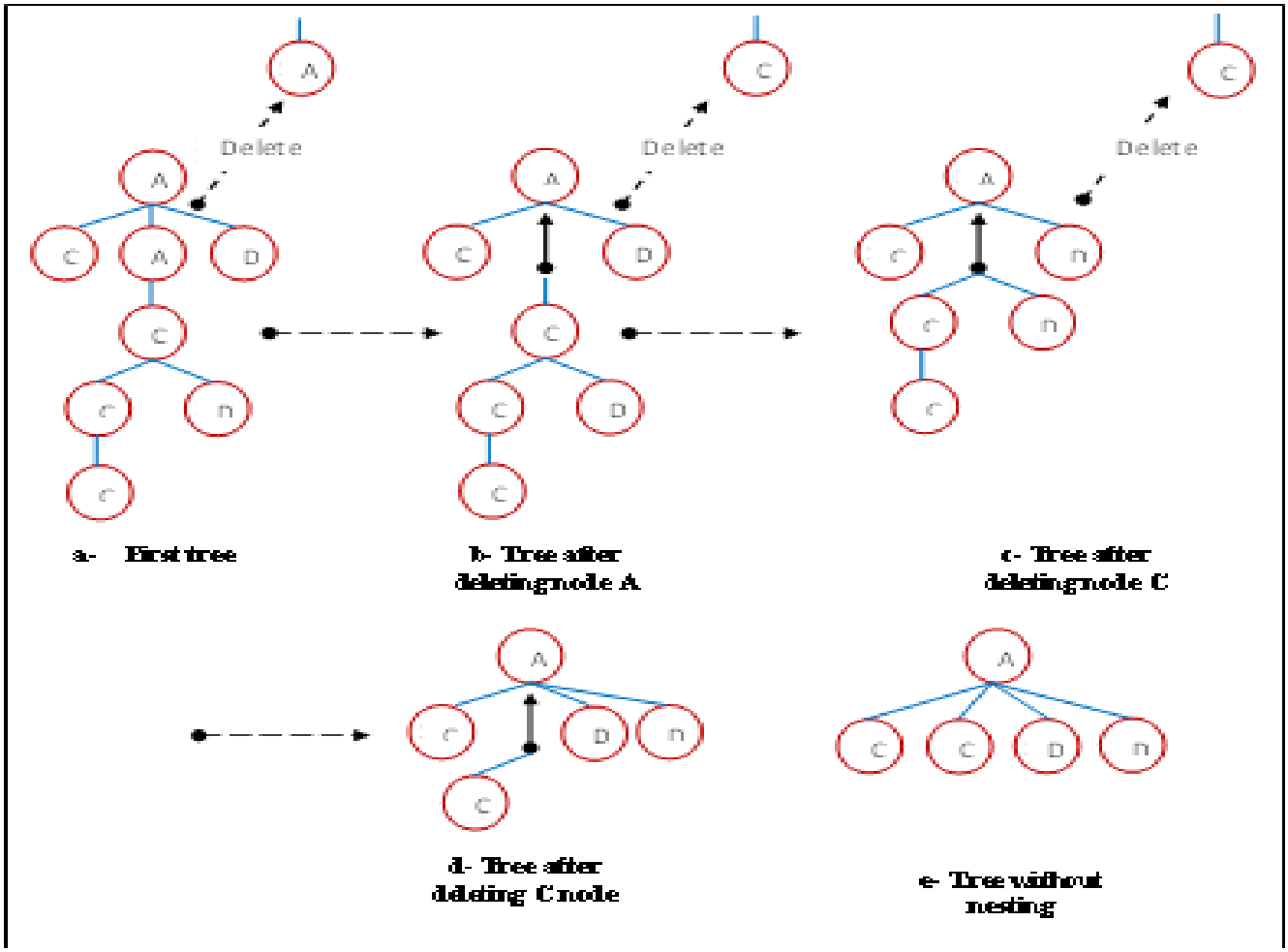

Figure: 1 nesting reduction

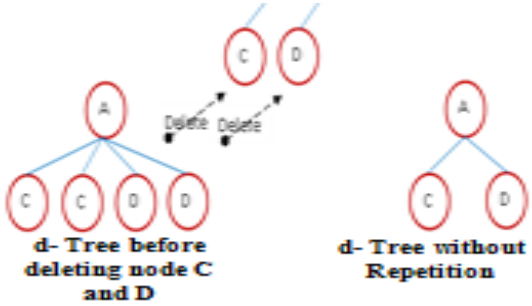

Figure (2): Repetition Reduction

Pseudo code list (2) reduces Repeat.

Input

TreeNode : Node
CPath :currentPath
h : hash table

Output

Tout : as TreeNode

Variables

S : string
destination: Treenode

Procedure

```
1. S = CPath + "/" + node-name;
2. If  hash table is not containing this path (S)
then
        Add this path to hash table
       For (first-child to last-child)
          Call reduce Repeat Procedure
       End for
3. else
        destination = the Node in hash table which  contain- the same  path
        For  (TreeNode -first-child To TreeNode last-child)
            Move child to destination
            Delete  TreeNode
        End for
4. End if
5. End
```

The algorithm Creates path by adding Cpath to node name, checks hash table if it has not this path it adds it to hash table and checks all children, else, it takes the node which is the same in hash table and adds all children of treeview to this node lastly, it deletes treeview and returns to loop until reaching the last node.

VIII. TREE SIMILARITY MEASURE

The editing operations available in the tree edit distance (computing the distance between two trees) are replacing, deleting, and inserting a node. To each of these operations a cost is assigned, that can depend on the labels of the involved nodes. The problem is to find a sequence of such operations transforming a tree T1 into a tree T2 with minimum cost. The distance between T1 and T2 is then defined to be the cost of such a sequence. In this work, we consider