A Framework for Clustering and Dynamic Maintenance of XML Documents

Ahmed Al-Shammari(✉), Chengfei Liu, Mehdi Naseriparsa, Bao Quoc Vo, Tarique Anwar, and Rui Zhou

Swinburne University of Technology, Melbourne, VIC 3122, Australia
{aalshammari,cliu,mnaseriparsa,bvo,tanwar,rzhou}@swin.edu.au

Abstract. Web data clustering has been widely studied in the data mining communities. However, dynamic maintenance of the web data clusters is still a challenging task. In this paper, we propose a novel framework called XClusterMaint which serves for both clustering and maintenance of the XML documents. For clustering, we take both structure and content into account and propose an efficient solution for grouping the documents based on the combination of structure and content similarity. For maintenance, we propose an incremental approach for maintaining the existing clusters dynamically when we receive new incoming XML documents. Since the dynamic maintenance of the clusters is computationally expensive, we also propose an improved approach which uses a lazy maintenance scheme to improve the performance of the clusters maintenance. The experimental results on real datasets verify the efficiency of the proposed clustering and maintenance model.

Keywords: Clustering · XML documents · Structure and content similarity · Dynamic maintenance

1 Introduction

XML has become a standard data exchange format these days, which provides interoperability and simplicity among Web-based services such as financial transactions, transportation, and healthcare services [10]. In general, these services are required to meet the minimum response time for transferring large amounts of XML data between users and the application providers. Therefore, identifying groups of users with similar requests can potentially reduce the required response time. Clustering is one of the most crucial techniques for organising the disseminated documents into groups based on their similarities [7,14]. Several studies have proposed different XML clustering models to support the compression and aggregation techniques in reducing the latency and bandwidth over the Web services [1,2]. For clustering XML documents, the similarities between them are computed and the measures are used to group them into clusters. When the new documents arrive, the existing clusters are to be updated and maintained dynamically in an efficient manner. For the dynamic maintenance, some clustering techniques such as the partitioning-based are inapplicable in the dynamic environment [11]. There are two reasons behind this. Firstly, these approaches assume a fixed number of clusters. However, in the dynamic environment the number of clusters may change frequently over a period of time. Secondly, the partitioning-based approach completely recalculates the cluster properties to update the clusters each time with the new XML documents. This complete re-calculation approach is inefficient. Conversely, the agglomerative (bottom-up) clustering method does not require a predefined number of clusters [4,5]. There has not been much works on developing methods for efficient maintenance of XML document clusters in a dynamic environment. Some studies have proposed clustering models for XML documents based on the structure and content similarity [12]. These models are inefficient for clustering XML documents because they require a long execution time to calculate the pairwise distance between the documents. For instance, fractal clustering models [2] use a fractal similarity method that needs to calculate the scale and offset factors to find the similarity between the XML documents. To address the above limitations, we introduce a novel framework for clustering and maintenance of the XML documents called XClusterMaint. The XClusterMaint framework includes the followings: (1) clustering, and (2) the clusters maintenance. The main contributions of this paper are summarised as follows:

– We propose a fast clustering model for the XML documents based on a combination of the structure and content similarity. The proposed model requires a low computational cost in comparison with the existing clustering models.
– We introduce an incremental approach for the dynamic maintenance of existing clusters when new documents arrive. The maintenance includes adding the new documents to their closest cluster, and updating the cluster properties.
– We further improve the performance of the dynamic maintenance of clusters by proposing a lazy maintenance scheme. It keeps track only of the unstable cluster spaces to minimize the computations.
– We validate the proposed framework with extensive experiments on real-world datasets and demonstrate the efficiency of our clustering and maintenance approaches.

The rest of the paper is organised as follows. Section 2 presents the related work, which is followed by the problem definition in Sect. 3, and a high-level solution sketch in Sect. 4. Thereafter, we present the initial clustering of XML documents in Sect. 5, and the dynamic cluster maintenance in Sect. 6. The experimental results are presented in Sect. 7, and the paper is finally concluded in Sect. 8.

2 Related Work

This section highlights the basic findings and gaps in the studies related to clustering methods. Many studies have addressed the problem of XML document clustering. XML document includes two main features: structure and content. However, most current XML clustering algorithms do not concentrate on both of these features due to their demand of more processing time and computational storage. Clustering of static XML documents based on the similarity/distance measure has attracted a great deal of attention. Static clustering approaches can be generally separated into the following categories: content-based approaches, structure-based approaches, and hybrid approaches. Clustering XML by structure and content features is more efficient and useful whenever XML documents share overlapping [6]. In addition, in a heterogeneous environment clustering by both features achieves a high performance in comparison with other approaches using content/structure only [12,15]. We focus on the clustering XML documents algorithms by considering both structure and content similarity since there is no work done on the dynamic maintenance of the XML documents clustering in the literature.

Yongming et al. (2008) [15] introduced a clustering technique for XML documents based on similarity measures which exploits both the structure and content features of XML data. The leaf path and nested elements of XML data are the major features forming the vector-based dataset before the clustering process is performed. Experiments showed that the performance of the extended vector space model is greater than the basic VSM, showing higher purity and lower entropy. Al-Shammary and Khalil (2011) [2] proposed Fractal clustering algorithms by structure and content similarity. These models do not require a pre-defined number of clusters. In the pre-processing, XML is transformed as a vector by using the tf.idf weighting scheme. Then, Fractal coefficient is used to find the similarity among vectors and finally group them based on minimum fractal mean square error. The experimental results have shown better performance of the proposed self-similarity model than other clustering techniques such as k-means and PCA combined with k-means.

However, partitioning-based clustering algorithms are inapplicable in the dynamic environment. The main reason is that the proposed algorithms assume a fixed number of clusters [11]. On the other hand, clustering algorithms [2,6] have also shown some drawbacks such as the lack of incremental maintenance for the existing clusters and disregard the content similarity of XML documents. Therefore, our work focuses on proposing a framework for both clustering and maintenance of the clusters over XML documents.

3 Problem Statement

Suppose we have a given set of XML documents $\mathcal{D} = \{\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3, ..., \mathrm{d}_{|\mathcal{D}|}\}$. We use two aspects to reflect an XML document: (a) data structure, and (b) data content. Therefore for a document $d_i \in \mathcal{D}$, we generate two vectors called $\vec{v}_i^s$ and $\vec{v}_i^c$ to represent the document structure and content properties respectively. Then, we combine these vectors to generate the XML vector that represents the corresponding XML document. The first problem that we address in this paper is defined as grouping these documents $\mathcal{D}$ based on their structure and content similarity to generate the initial set of clusters $\mathcal{C} = \{c_1, ..., c_{|\mathcal{C}|}\}$. These clusters contain the summarised information of the documents. To assign the

XML vectors into their proper clusters, we have to measure the distance between these vectors.

Definition 1 ***(Vector distance).*** *For two XML vectors $\vec{v}_1$ and $\vec{v}_2$, the vector distance (dist) is defined as their Euclidean distance presented in Eq. 1*

$$dist(\vec{v}_1, \vec{v}_2) = \sqrt{\sum_{i=1}^{n} \left(\vec{v}_1.w_i - \vec{v}_2.w_i \right)^2} \tag{1}$$

Here w_i is the weight of the ith entry in the XML vector. After measuring the distance between the XML vectors, we group these vectors into a set of clusters based on their distance measure.

Definition 2 ***(Cluster).*** *A cluster $c \in \mathcal{C}$ contains a set of XML vectors, and has the following properties: (a) the centroid m_c which is defined as the central mean vector of the c which is presented in Eq. 2, (b) the radius r_c^1 which is required a pre-defined to the set of clusters $\mathcal{C}$.*

$$\vec{m}_c = \frac{\sum_{i=1}^{|c|} \vec{v}_i}{|c|} \tag{2}$$

The second problem is incrementally maintaining the generated clusters when the new incoming documents arrive. Given a set of clusters $\mathcal{C}$ and an incoming document d_{new}, maintenance of the existing clusters defined as a set of updates for the cluster properties with tracking the affected XML vectors. In the tracking process, we may assign the affected XML vectors to the nearest clusters or we may initialise new clusters for these vectors. However, dynamic maintenance of the clusters incurs high computational costs. Therefore, there is a need for an efficient maintenance approach to improve the performance of the clusters maintenance.

4 The Solution Sketch

Figure 1 shows our proposed framework called XClusterMaint. Technically, XClusterMaint that takes care of both the clustering and the clusters maintenance. In the clustering, we start with traversing the order labelled XML tree, and then we generate the XML vector which is a combination of the structure and content vectors respectively. Technically, the term frequency-inverse document frequency (tf.idf) weighting scheme [8] is used to assign the weights to the terms of XML document, and the weights are stored in a vector matrix. Afterwards, Euclidean distance [3] is used for the similarity measurement by computing the minimum distance between the XML vectors. Then, the similar XML vectors are distributed into the clusters based on the agglomerative clustering model. The steps of clustering are presented in Sect. 5. In the clusters maintenance, an incremental approach is proposed for maintaining the properties of the existing clusters when the new incoming document arrives. Finally, we improve the efficiency of the clusters maintenance based on lazy maintenance scheme. The steps of clusters maintenance are presented in Sect. 6.

5 Clustering of XML Documents

In this section we focus on the main steps of the clustering of XML documents. The steps are as follows: (a) generating the vectors for the XML documents, (b) computing the similarity of the documents using their vectors, and (c) allocating the documents to their proper clusters.