Chawathe's (II) algorithm as the basic point of reference for tree edit distance algorithms. This algorithm has quadratic complexity (O (MN), M and N are the dimensions of the matrix that represents the edit graph). Also, it fits well in the context of XML data, since it permits insertion and deletion only at leaves, as show in figure (3) and Pseudo code list (3).

Pseudo code list (3) CalculateDistance.

Input

S: Tree Node
T: Tree Node

Output

D [i, j]: array with two dimensions contain the distance

Procedure

```
1.  For (S-first-child To S-last-child )
      Calculate distance for first row
      D[i, 0] = D[i - 1, 0]
      +CalculateDistance(S.Nodes [i - 1]) +1
2.  end for
3.  For (T-first-child To T-last-child )
      Calculate distance for first column
      D [0, j] = D [0, j –1] +
      CalculateDistance (T.Nodes [j - 1]) + 1
4.  end for
5.  For (S-first-child To S-last-child )
       For (T-first-child To T-last-child )
            m1 = D[i - 1, j] + CalculateDistance
            (S.Nodes[i - 1]) + 1
            m2 = D[i, j - 1] + CalculateDistance
            (T.Nodes[j - 1]) + 1
            m3 = D[i - 1, j - 1] +
            CalculateDistance
            (S.Nodes[i - 1],T.Nodes[j - 1])
            Select  minimum distance D[i, j] =
            Math.Min(Math.Min(m1,m2), m3)
       end for
    end for
6.  end
```

This procedure calculates distances between XML documents,it stores distance in matrix first loop will calculates first row in matrix and the second one Calculates the first column, while the third loop calculates the other rows and columns for the matrix of distances.