5.1 Generating the XML Vectors

Any XML document in the dataset is modelled as a rooted tree. The XML tree has two kinds of nodes: (a) structure node and (b) content node. The structure refers to the nested tags (elements) that organise the content information while the content refers to the data values of the elements. We use depth-first search algorithm for traversing and indexing XML nodes level by level since all the nodes obtain a unique number as their index. To generate the XML vectors, we firstly generate the structure vector $\vec{v}^s$ and content vector $\vec{v}^c$.

Definition 3 ***(XML vector).*** *An XML document $d_i \in \mathcal{D}$ represented as a vector $\vec{v}_i = (\vec{v}_i^s.w_1, \vec{v}_i^s.w_2, ..., \vec{v}_i^s.w_m, \vec{v}_i^c.w_{m+1}, ..., \vec{v}_i^c.w_n)$ where $\vec{v}_i^s.w_j$ reflects the weight (total frequency) of the XML term j in the structure vector ($1 \leq j \leq m$) and $\vec{v}_i^c.w_{m+k}$ reflects the weight of the XML term k in the content vector ($1 \leq k \leq n - m$)[1].*

We select m terms for the structure vector and $n - m$ for the content vector, where m and n are usually application dependent and constrained by storage. For each term t in the structure or content vector, we use the tf.idf scheme to calculate the weight. The tf measures the frequency of the term t in the document denoted by $tf(t, d)$ while the *idf* measures the importance of the term

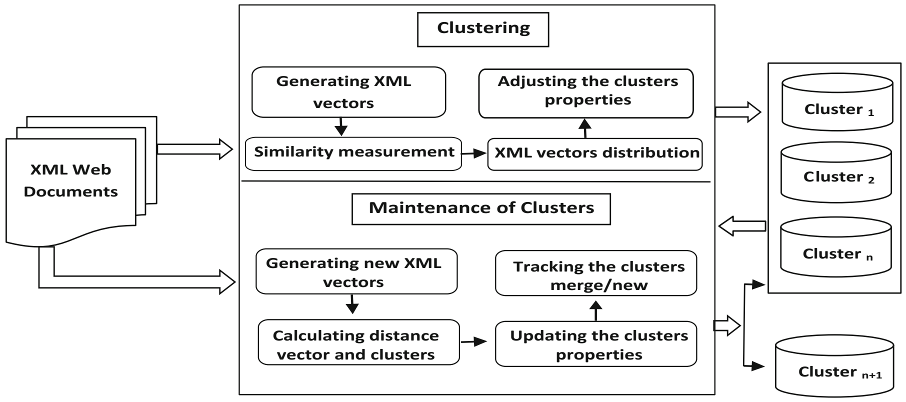

Clustering
Generating XML vectors
Adjusting the clusters properties
Similarity measurement
XML vectors distribution
XML Web Documents
Maintenance of Clusters
Generating new XML vectors
Tracking the clusters merge/new
Calculating distance vector and clusters
Updating the clusters properties
Cluster 1
Cluster 2
Cluster n
Cluster n+1

Fig. 1. The main components of XClusterMaint framework

[1] For simplicity, in this paper, we set $m = \frac{n}{2}$.

in the entire set of documents denoted by $idf(t) = log\frac{N}{df(t)}$ where $df(t)$ presents the number of documents that contain t in the dataset and N is the total number of XML documents in the dataset. Formula 3 presents the tf.idf formula for a term t in the document d.

$$w_{t,d} = tf(t, d) \times idf(t) \tag{3}$$

After generating $\vec{v}^s$ and $\vec{v}^c$, we combine these vectors to generate the XML vector of a document. This vector is used to measure the similarity score between the documents. The Eq. 4 presents the combination formula where α is the tuning parameter which trades off between the importance of the structure and content terms of the document.

$$\vec{v} = (\alpha \times \vec{v}^s, (1 - \alpha) \times \vec{v}^c) \tag{4}$$

For example purpose, assume we have 6 documents in the dataset. The XML vector for each document is generated by applying the combination formula presented in Eq. 4 using the tuning parameter $\alpha = 0.6$. For each vector, there are 3 weights for the structural terms and 3 weights for the content terms. Figure 1 presents the vectors for these documents.

Table 1. Vectors generation

Vectors	Structure			Content			Generating the vectors
	w_{t_1}	w_{t_2}	w_{t_3}	w_{t_1}	w_{t_2}	w_{t_3}	
v_1	0.3	0.6	0.2	0.4	0.9	0.3	(0.18,0.36,0.12,0.16,0.36,0.12)
v_2	0.2	0.3	0.3	0.6	0.3	0.2	(0.12,0.18,0.18,0.24,0.12,0.08)
v_3	0.3	0.6	0.2	0.6	0.9	0.3	(0.18,0.36,0.12,0.24,0.36,0.12)
v_4	0.3	0.6	0.4	0.4	0.9	0.3	(0.18,0.36,0.24,0.16,0.36,0.12)
v_5	0.2	0.2	0.3	0.6	0.3	0.2	(0.12,0.12,0.18,0.24,0.12,0.08)
v_6	0.2	0.5	0.2	0.3	0.8	0.3	(0.12,0.30,0.12,0.12,0.32,0.12)

5.2 Similarity Measurement

We use the data vectors to measure the similarity degree between their corresponding documents. The Euclidean distance measures the similarity between vectors that has several advantages in data clustering, such as simplicity and accuracy. Therefore, we use Eq. 1 to calculate the Euclidean distance between a pair of XML vectors, for instance $\vec{v}_1$ and $\vec{v}_2$. In order to find the similar documents, we measure the distance between all the XML vector pairs. The output of this step is the similarity score for each vector with all other vectors.

Example 1. The distance between the XML vectors in Table 1 are as follows: $dist(\vec{v}_2,\vec{v}_5) = 0.06$, $dist(\vec{v}_1,\vec{v}_3) = 0.08$, $dist(\vec{v}_1,\vec{v}_6) = 0.1019$,$dist(\vec{v}_1,\vec{v}_4) = 0.12$,$dist(\vec{v}_3,\vec{v}_4) = 0.1442$,$dist(\vec{v}_3,\vec{v}_6) = 0.1523$,$dist(\vec{v}_4,\vec{v}_6) = 0.1574$, $dist(\vec{v}_5,\vec{v}_6) = 0.2049$, $dist(\vec{v}_2,\vec{v}_6) = 0.2720$,$dist(\vec{v}_2,\vec{v}_3) = 0.3143$, $dist(\vec{v}_1,\vec{v}_2) = 0.3243$, $dist(\vec{v}_3,\vec{v}_5) = 0.3521$,$dist(\vec{v}_1,\vec{v}_5) = 0.3611$.

5.3 XML Vectors Distribution

After measuring the pairwise similarity between the XML vectors, we initialize the clusters for these vectors. To initialize the clusters, we start with sorting the pairwise distance between every two vectors, as shown in Example 1. The pair with the minimum distance is first checked whether it is less than a given threshold δ. The two vectors of this pair are merged into a cluster if it is true. This process is carried out to all the other vector pairs $(\vec{v}_i,\vec{v}_j)$ for which $\text{dist}(\vec{v}_i,\vec{v}_j) < \delta$, in the order of increasing pairwise distance. After this first round, the pairwise distance between the centroids of every two clusters are computed and sorted in increasing order. Following the same process as the first round, the clusters are merged if their distance is less than δ. These rounds are continued until all the pairs satisfying the pairwise distance condition have been processed. Considering Example 1 and $\delta = 0.21055$, the distance between $\vec{v}_1$ and $\vec{v}_3$, $\vec{v}_6$, and $\vec{v}_4$ is less than δ. While, the distance between $\vec{v}_1$ and $\vec{v}_2$, $\vec{v}_1$ and $\vec{v}_5$ is greater than δ. As a result, $\vec{v}_2$ and $\vec{v}_5$ have a high similarity and they will assign to the first clusterc_1. While, $\vec{v}_1$ $\vec{v}_3$, $\vec{v}_6$, and $\vec{v}_4$ will assign to the second cluster c_2.

6 Maintenance of the Clusters

As we mentioned earlier, the existing clustering algorithms are inapplicable for the dynamic maintenance of the clusters. Therefore, we propose two approaches for the incremental maintenance: (1) Baseline approach, and (2) Improved approach. When new XML documents arrive, these approaches maintain the properties of the existing clusters incrementally. The first approach uses an Eager Maintenance scheme of the Clusters (EMC) that tracks the entire XML vectors in the cluster. The second approach uses a Lazy Maintenance scheme of the Clusters (LMC) which tracks only a part of the XML vectors. The technical details for the proposed approaches are presented in Sects. 6.1 and 6.2 respectively. Maintenance of the clusters starts with the generating new XML vectors. Then, a decision is made to either assign the new XML vector $\vec{v}$ to the nearest cluster $c \in \mathcal{C}$ by calculating the minimum distance between the new vector and the existing clusters or initialise new cluster. Once the new XML vector is assigned to its nearest cluster, the new cluster centroid will be adjusted. The process of adjusting the new cluster centroid is performed incrementally by using the following formula:

$$\vec{m}_{c_{new}} = \frac{|c| \times \vec{m}_c + \vec{v}}{|c| + 1} \tag{5}$$

Where $|c|$ is the cluster size before adding the new XML vectors.