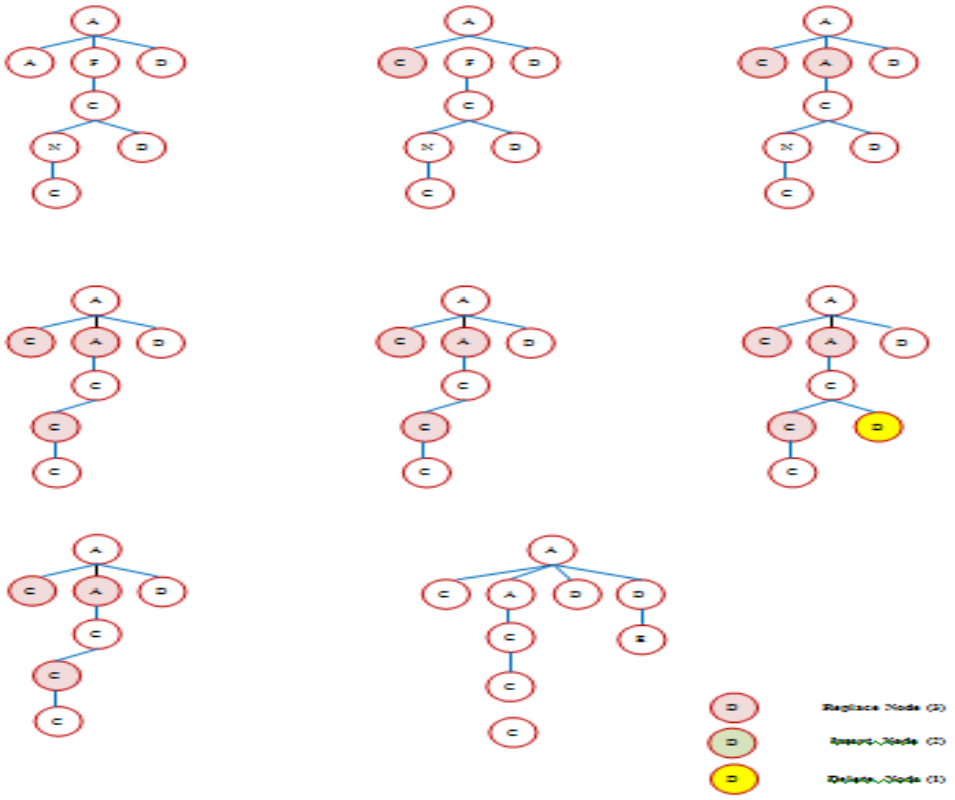

Figure (3): The minimum cost to transform T1 to T2 equal (6) unit

IX. CLUSTERING

In data mining, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters. Strategies for hierarchical clustering generally fall into two types:

- ***a.*** ***Agglomerative****:* This is a "bottom up" approach: each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy.
- ***b.*** ***Divisive****:* This is a "top down" approach: all observations start in one cluster, and splits are performed recursively as one moves down the hierarchy.

In general, the merges and splits are determined in a greedy manner. The results of hierarchical clustering are usually presented in a dendrogram. Distance between any two clusters can be computed using any of the following criterions:

- ***a)*** ***Single-linkage*** clustering (also called the *connectedness* or *minimum* method), we consider the distance between one cluster and another cluster to be equal to the shortest distance from any member of one cluster to any member of the other cluster. If the data consist of similarities, we consider the similarity between one cluster and another cluster to be equal to the greatest similarity from any member of one cluster to any member of the other cluster.
- ***b)*** ***Complete-linkage*** clustering (also called the *diameter* or *maximum* method), we consider the distance between one cluster and another cluster to be equal to the greatest distance from any member of one cluster to any member of the other cluster.
- ***c)*** ***Average-linkage*** clustering, we consider the distance between one cluster and another cluster to be equal to the average distance from any member of one cluster to any member of the other cluster.

X. MST AND SINGLE-LINKAGE CLUSTERING

After calculating the tree edit distance the prim's algorithm is used to find minimum spinning tree (MST) ,Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connectedweightedundirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. figure (4.a) dataset contains 8 XML files and the weight between each pair, which is calculated by tree edit distance algorithm, minimum spanning tree (MST) of a graph implemented on this dataset as shownin figure (4.b), the single link clusters for a clustering threshold equal four can be identified by deleting all the edges with weight w≥ 4 from the MST of G. The connected components of the remaining graph are the single link clusters, There are 1 connected component that include nodes (A,C,D,E,F,G) and 2 Nodes (B,H) which are not connected to other nodes they be considered as single-node clusters .This indicates the presence of 3 clusters: cluster 1 with (A,C,D,E,F,G) as members ,cluster 2 with (B) as member and cluster 3 with (H) as member. As shown in figure (4.c).

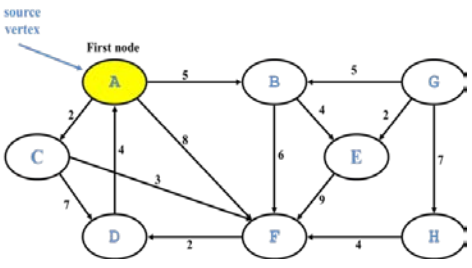

Figure (4.a): dataset contains 8 XML files and the weight between each pair

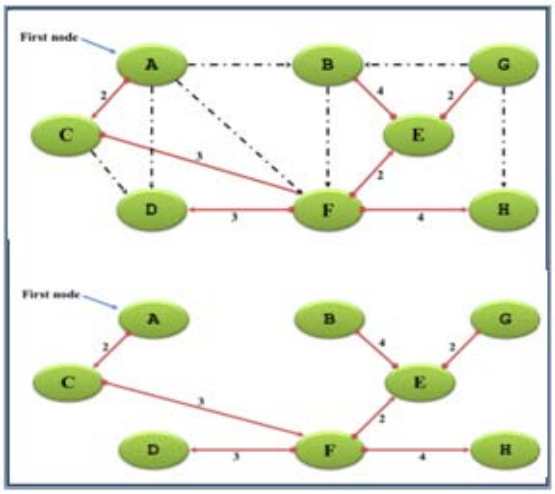

Figure (4.b) minimum spanning tree (MST) of a graph

MST is implemented in figure 4.c where it started from (A→C) (C→F) (F→D) (F→E) (F→H) (E→B) (E→G).

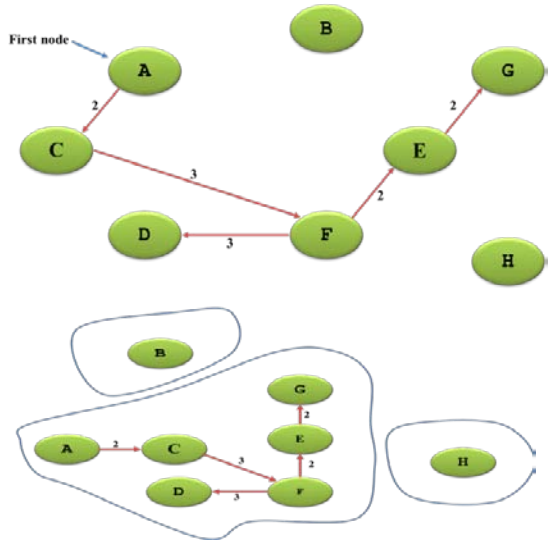

Figure (4.c) single linkage clustering

Single linkage clustering isimplemented in figure (4.c) where it deletes each link having a threshold bigger or equal to 4 , therefore; it deletes links (F →H) and (E → B), that generates three clusters (A, C, D, F, E, G), (H) and (B).

Prim's algorithm is a greedy algorithm that finds a minimum spanning tree for a connectedweightedundirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

The above algorithm involves four stages to calculate the MST

1. Select node zero as first node
2. reset select node row
3. Select minimum weight on this node's columns by function (getminimum)
4. Add select node name and weight of link to list (Mst.add) depending on function (getminimum),go to step 2.