6.1 Baseline Maintenance Approach

We first present the baseline maintenance approach which is implemented based on EMC. We introduce a second yet smaller radius r_c^2, such that $(r_c^2 < r_c^1)$[2] to divide a cluster $c \in \mathcal{C}$ into two spaces as followings:

- Stable space (S): This space contains the XML vectors that reside in the second radius of a cluster r_c^2. The distance between these vectors and their centroid is less than or equal the second radius, where $dist(\vec{v}, \vec{m}_c) \leq r_c^2$.
- Boundary space (B): This space contains the XML vectors that reside out of the second radius r_c^2 but in the first radius r_c^1, i.e., $r_c^2 < dist(\vec{v}, \vec{m}_c) \leq r_c^1$. These vectors may be unstable and should be considered in the cluster maintenance. For instance, when the new centroid moves to a specific side, some of these vectors may be outside of the cluster boundary since the distance between these vectors and the new centroid is bigger than r_c^1 as shown in Fig. 2.

Basically, EMC uses two sets of maintenance operations as follows:

- $moveOut(\vec{v}, S, c)$ this operation moves the XML vector $\vec{v}$ from the stable space to the boundary space of the cluster c.
- $moveIn(\vec{v}, S, c)$ this operation moves the XML vector $\vec{v}$ from the boundary space to the stable space of the cluster c.
- $moveOut(\vec{v}, B, c)$ this operation moves the XML vector $\vec{v}$ from the boundary space to the outside of the cluster c.
- $moveIn(\vec{v}', B, c)$ this operation receives the XML vector $\vec{v}'$ from the outside of the cluster to the boundary space of the cluster c.

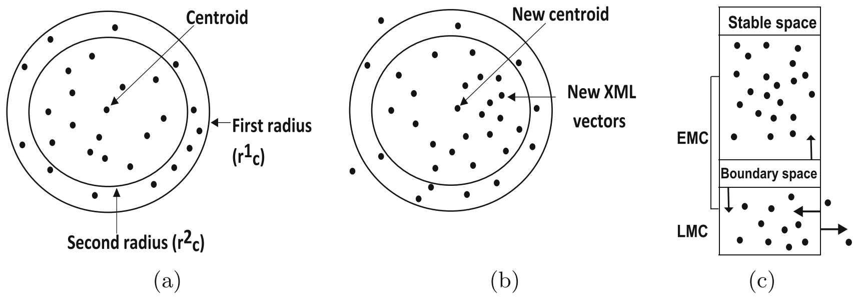


Fig. 2. Maintenance of the cluster

The EMC maintains the existing clusters by tracking the XML vectors in the stable and unstable spaces in the maintenance process. It includes two checking conditions: (1) cluster merge and (2) cluster initialisation. Firstly, a decision is

[2] r_c^2 is usually a fraction of r_c^1, i.e. $r_c^2 = \lambda r_c^1, \lambda \in (0, 1)$. In the paper, we find $\lambda = 0.8$ is fairly good.

made to merge the vectors that are located outside of the first cluster boundary to the nearest cluster. Secondly, a decision is made to initialise a new cluster if the minimum distance between the vector and the existing centroids is bigger than a distance threshold θ such that $min(dist(\vec{v}, \vec{m}_{c_i}, 1 \leq i \leq |\mathcal{C}|)) > \theta$.

6.2 Improved Maintenance Approach

The improved maintenance approach uses the Lazy Maintenance of the Clusters scheme (LMC) for maintaining the existing set of the clusters efficiently. LMC scheme only tracks the XML vectors in the unstable space. Therefore, this approach does not require to check the XML vectors in the stable space of a cluster, which cause it to accelerate the process of the cluster maintenance (see Fig. 2). The EMC and LMC tracking spaces are presented in Fig. 2. Clearly, the EMC works on both spaces while the LMC only works on the boundary space. According to this approach, we guarantee that the entire XML vectors do not require an adjustment in the stable space of the cluster. LMC uses a set of maintenance operations are as follows:

- $moveOut(\vec{v}, B, c_{new})$ this operation moves the XML vectors from the boundary space to the outside of the cluster.
- $moveIn(\vec{v}', B, c_{new})$ this operation receives the XML vectors from the outside of the cluster to the boundary space.

Theorem 1. *Let c and c_{new} denote the cluster before and after adding the new incoming documents respectively. Then r_c^1-r_c^2 denote the boundary threshold δ, and $\vec{m}_c$ is the old cluster centroid. There is no adjustment required for the stable space of the cluster c_{new} if the distance between the centroids $dist(\vec{m}_c, \vec{m}_{c_{new}})$ $\leq \delta$.*

Proof. Assume $\vec{v}$ is a vector that resides in the stable space of the cluster c such that $dist(\vec{v}, \vec{m}_c) = r_c^2$. Then if the distance of $\vec{v}$ with the new cluster c_{new} is $dist(\vec{v}, \vec{m}_{c_{new}}) > r_c^1$, thus $\vec{v}$ is out of cluster c_{new}. We prove that this can never happen. Since the new centroid $m_{c_{new}}$ is moved within the distance of (r_c^1-r_c^2), the maximum distance for the new vector $\vec{v}$ will be $dist(\vec{v}, \vec{m}_{c_{new}}) = r_c^2 + (r_c^1 - r_c^2) = r_c^1$ which means in the worst case it resides in the boundary of c_{new}. As a result, the distance of $\vec{v}$ in the new cluster c_{new} is $dist(\vec{v}, \vec{m}_{c_{new}}) \leq r_c^1$. Therefore, the primary assumption is not true and the vectors of the stable space will never move out of the c_{new}; thus, no adjustment required.

Basically, there are two possibilities might be happened after adjusting a new cluster centroid. Firstly, the new centroid of the cluster might be changed slightly. Secondly, The new centroid of the first cluster might be changed dramatically (it takes a sudden jump). For the first possibility, we proposed incremental maintenance of clusters is capable of maintaining the accumulated clusters incrementally. Secondly, recalculation for centroid is required if the distance between old and new centroids is bigger than the boundary threshold δ. Therefore, we propose a local re-calculation approach to recalculate the cluster properties. In particular, this approach is performed locally for a part of clusters. The distance between a pair of cluster centroids is calculated based on the vector distance formula 1.

Algorithm 1. Improved Maintenance

```
   Input  : $\mathcal{C} = \{c_1, c_2, c_3, ..., c_{|\mathcal{C}|}\}$, new document $d_{new}$, minimum distance
            threshold $\theta$, tuning Parameter $\alpha$
   Output: Updated set of clusters $\mathcal{C}_{new}$
1  $\vec{v}_{new} \leftarrow createVector(d_{new}, \alpha)$
2  $c_{near} \leftarrow$findNearestCluster($\vec{v}_{new}$,$\mathcal{C}$)
3  if $dist(\vec{v}_{new}, \vec{m}_{c_{near}}) \geq \theta$ then
4      Initialize a new cluster and assign $\vec{v}_{new}$ to the new cluster $c_{new}$
5      $\mathcal{C}_{new} \leftarrow \mathcal{C} \cup c_{new}$
6  else
7      $c_{new} \leftarrow c_{near}$
8      Assign vector $\vec{v}_{new}$ to $c_{new}$
9      Update($\vec{m}_{c_{new}}$)
10     for each $\vec{v}_i$ in $c_{new}$ do
11         if $dist(\vec{v}_i, \vec{m}_{c_{new}}) > r_c^1$ then
12             $moveOut(\vec{v}_i, B, c_{new})$
13     for each vector $\vec{v}'$ near to cluster $c$ do
14         if $dist(\vec{v}', \vec{m}_{c_{new}}) <= r_c^1$ then
15             $moveIn(\vec{v}', B, c_{new})$
16     $\delta \leftarrow$Determine the boundary threshold $r_c^1$-$r_c^2$
17     if $dist(\vec{m}_{c_{near}}, \vec{m}_{c_{new}}) > \delta$ then
18         Recalculate the properties of the cluster $c_{new}$
19 $\mathcal{C}_{new} \leftarrow c_{new} \cup (\mathcal{C} \setminus c_{near})$
20 return $\mathcal{C}_{new}$
```

Algorithm 1 presents the maintenance steps for the proposed framework using the improved maintenance approach. In lines 1–2, we do the preparation for the maintenance. Lines 3–5 check the distance between the XML vectors and the centroid. If the distance is bigger than the distance threshold θ, we initialize a new cluster. In lines 7–9, we assign the XML vectors to the nearest cluster c_{near}. In lines 10–15, we track the XML vectors in the boundary space by using a set of maintenance operations. Lines 16–18 verify the distance between the old and new cluster centroids. If the distance is bigger than the boundary threshold δ, we do the local recalculation for the cluster.

7 Experimental Results

This section highlights the experimental results for the XClusterMaint framework. The comparison analysis between the clustering models is discussed in

Sect. 7.1. For the effectiveness test, we prove the correctness of the incremental maintenance approaches based on the cluster radius. Therefore, this paper addresses the efficiency of the clusters maintenance which is presented in Sect. 7.2. The experiments are implemented with Visual Basic 2012 and executed by a processor Intel, Core (i5)-3570 CPU 3.40 GHz. We conducted experiments on the real datasets. Specifically, we use four datasets that include 16,000 documents. These documents contain the information of the scheduled flights [9]. The datasets have the following sizes: 400 MB, 980 MB, 920 MB, and 640 MB.

7.1 Comparison with Fractal Clustering Model

We quantify the effects of the proposed clustering model according to the execution time in comparison with the fractal clustering model which is discussed in the literature 2. The execution time is the actual processing time that covers the entire processes for the clustering. It is an essential indicator to verify the efficiency of the proposed framework. The results show that our clustering model requires less execution time for clustering 4000 XML documents in comparison with fractal clustering model as shown in Fig. 3. Technically, both clustering models use tf.idf formula to represent the structure and content of XML documents as vectors. However, the main reason behind this difference is that the fractal clustering model needs further execution time to capture the similar vectors by computing the offset and scale factors of fractal similarity to determine the similar vectors.

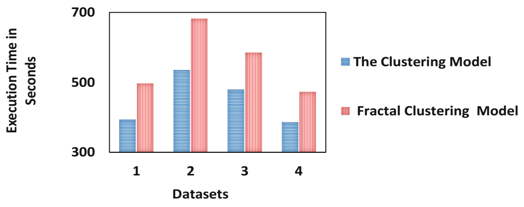
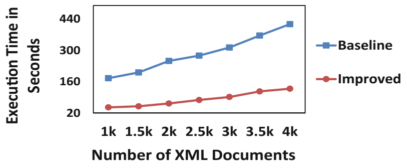


Fig. 3. Comparison of the execution time for the XML clustering models

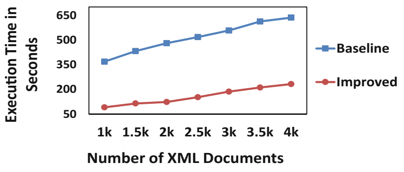
7.2 Efficiency of the Clusters Maintenance

Figure 4 shows the execution time of our proposed approaches by varying the number of inserted XML documents in the experimented datasets. We note that both datasets 2 and 3 require a long execution time in comparison with datasets 1 and 4. When the number of XML documents is set from 1000 to 4000 with an increase of 500 documents at each iteration. We set the radius $r_c^1 = 0.682$ and $r_c^2 = 0.547$ respectively. The value of the first cluster radius r_c^1 is predefined depending on the maximum distance between the centroid $\vec{m}_c$ and the contained XML vectors in each cluster. While the value of the second cluster radius

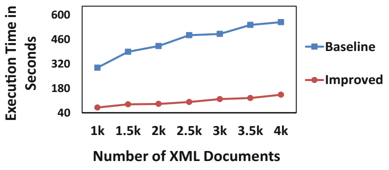
r_c^2 is determined depending on the average of the minimum and maximum distances. Based on the Euclidean distance, the maximum distance is 0.682 and the minimum distance is 0.413. We note that the improved maintenance approach requires a less execution time for maintaining the clusters in comparison with the baseline maintenance approach.



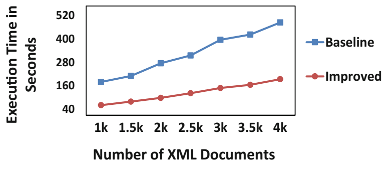
(a) Dataset 1



(b) Dataset 2



(c) Dataset 3



(d) Dataset 4

Fig. 4. Execution time for the baseline and improved maintenance approaches

The main reason is that the lazy maintenance scheme tracks only the XML vectors inside the boundary space of the clusters. While the baseline approach depends on the eager maintenance scheme which tracks the XML vectors inside the stable space and the boundary space of the clusters. We also examine the effect of the varying r_c^1 and r_c^2 on the efficiency of the cluster maintenance as shown in Fig. 5. The first and second experimented datasets are used in the test of varying the cluster radius.

Figures 5a and c present the effect of varying r_c^1 on the maintenance performance in two datasets. We set the value of the first radius and the second radius as $r_c^1 = 0.506$, 0.612, 0.718 and 0.824; and $r_c^2 = 0.412$, 0.515, 0.618 and 0.721 respectively. We note that when the value of r_c^1 increases, the execution time decreases for both maintenance approaches on both datasets. That's because when the value of the r_c^1 gets bigger, the number of generated clusters is smaller. Therefore, most XML vectors are assigned to the small set of the existing clusters without having to initialize the new clusters. Figures 5b and d present the effect of varying of the second radius r_c^2 on the maintenance performance. We fix $r_c^1 = 0.824$, and vary $r_c^2 = 0.412$, 0.515, 0.618 and 0.721. Clearly, varying r_c^2 has almost no effect on the performance of the baseline maintenance approach. That's because the baseline approach works on all the vectors in the stable and boundary spaces for maintenance. In the improved approach; however, by varying r_c^2 to a smaller value, the number of vectors in the stable space decreases which leads to bigger execution time. Conversely, by setting r_c^2 to a bigger value,

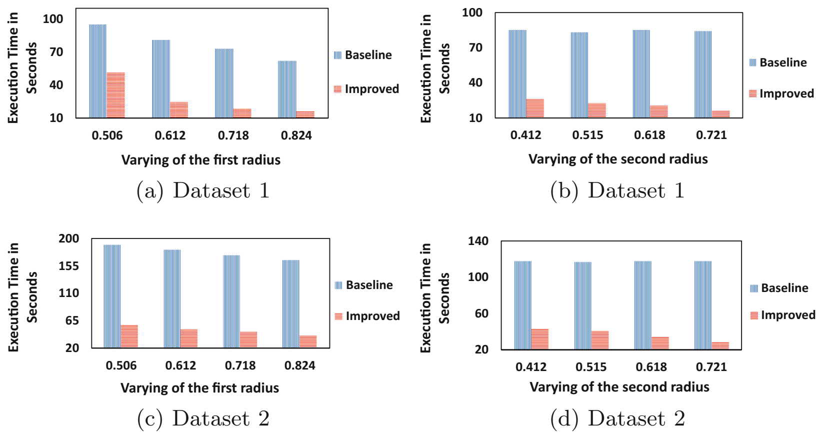


Fig. 5. Execution time for the baseline and improved maintenance approaches with varying the values of r_c^1 and r_c^2

the number of vectors inside the stable space increases. As a result, only small number of vectors in the boundary space are checked for the maintenance which leads to improving the performance on both datasets.

8 Conclusion

In this paper, we introduce a novel framework called XClusterMaint that serves both the clustering and the clusters maintenance of XML documents. For clustering, we generate a set of initial clusters for the XML documents based on the combination of structure and content similarity. For maintenance, we maintain the properties of the existing clusters dynamically by using two incremental approaches for the clusters maintenance: (1) Baseline approach, and (2) Improved approach. In the first approach, we use the Eager Maintenance scheme of the Cluster (EMC) which tracks the XML vectors that reside in the stable and boundary spaces by using two sets of maintenance operations. In the second approach, we use the Lazy Maintenance scheme of the Cluster (LMC) to improve the performance of the cluster maintenance. The LMC scheme tracks only the XML vectors that reside in the boundary space by using a set of maintenance operations. Our experiments verify that the proposed LMC scheme is more efficient than EMC scheme. The resultant development for the clustering and maintenance of XML documents would be capable of improving the performance of real world applications by reducing the required response time. For future work, we will consider further extensions for the XML documents clustering as well as the clusters maintenance to be more effective in the dynamic environment.

Acknowledgements. This work was partially supported by the ARC Discovery Project under Grant No. DP170104747 and the Iraqi Ministry of Higher Education and Scientific Research.

References

1. Abbas, A.M., Bakar, A.A., Ahmad, M.Z.: Fast dynamic clustering SOAP messages based compression and aggregation model for enhanced performance of web services. J. Netw. Comput. Appl. **41**, 80–88 (2014)
2. Al-Shammary, D., Khalil, I.: Dynamic fractal clustering technique for SOAP web messages. In: IEEE International Conference on Services Computing (SCC), pp. 96–103 (2011)
3. Cha, S.H.: Comprehensive survey on distance/similarity measures between probability density functions. Int. J. Math. Models Methods Appl. Sci. **1**(2), 1 (2007)
4. Cheng, W., Zhang, X., Pan, F., Wang, W.: HICC: an entropy splitting-based framework for hierarchical co-clustering. Knowl. Inf. Syst. **46**(2), 343–367 (2016)
5. Cochez, M., Mou, H.: Twister tries: approximate hierarchical agglomerative clustering for average distance in linear time. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp. 505–517 (2015)
6. Costa, G., Manco, G., Ortale, R., Ritacco, E.: Hierarchical clustering of XML documents focused on structural components. Data Knowl. Eng. **84**, 26–46 (2013)
7. Ding, R., Wang, Q., Dang, Y., Fu, Q., Zhang, H., Zhang, D.: Yading: fast clustering of large-scale time series data. Proc. VLDB Endow. **8**(5), 473–484 (2015)
8. Salton, G., McGill, M.J.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
9. OpenFlights, 15 December 2016. https://datahub.io/dataset/open-flights
10. Phan, K.A., Tari, Z., Bertok, P.: Similarity-based soap multicast protocol to reduce bandwidth and latency in web services. IEEE Trans. Serv. Comput. **1**(2), 88–103 (2008)
11. Silva, J.A., Faria, E.R., Barros, R.C., Hruschka, E.R., de Carvalho, A.C., Gama, J.: Data stream clustering: a survey. ACM Comput. Surv. (CSUR) **46**(1), 13 (2013)
12. Tran, T., Nayak, R., Bruza, P.: Combining structure and content similarities for XML document clustering. In: Proceedings of the 7th Australasian Data Mining Conference, vol. 87, pp. 219–225 (2008)
13. Wang, D., Li, T.: Document update summarization using incremental hierarchical clustering. In Proceedings of the 19th ACM International Conference on Information and Knowledge Management, pp. 279–288 (2010)
14. Yan, J., Cheng, D., Zong, M., Deng, Z.: Improved spectral clustering algorithm based on similarity measure. In: International Conference on Advanced Data Mining and Applications, pp. 641–654 (2014)
15. Yongming, G., Dehua, C., Jiajin, L.: Clustering XML documents by combining content and structure. In: International Symposium on Information Science and Engineering, ISISE 2008, vol. 1, pp. 583–587 (2008)