Pseudo code list (4) prim's algorithm

Input

minimum no = 0 ;
Matrix [i, j]: array with two dimensions contain the distance

Output

D [i, j]: array with two dimensions contain the distance
Mst :list

Procedure

1. For (i=0 to matrix-length) zero(matrix, arr1[i]) minimum_no = getminimum(matrix, arr1, out Where, ref
-next_node)
If (minimum_no != 0) then
sum + = minimum_no
end if
2. add the name and weight of link to list (Mst.add)
3. end for
4. End

Pseudo code list (5) getminimum

Input

arr [,]: array with two dimensions contain the distance
T: Tree Node

Output

D [i, j]: array with two dimensions contain the minimum distance

Variable

w = 0
Intmax : variable equal to the array length

Procedure

```
1.      int max = arr.GetLength(1)
2.      for ( 0  to  index.Length)
for (0  to max)
if arr[index[j],i]  And big or equal  0) then
temp = arr[index[j], i]
else
exit ;
endif
end for
3.      end for
4.      w= 0;
5.      for ( 0  to  index.Length)
for (0  to max)
if (arr[index[j], i] <= temp &&arr[index[j], i]
>= 0)
temp = arr[index[j], i]
w= i
s = index[j]
return temp
end if
end for
6.      end for
7.      End
```

The getminimum function returns the minimum no in the row where step 2 is uses to check column and takes the first value bigger than zero. After that step 5 selects the minimum no in the row.

XI. EXPERIMENTAL RESULTS

Experiments are conducted on real and synthetic XML documents. Two sets of 1000 documents were generated

from 10 real-case and synthetic DTDs, using an adaptation of the IBM XML documents generator. We varied the MaxRepeats parameter to determine the number of times a node will appear as a child of its parent node. For a real dataset, we considered the online version of the ACM SIGMOD Record. We experimented on a set of 203 documents corresponding to OrdinaryIssuePage.dtd (80 documents), roceedingsPage.dtd (23 documents) And IndexTermsPages.dtd (100 documents).In this section the performance will be examined on both the real and synthetic XML documents by using our algorithm, when using threshold equal to 5 the number of cluster that will appear is seven, with high PR and R values on synthetic XML document as shown in table (1).

Table 1: Clustering process on synthetic data with threshold = 5

Cluster No.	***DTD***	***Synthetic***		
		a	***b***	***c***
1	fruitbasket.dtd population.dtd personal.dtd customer.dtd	400	37	0
2	bookstore.dtd	83	0	17
3	memo.dtd	100	0	0
4	tvschedule.dtd	100	0	0
5	newspaper.dtd	100	0	0
6	recipes.dtd	80	0	20
7	catalog.dtd	100	0	0
PR=0.963	***R=0.963***			***F-value=0.963***
threshold =5				

While three clusters concluded from real data which produce PR, R values less than synthetic data because that some files produced are mis-clustered as shown in table (2).

Table (2): Clustering process on real data with threshold = 5

Cluster No.	***DTD***	***Real-Life***		
		a	***b***	***c***
1	IndexTermsPages.dtd	100	0	0
2	OrdinaryIssuePages.dtd	29	0	51
3	ProceedingsPage.dtd	16	16	7
PR=90%	***R=71%***	***F-value=0.8***		
threshold =5				

XII. TIMING ANALYSIS

We note that the process of simplification of structure for XML files reduced the time required to calculate the distance between two files to 95%, where the blue line represent time line to calculate distance without reduce structure and red line represent the time line with reduce structure

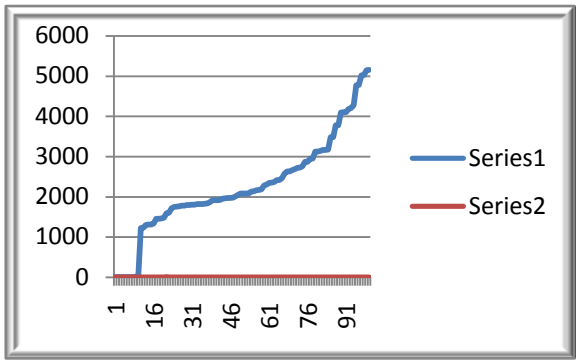

Figure (5): Timing Results (to compute pairwise distance) for big data with reduce and without reduce structure

XIII. CONCLUSION

XML is becoming a standard in many applications because of its universal and powerful tree structure. On the internet for example, unstructured documents are being replaced by such structured documents, so that approaches that have been designed to tackle internet resources need to be revisited in order to take advantage of the new structured nature of the documents.

This work successfully applied clustering methodologies for grouping XML documents which have similar structure, by modeling them as rooted ordered labeled trees, and utilizing their structural summaries to reduce time cost while maintaining the quality of the clustering results. We performed extensive evaluation using synthetic and real data sets, providing timing analysis as well as precision PR and recall R values for each test case.

Our results showed that:

a) XML document is better represented as tree model by using DOM, because DOM parser is faster than SAX because it access whole XML document in memory.

b) By use Structural summaries the time needed to calculate the tree distances is decreased for whole clustering procedure.

c) Chawathe's algorithm with structure summaries improves high performance and shows excellent clustering quality.

d) Excellent results were obtained when assigning new incoming XML documents to already discovered clusters, instead of applying a clustering method again to the whole set of documents, including the new ones, Re-clustering is expensive since all pairwise distances should be calculated again.

XIV. REFERENCES

[1]. R. Cover. Xml applications and initiatives. http://xml.coverpages.org/ xmlApplications.html, 2005.

[2]. F. M. Suchanek, A. S. Varde, R. Nayak, and P. Senellart. The hidden web, xml and the semantic web: scientific data management perspectives. Pages 534–537.ACM, (2011).

[3]. T. Tran, S. Kutty, and R. Nayak.Utilizing the structure and content information for xml document clustering. In S. Geva, J. Kamps, and A. Trotman,

editors, Advances in Focused Retrieval, volume 5631 of Lecture Notes in Computer Science, pages 460–468. Springer Berlin / Heidelberg, (2009).

[4]. S. Kutty, T. Tran, R. Nayak, and Y. Li. Clustering XML documents using closed frequent subtrees: A structural similarity approach. In N. Fuhr, J. Kamps, M. Lalmas, and A. Trotman, editors, Focused Access to XML Documents, volume 4862 of Lecture Notes in Computer Science, pages 183–194. Springer Berlin / Heidelberg,(2011).

[5]. Joe Tekli, Richard Chbeir, KokouYetongnon: Efficient XML Structural Similarity Detection using Sub-tree Commonalities(2007).

[6]. A. Doucet and M. Lehtonen. Unsupervised classification of text-centric XML document collections. In 5th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX, pages 497–509, (2006).

[7]. Tran, Tien and Nayak, Richi and Bruza, Peter DCombining structure and content similarities for XML document clustering. In: 7th Australasian Data Mining Conference, Glenelg, South Australia. (27-28 November 2008).

[8]. Shen and Wang (Wisconsin Univ.): X-Diff: an effective change detection algorithm for XML documents (2003).

[9]. Nierman, A. and H.V. Jagadish, Evaluating structural similarity in XML documents. University of Michigan (2004).

[10]. Lee, J.W., K. Lee and W. Kim, 2002. Preparations for semantics-based XML mining. Proceedings of the IEEE International Conference on Data Mining,Nov. 29-Dec. IEEE Xplore Press, San Jose,USA, pp: 345-352. DOI: 10.1109/ICDM.989538(2002).

[11]. Cobénaet. al. GrégoryCobéna, Serge Abiteboul, and Amelie Marian. Detecting Changes in XML Documents. In International Conference on Data Engineering (ICDE'02), San Jose, California, (2002).

[12]. EPA United States Environmental Protection, Agency the State/EPA Environmental Information Exchange Network (2002).

:

