Collaborative Clustering of XML Documents

Sergio Greco, Francesco Gullo, Giovanni Ponti, Andrea Tagarelli

Dept. of Electronics, Computer and Systems Sciences (DEIS), University of Calabria
Via P. Bucci, 41C, 87036 Arcavacata di Rende (CS), Italy
E-mail addresses: {greco, fgullo, gponti, tagarelli}@deis.unical.it

***Abstract*—This paper presents a distributed collaborative approach to XML document clustering. According to a previous study [1], XML documents are mapped to a transactional domain, based on a data representation model which exploits the notion of XML tree tuple. This XML transactional model is well-suited to the identification of semantically cohesive substructures from XML documents, according to structure as well as content information. The proposed clustering framework employs a centroid-based partitional clustering paradigm in a distributed environment. Each peer in the network is allowed to compute a local clustering solution over its own data, then exchanges cluster centroids with other peers. The exchanged centroids correspond to recommendations offered by a peer to peers allowed to compute global representatives. Exploiting these recommendations, each peer becomes responsible for computing a global set of centroids for a given set of clusters. The overall clustering solution is hence computed in a collaborative way according to data from all the peers. Our approach has been evaluated on real XML document collections varying the number of peers. Results have shown that collaborative clustering leads to accurate overall clustering solutions with a relatively low load in the network.**

***Keywords*-XML, collaborative distributed clustering, XML structure and content information, transactional data.**

I. Introduction

The increasing availability of heterogeneous XML informative sources has raised a number of issues concerning how to represent and manage semistructured data. Indeed, XML allows the definition of semantic markup, that is, customized tags describing the data enclosed by them. As a consequence, the variety of application scenarios within XML is used makes XML information sources exhibit not only different structures and contents but also different ways to semantically annotate the data. However, most of the available information from such XML sources is of syntactic kind, consequently a basic assumption is that if certain syntactic properties are satisfied then semantic relationships in XML data can be discovered. Moreover, most XML documents available are often schema-less, that is no information about the document type definition is provided, making inferring semantics even more difficult.

In this context, a challenge is inferring semantics from XML documents according to the available syntactic information, namely *structure* and *content* features. This has several interesting application domains, such as integration of data sources and query processing, document retrieval, and Web intelligence, which can be seamlessly generalized to any kind of semistructured data.

The clustering problem finds in text databases a fruitful research area. The motivation behind any clustering problem is to discover an inherent structure of relationships in the data, and expose this structure as a set of clusters, where the objects within the same cluster are each other highly similar but very dissimilar from objects in different clusters. Since today semistructured text data has become more prevalent on the Web, and XML is the de-facto standard for such data, *clustering XML documents* has increasingly attracted great attention. Any application domain that needs an organization of complex document structures (e.g., hierarchical structures with unbounded nesting, object-oriented hierarchies) as well as data containing a few structured fields together with some largely unstructured text components can be profitably assisted by an XML document clustering task. Clearly, clustering XML documents by facing both structure and content information turns out to be more difficult than the structure-only case. Indeed, mining XML content inherits some problems faced in traditional knowledge discovery in text (e.g., semantic ambiguity), while new ones arise when content is, as usual, contextually dependent on the logical XML structure.

Another problem with managing collections of XML documents is that often the size of such data is huge and inherently distributed, therefore classical centralized approaches may be not efficient. On the other hand, transferring all data to a central clustering service is prohibitive in large-scale systems.

Unfortunately, existing methods and systems for clustering XML data are designed to work only on a centralized environment. This partly depends on an inherent difficulty in devising suitable representation models for taking into account both structure and content information in such data. Moreover, most clustering strategies cannot easily be distributed, since there is an additional level of complexity due to the design and implementation of scalable and effective protocols for communication which allow nodes to minimize exchanged data.

Contribution

Our proposal is focused on the development of a distributed framework for efficiently clustering XML documents. The distributed environment consists of a peer-to-peer network where each node in the network has access to a portion of the whole document collection and communicates with all the other nodes to perform a clustering task in a collaborative fashion.

The proposed framework is essentially based on an approach to modeling and clustering XML documents by structure and content presented in our earlier works [1], [2]. Following the lead of these works, XML documents are transformed into *transactional data* based on the notion of *tree tuple*. XML tree tuples enable a flat, relational-like XML representation that is well-suited to meet the requirements for clustering XML documents according to structure and content information.

We resort to the well-known paradigm of *centroid-based partitional clustering* [3] to conceive our distributed, transactional clustering algorithm. It should be emphasized that such a clustering paradigm is particularly appealing to a distributed environment. Indeed, the availability of a summarizing description of the clustered data provided by the cluster centroids is highly desirable especially when the input data is spread across different peers. Cluster centroids are hence used to describe portions of the document collection and can conveniently be exchanged with other nodes on the network.

The key idea underlying the collaborative clustering approach in our framework is intuitively described as follows. Each node yields a local clustering solution (i.e., a partition of its own set of XML data). For each local cluster, the corresponding (local) centroid is computed and sent to nodes which are in charge of computing the "global" centroids. More precisely, every node computes a subset of the k global centroids; the i-th node computing the global centroid for a set of clusters, receives from each other node the centroid for the corresponding local cluster. Once computed, the global centroids are finally sent back to all the nodes to update their local clusters.

We conducted experiments on two large, real-world collections of XML documents, which are particularly suitable to assess the ability of the proposed framework in performing collaborative clustering of XML documents by structure and content. The number of nodes collaborating in the distributed environment for the computation of clusters was varied and the communication among nodes was kept minimized. Results have shown that, although the final clustering accuracy is typically reduced w.r.t. the centralized case, the parallelism due to a relatively small number of collaborating nodes in the network leads to a drastic reduction of the overall runtime needed for the clustering task.

Related work

A major issue in XML document clustering is the definition of a representation model which is well-suited to handle both structure and content information in XML data. Representing semistructured data has been traditionally addressed by labeled rooted trees. Consequently, handling with such data has leveraged results from research on tree matching, including a number of algorithms for computing tree edit distances (e.g., [4]). Since the complexity issues relating to edit distances, summarization models have been also proposed to concisely represent XML data while preserving some structural relationships between XML elements (e.g., [5], [6], [7]).

Recently attention has been drawn toward using simple Vector-space models to represent XML data, which substantially differ in the definition of feature space (e.g., [8], [9], [10]). In [9], an XML document is projected into a vectorial space whose features take into account the frequency of structure within the documents. Each feature is characterized by a number of properties relating to subpaths, such as the path length, the root node label, and the number of path nodes. In [10], XML documents are transformed into sets of attribute-values by focusing on different tree relationships among the nodes within the document trees (e.g., parent-child and next-sibling relations, set of distinct paths). In [8], the k-Means algorithm is used and two feature sets are generated from the XML element texts and labels, respectively.

In our earlier works [1], [2], we originally introduced an XML representation model that allows for mapping XML document trees into *transactional data*. In a generic application domain, a transaction data set is a multi-set of variable-length sequences of objects with categorical attributes; in the XML domain, we devise a transaction as a set of items, each of which embeds a distinct combination of structure and content features from the original XML data. Within this view, XML documents are not directly transformed to transactional data, rather they are initially decomposed on the basis of the notion of *tree tuple*. Intuitively, given any XML document, a tree tuple is a tree representation of a complete set of distinct concepts that are correlated according to the structure semantics of the original document tree. Tree tuples extracted from the same tree maintain similar or identical structure while reflect different ways of associating content with structure as they can be naturally inferred from the original tree.

Traditional clustering techniques assume data is memory-resident. However, this assumption does not hold in many large scale systems. In this respect, the development of clustering methods in parallel and distributed environment is becoming important. Indeed, distributed computing plays a key role since clustering and, in general, data mining tasks often require huge amounts of resources in storage space and computation time. Moreover, data is often inherently distributed into several databases, making a centralized analysis of such data inefficient and prone to security risks.

One of the earliest studies on distributed data mining is proposed in [11], where an agent-based architecture is defined in such a way that each agent has a local model of the world and agents cooperate to improve solutions. The problem of document clustering in a distributed peer-to-peer network has been addressed recently. For instance, in [12], the significance of centroid-based partitional clustering like k-Means is leveraged as an efficient approach to distributed clustering of documents. In [13], a collaborative approach to distributed clustering of document clustering is presented. In this collaborative approach, the individual local clustering solutions are improved exploiting the distributed environment on the basis of recommendations exchanged by the various peers. Since the data domain is textual, the document cluster summaries are modeled in form of keyphrases.

The collaborative approach to distributed clustering proposed in [13] is very close to ours. However, to the best of our knowledge, the method presented in this work addresses for the first time the problem of clustering XML documents by structure and content in a distributed network.

II. XML Transactional Representation

A. Preliminaries on XML trees and paths

A *tree* T is a tuple $T = \langle r_T, N_T, E_T, \lambda_T \rangle$, where $N_T \subseteq \mathbb{N}$ denotes the set of nodes, $r_T \in N_T$ is the distinguished root of T, $E_T \subseteq N_T \times N_T$ denotes the (acyclic) set of edges, and $\lambda_T : N_T \mapsto \Sigma$ is a function associating a node with a label in the alphabet Σ. Let Tag, Att, and Str be alphabets of tag names, attribute names, and strings respectively. An *XML tree* XT is a pair $XT = \langle T, \delta \rangle$, such that: *i)* T is a tree defined on the alphabet $\Sigma = Tag \cup Att \cup \{\mathtt{S}\}$, where symbol $\mathtt{S} \notin Tag \cup Att$ is used to denote the `#PCDATA` content model; *ii)* given $n \in N_T$, $\lambda_T(n) \in Att \cup \{\mathtt{S}\} \Leftrightarrow n \in Leaves(T)$; *iii)* $\delta : Leaves(T) \mapsto Str$ is a function associating a string to a leaf node of T.

An *XML path* p is a sequence $p = s_1.s_2.\ldots.s_m$ of symbols in $Tag \cup Att \cup \{\mathtt{S}\}$. Symbol s_1 denotes the tag name of the document root element. An XML path can be categorized into two types: *tag path*, if $s_m \in Tag$, or *complete path*, if $s_m \in Att \cup \{\mathtt{S}\}$. We denote as $\mathcal{P}_{XT}$ the set of complete paths in XT. The length of the longest path in $\mathcal{P}_{XT}$ determines the depth of XT, denoted as *depth*(XT).

Let $XT = \langle T, \delta \rangle$ be an XML tree, and $p = s_1.s_2.\ldots.s_m$ be an XML path. The application of p to XT identifies a set of nodes $p(XT) = \{n_1, \ldots, n_h\}$ such that, for each $i \in [1..h]$, there exists a sequence of nodes, or *node path*, $np_i^p = [n_{i_1}, \ldots, n_{i_m}]$ with the following properties:

- $n_{i_1} = r_T$ and $n_{i_m} = n_i$;
- $n_{i_{j+1}}$ is a child of n_{i_j}, for each $j \in [1..m\text{-}1]$;
- $\lambda(n_{i_j}) = s_j$, for each $j \in [1..m]$.

Moreover, we say that the application of a path to an XML tree yields an *answer*. Given an XML tree XT and a path p, the answer of p on XT is defined as either $\mathcal{A}_{XT}(p) \equiv p(XT)$ (i.e., the set of node identifiers $p(XT)$) if p is a tag path, or $\mathcal{A}_{XT}(p) = \{\delta_T(n) \mid n \in p(XT)\}$ (i.e. the set of string values associated to the leaf nodes identified by p) if p is a complete path.

B. XML tree tuples

Tree tuple resembles the notion of tuple in relational databases and has been proposed to extend functional dependencies to the XML setting [14], [15]. In a relational database, a tuple is a function assigning each attribute with a value from the corresponding domain.

Definition 1 ([1], [2]): Given an XML tree XT, an XML tree tuple τ *derived from XT is a maximal subtree of XT such that, for each (tag or complete) path p in XT, the size of the answer of p on τ is not greater than 1, i.e., $|\mathcal{A}_\tau(p)| \leq 1$.* □

We denote with $\mathcal{T}_{XT}$ and $\mathcal{T}$ the set of tree tuples that can be derived from any given tree XT and from the collection $\mathcal{XT}$, respectively. Also, we use $\mathcal{P}_\tau$ to denote the set of complete paths in a tree tuple τ.

C. A transactional model for XML tree tuples

Given a set $\mathcal{I} = \{e_1, \ldots, e_m\}$ of distinct categorical values, or *items*, a transactional database is a multi-set of transactions $tr \subseteq \mathcal{I}$. In our setting, the item domain is built over all the leaf elements in a given collection of XML tree tuples, that is the set of distinct answers of complete paths applied to the tree tuples. A transaction is then modeled with the set of items associated to the leaf elements of a specific tree tuple. The intuition behind such a model lies mainly on the definition of XML tree tuple itself: each path applied to a tree tuple yields a unique answer, thus each item in a transaction indicates information on a concept that is distinct from that of other items in the same transaction.

Definition 2 ([1], [2]): Given an XML tree tuple τ and a path $p \in \mathcal{P}_\tau$, an XML tree tuple item *in τ is a pair $\langle p, \mathcal{A}_\tau(p) \rangle$. The* XML transaction *corresponding to τ is the set of XML tree tuple items of τ, which is $\mathcal{I}_\tau = \{\langle p, \mathcal{A}_\tau(p) \rangle \mid p \in \mathcal{P}_\tau\}$. Given a collection $\mathcal{XT}$ of XML trees, the* XML transaction set *$\mathcal{S}$ for $\mathcal{XT}$ is defined as $\mathcal{S} = \bigcup_{XT \in \mathcal{XT}} \mathcal{S}_{XT}$, where $\mathcal{S}_{XT} = \{\mathcal{I}_\tau \mid \tau \in \mathcal{T}_{XT}\}$.* □

III. XML Transactional Clustering

In this section, we describe how XML tree tuples modeled as transactions can be compared each other and clustered by applying a centroid-based partitional algorithm suitably designed for a collaborative environment.

A. XML tree tuple item similarity

As discussed in the previous section, XML features are represented by tree tuple items. To compare XML data in our transactional domain, we define a measure of similarity between tree tuple items according to their structure and content features.

Definition 3 ([1], [2]): Let e_i and e_j be two tree tuple items. The *tree tuple item similarity* function is defined as

$$sim(e_i, e_j) = f \times sim_S(e_i, e_j) + (1 - f) \times sim_C(e_i, e_j),$$

where sim_S (resp. sim_C) denotes the structural (resp. content) similarity between the items, and $f \in [0..1]$ is a factor that tunes the influence of the structural part to the overall similarity. □

Since the combination of structure and content information characterizes an XML tree tuple item, it is advisable to take tolerance on computing similarity between XML tree tuple items. For this purpose, we introduce a similarity threshold that represents the minimum similarity value for considering two XML tree tuple items as similar.

*Definition 4 ([1], [2]): Given a real value $\gamma \in [0..1]$, two XML tree tuple items e_i and e_j are said to be γ-*matched *if $sim(e_i, e_j) \geq \gamma$.* □

Similarity by Structure: Structural similarity between two tree tuple items e_i and e_j is evaluated by comparing their respective tag paths.

Computing the similarity between any two paths is essentially accomplished by referring to it as a simple case of string matching of their respective element names, and finally averaging the (weighted) matchings. Given any two tags t and t', the Dirichlet function (δ) is applied in such a way that

$\delta(t,t')$ is equal to one if the tags match, otherwise $\delta(t,t')$ is equal to zero.

Definition 5: Let e_i and e_j be XML tree tuple items, $p_i = t_{i_1}.t_{i_2}. \ldots .t_{i_n}$ and $p_j = t_{j_1}.t_{j_2}. \ldots .t_{j_m}$ be their respective tag paths. The structural similarity *between e_i and e_j is defined as*

$$sim_S(e_i,e_j) = \frac{1}{n+m}\left(\sum_{t\in p_i} sim(t,p_j) + \sum_{t\in p_j} sim(t,p_i)\right)$$

such that, for each $t_{i_h} \in p_i$

$$sim(t_{i_h},p_j) = avg_{\,t_{j_k}\in p_j}\left\{\frac{1}{1+|h-k|}\times\delta(t_{i_h},t_{j_k})\right\}$$

□

It should be noted that the tag matchings are corrected by a factor which is inversely proportional to the absolute difference of location of the tags in their respective paths: this avoids that two paths having the same but differently located tags are identified as highly similar to each other.

Similarity by Content: Content features are generated from the texts associated to XML tree tuple items. We refer to a *textual content unit* (for short, TCU) as the preprocessed text of a tree tuple item, i.e., a `#PCDATA` element content or an attribute value. Text preprocessing is accomplished by means of language-specific operations such as lexical analysis, removal of stopwords and word stemming [16].

Two statistical criteria are typically considered for measuring syntactic relevance of terms, namely term density in a given text and term rarity in the text collection. The popular $tf.idf$ (term frequency - inverse document frequency) weighting function [16] takes both criteria into account. However, our XML transactional domain requires a more refined and structured modeling of term relevance, which is able to consider the term occurrences with respect to a context that includes TCUs, tree tuples and original document trees suitably.

*Definition 6 ([1], [2]): Given a collection of XML trees $\mathcal{XT}$, let w_j be an index term in a TCU u_i, which belongs to a tree tuple $\tau \in \mathcal{T}$ extracted from a tree $XT \in \mathcal{XT}$. The $ttf.itf$ (*Tree tuple Term Frequency - Inverse Tree tuple Frequency*) weight of w_j in u_i with respect to τ is defined as*

$$ttf.itf(w_j,u_i|_\tau) = tf(w_j,u_i)\times\exp\left(\frac{n_{j,\tau}}{N_\tau}\right)\times\frac{n_{j,XT}}{N_{XT}}\times\ln\left(\frac{N_{\mathcal{T}}}{n_{j,\mathcal{T}}}\right)$$

where:

- *$tf(w_j,u_i)$ is the number of occurrences of w_j in u_i,*
- *$n_{j,\tau}$ is the number of TCUs in τ that contain w_j,*
- *N_τ is the number of TCUs in τ,*
- *$n_{j,XT}$ is the number of TCUs in XT that contain w_j,*
- *N_{XT} is the number of TCUs in XT,*
- *$n_{j,\mathcal{T}}$ is the number of TCUs in $\mathcal{T}$ that contain w_j,*
- *$N_{\mathcal{T}}$ is the number of TCUs in $\mathcal{T}$.* □

Using the $ttf.itf$ weighting function, the relevance of a term increases with the term frequency within the local TCU, with the term popularity across the TCUs of the local tree tuple (transaction) and the TCUs of the local document tree, and with the term rarity across the whole collection of TCUs.

Content similarity between any two tree tuple items is measured by comparing their respective TCUs. Given a collection of XML tree tuples $\mathcal{T}$, any TCU u_i is modeled with a vector $\vec{u}_i$ whose j-th component corresponds to an index term w_j and contains the $ttf.itf$ relevance weight. The size of each TCU vector is equal to the size of the collection vocabulary, i.e., the set of index terms extracted from all TCUs in $\mathcal{T}$. The well-known *cosine similarity* [17] is used to measure the similarity between TCU vectors.

Definition 7: Let e_i and e_j be tree tuple items, and $\vec{u}_i$ and $\vec{u}_j$ their respective TCU vectors. The content similarity *between e_i and e_j is defined as*

$$sim_C(e_i,e_j) = \frac{\vec{u}_i\cdot\vec{u}_j}{\|\vec{u}_i\|\times\|\vec{u}_j\|}$$

□

B. The CXK-means clustering algorithm

XML tree tuples modeled as transactions can be efficiently clustered by applying a partitional algorithm devised for the XML transactional domain.

Generally, given a set of data objects and a positive number k, a partitional clustering algorithm identifies k non-empty, disjoint groups each containing a homogeneous subset of objects. An important class of partitional approaches is based on the notion of *centroid*, or *representative*, of cluster: each object is assigned to a cluster C according to its distance from a data point c, which is the centroid of C.

In [1], [2], we developed a centroid-based partitional clustering algorithm, which is essentially a variant of the K-means algorithm for the XML transactional domain. From clustering strategy viewpoint, this algorithm works as a traditional centroid-based method to compute $k+1$ clusters: starts choosing k objects as the initial cluster centroids, then iteratively reassigns each remaining object to the closest cluster until all cluster centroids do not change. The $(k+1)$-th cluster, called *trash cluster*, is created to contain unclustered objects, i.e. objects having an empty intersection with each cluster centroid and so are not assigned to any of the first k clusters.

Two major aspects in the XML transactional clustering algorithm are (i) the notion of proximity used to compare XML transactions and (ii) the notion of cluster centroid.

In generic transactional domains, a widely used proximity measure is the Jaccard coefficient, which determines the degree of matching between any two transactions as directly proportional to their intersection (i.e., number of common items) and inversely proportional to their union. However, computing exact intersection between XML transactions is not effective, since XML tree tuple items may share structural or content information to a certain degree even though they are not identical. For this purpose, the notion of standard intersection between sets of items is enhanced with one able to capture even minimal similarities from content and structure features of XML elements.

Definition 8 ([1], [2]): Let tr_1 and tr_2 be two transactions, and $\gamma \in [0..1]$ be a similarity threshold. The set of *γ-shared items* between tr_1 and tr_2 is defined as

$$match^\gamma(tr_1,tr_2) = match^\gamma(tr_1 \rightarrow tr_2) \cup match^\gamma(tr_2 \rightarrow tr_1),$$

```
Global Input:
  A set $\mathcal{S}$ of XML transactions distributed over $m$ nodes;
  The desired number $k$ of clusters;  A similarity threshold $\gamma$.
Global Output:
  A partition $\mathcal{C}$ of $\mathcal{S}$ in $k$ clusters distributed over $m$ nodes;
---------------------------------------------------------------------------
Process $N_0$
Method:
  define a partition of $\{1..k\}$ into $m$ subsets $Z_1, \ldots, Z_m$;
  for $i = 1$ to $m$ do
    send $(\{Z_1, \ldots, Z_m\}, k, \gamma)$ to $N_i$;
---------------------------------------------------------------------------
Process $N_i$
Input:
  A set $\mathcal{S}^i = \{tr^i_1, \ldots, tr^i_{n_i}\}$ of XML transactions;
Output:
  A partition $\mathcal{C}^i = \{C^i_1, \ldots, C^i_k\}$ of $\mathcal{S}^i$ into $k$ clusters.
Method:
  receive $(\{Z_1, \ldots, Z_m\}, k, \gamma)$ from $N_0$;
  let $Z_i = \{i_1, \ldots, i_{q_i}\}$, with $0 \leq q_i \leq k$;
  /* selects $q_i$ initial global clusters */
  select $\{c_{i_1}, \ldots, c_{i_{q_i}}\}$ transactions coming from distinct original trees;
  $C^i_j = \{\}, \forall j \in [1..k]$;
  repeat
    send (broadcast) $\{c_{i_1}, \ldots, c_{i_{q_i}}\}$ to $N_1, \ldots, N_m$;
    receive $c_j$ from $N_h$ with $h \in [1..m]$ and $j \in Z_h$;
    repeat  /* computes local clusters */
      $C^i_j := \{tr \mid tr \in S^i \wedge sim^\gamma_J(tr, c^i_j) > sim^\gamma_J(tr, c^i_l),\ l \in [1..k]\}$,
               $\forall j \in [1..k]$;
      $C^i_{k+1} := \{tr \mid sim^\gamma_J(tr, c^i_j) = 0\}, \forall j \in [1..k]$;
      $c^i_j :=$ computeLocalRepresentative$(C^i_j), \forall j \in [1..k]$;
    until $\mathcal{Q}(\mathcal{C}^i)$ is maximized;
    if $c^i_j$ does not change, $\forall j \in [1..k]$ then
      send (broadcast) $([], done)$;
    else
      send $(\{\langle c^i_j, |C^i_j|\rangle | j \in Z_h\}, continue)$ to $N_h, \forall h \in [1..m]$;
    receive $(\{c^h_j | j \in Z_h\}, V_h)$ from $N_h, \forall h \in [1..m]$
    if $(\exists h \in [1..m]$ s.t. $V_h = continue)$ then
      for $j \in Z_i$ do  $c_j =$ ComputeGlobalRepresentative$(\{c^1_j, \ldots, c^m_j\})$;
  until $V_1 = \cdots = V_m = done$;
---------------------------------------------------------------------------
Function ComputeLocalRepresentative$(C)$ : $rep$;
  $I_C = \{e \mid e \in tr \wedge tr \in C\}$;
  let $P_C = \{p/h \mid \exists h$ items $(p, u) \in I_C\}$;
  let $\vec{\mu_C} = \text{avg}_{e_i \in I_C}\{\vec{u}_i\}$;
  for each $e \in I_C$ do
    let $rank_S(e) = sum\{h \mid \exists e' = (p', u') \in I_C \wedge p'/h \in P_C \wedge$
                      $sim_S(e, e') \geq \gamma\}/|P_C|$;
    let $rank_C(e) = (\vec{u} \cdot \vec{\mu_C})/(\|\vec{u}\| \times \|\vec{\mu_C}\|)$,
                  where $\vec{u}$ is the $e$'s TCU vector;
    $rank(e) = f \times rank_S(e) + (1 - f) \times rank_C(e)$;
  let $R_C$ be the list containing the element in $I_C$ ordered by $rank$ values;
  return GenerateTreeTuple$(R_C)$;
---------------------------------------------------------------------------
Function ComputeGlobalRepresentative$(C)$ : $rep$;
  $I_C = \{e \mid e \in tr \wedge tr \in C\}$;
  let $P_C = \{p/h \mid \exists h$ items $(p, u) \in I_C\}$;
  let $\vec{\mu_C} = \text{avg}_{e_i \in I_C}\{\vec{u}_i\}$;
  for each $e \in I_C$ do
    let $g\_rank_S(e) = sum\{h \mid \exists e' = (p', u') \in I_C \wedge p'/h \in P_C \wedge$
                      $sim_S(e, e') \geq \gamma\}/|P_C|$;
    let $g\_rank_C(e) = (\vec{u} \cdot \vec{\mu_C})/(\|\vec{u}\| \times \|\vec{\mu_C}\|)$,
                  where $\vec{u}$ is the $e$'s TCU vector;
    $g\_rank(e) = f \times g\_rank_S(e) + (1 - f) \times g\_rank_C(e)$;
  let $\overline{I}_C$ be the list containing the element in $I_C$ ordered by $g\_rank$ values;
  return GenerateTreeTuple$(\overline{I}_C)$;
---------------------------------------------------------------------------
Function GenerateTreeTuple$(I_C)$ : $rep$;
  let $I^*_C \subseteq I_C$ be the set of items in $I_C$ with the highest rank;
  $rep := conflateItems(I^*_C)$;
  $s_0 := \sum_{tr \in C} sim^\gamma(tr, rep)$;        /* refines representative */
  $I_C := I_C - I^*_C$;
  let $|tr_{max}|$ be the maximum length of transaction within $C$;
  while $(I_C \neq \emptyset\ \wedge\ |rep| \leq |tr_{max}|)$ do
    let $I^*_C \subseteq I_C$ be the set of items in $I_C$ with the highest rank;
    $rep' := conflateItems(rep \cup I^*_C)$;
    $s' := \sum_{tr \in C} sim^\gamma(tr, rep')$;
    if $(s' \geq s_0)$ then
      $I_C := I_C - I^*_C$;    $s_0 := s'$;    $rep := rep'$;
    else return $rep$;
  return $rep$;
```

Fig. 1. The *CXK-means* algorithm

where

$$match^\gamma(tr_i \rightarrow tr_j) = \{e \in tr_i \mid \exists e_h \in tr_j,\ sim(e, e_h) \geq \gamma,\ \nexists e' \in tr_i, sim(e', e_h) > sim(e, e_h)\}.$$

□

The set of γ-shared items resembles the intersection between transactions at a degree greater than or equal to a similarity threshold γ. This notion of (enhanced) intersection is also at the basis of the following similarity function.

Definition 9 ([1], [2]): Let tr_1 and tr_2 be two transactions, and $\gamma \in [0..1]$ be a similarity threshold. The *XML transaction similarity* function between tr_1 and tr_2 is defined as

$$sim^\gamma_J(tr_1, tr_2) = \frac{|match^\gamma(tr_1, tr_2)|}{|tr_1 \cup tr_2|}.$$ □

We adapted the XML transactional clustering algorithm to a collaborative distributed environment. Figure 1 sketches the main phases of the *CXK-means* algorithm, which has the following characteristics.

- Data are distributed over m nodes and each node communicates with all the other ones sending "local" representatives and receiving "global" representatives. An initial process corresponding to a node N_0 defines a partition of the k clusters into m subsets $Z_j, j \in [1..m]$. Each partition Z_j contains the identifiers of the clusters for which the node N_j has the responsibility of computing the global representatives.
- Each node N_i is in charge of computing local clusters $C^i_1, \ldots, C^i_k$ and local representatives $c^i_1, \ldots, c^i_k$, but also a subset of the global representatives $c_{i_1}, \ldots, c_{i_{q_i}}$ (using the local representatives computed by all nodes).
- The local representative of a cluster C is computed by starting from the set of γ-shared items among all the transactions within C. More precisely, for each transaction in C, the union of the γ-shared item sets with respect to all the other transactions in C is obtained; this guarantees no dependence of the order of examination of the transactions. Then, a raw representative is computed by selecting the items from these union sets with the highest frequency: the raw representative, however, may not have the form of a tree tuple, as some items therein may refer to the same path but with different answers. Function $conflateItems$ is applied to a set of items and, for each subset $I = \{e_{i1}, \ldots, e_{ik}\}$ of items sharing the same path p, yields one item that has p as path and the concatenation of the contents of items in I as its content. Finally, a greedy heuristic refines the current representative by iteratively adding the remaining most frequent items until the sum of pair-wise similarities between transactions and representative cannot be further maximized. Again, any refinement must guarantee that the resulting representative satisfies Def. 1.
- The global representative of a cluster C is computed by considering the m local representatives $c^1, \ldots, c^m$. The only difference with respect to the computation of the local cluster is that in the computation of the structural rank (here called g_rank) associated with an item e we

consider the rank associated with each item (instead of the number of items) having a γ-matching.

IV. Experimental Evaluation

A. Experimental setting

We assessed the proposed framework in performing clustering according to structure, content, or both information. We hereinafter refer to these kinds of solutions as *structure-driven*, *content-driven*, and *structure/content-driven* clustering, respectively. The first two types of clustering concern the detection of groups of XML data which are homogeneous by either structure or content. The third type (i.e., structure/content-driven clustering) includes a variety of scenarios, ranging from detecting common structures across different topics, or conversely, to identifying classes of tree tuples that both cover common topics and belong to the same structural category.

The three types of clustering correspond to different settings of the parameters f and γ, which control the XML transaction similarity function. According to [1], [2], we varied f within [0..1] with step 0.1, and γ within [0.5..1) with step 0.05—we chose $\gamma = 0.5$ as the maximum tolerance threshold in computing similarities. Also, since the setting of f depends on the clustering goal, we decided to partition the (discrete) interval [0..1] as follows: [0..0.3] for content-driven clustering, [0.4..0.6] for structure/content-driven clustering, and [0.7..1] for structure-driven clustering.

Network topology is characterized by the number of nodes. We performed experiments by varying this parameter from a minimum number of 1 up to 19 nodes, in order to assess the impact of the network size on the clustering task in terms of both effectiveness and efficiency. Clearly, a number of nodes equal to 1 refers to centralized clustering, which represents the baseline case. In the distributed case, data was equally partitioned over the nodes.

B. Data description

We used two real word document collections for the evaluation which are particularly suited to be used in each of the three types of clustering.

The *IEEE* data set refers to the IEEE collection version 2.2, which has been used as a benchmark in the INEX document mining track 2008.[1] *IEEE* consists of 4,874 articles originally published in 23 different IEEE journals from 2002 to 2004. Such articles follow a complex schema which includes front matter, back matter, section headings, text formatting tags and mathematical formulas. We kept most of the logical structure elements and removed the stylistic markups, as shown in the DTD of Figure 2. In our XML transactional domain, the *IEEE* collection has 211,909 transactions and 135,869 items. Also, the number of leaf nodes is 228,869, the maximum fan out is 43, and the average depth is about 5.

In *IEEE*, the article journals determine the categories that were used to partition the collection, which strictly follow the original INEX categorization. Precisely, two structural categories correspond to "Transactions" and "non-Transactions"

```
<!ELEMENT IEEE (article+)>
<!ELEMENT article (front_matter, body, back_matter?)>
<!ELEMENT front_matter (#PCDATA | abstract | author |
          bibliography | editor | editorial | figure |
          headers | keywords)*>
<!ELEMENT body (#PCDATA | ack | author |
          bibliography | figure | index | section |
          table | vita)*>
<!ELEMENT back_matter (#PCDATA | ack | author |
          bibliography | figure | footnote |
          section | table | vita)*>
<!ELEMENT section (#PCDATA | author | ack |
          bibliography | figure | index |
          sub_subsection | subsection | table |
          title | vita)*>
<!ELEMENT subsection (#PCDATA | author | ack |
          bibliography | figure | sub_subsection |
          table | title | vita)*>
<!ELEMENT sub_subsection (#PCDATA | bibliography |
          figure | table | title | vita)*>
<!ELEMENT ack (#PCDATA | figure | title)*>
<!ELEMENT author (#PCDATA | affiliation | email |
          first_name | surname)*>
<!ELEMENT editor (#PCDATA | affiliation | email |
          first_name | surname)*>
<!ELEMENT bibliography (#PCDATA | reference)*>
<!ELEMENT headers (header+)>
<!ELEMENT header (#PCDATA | title)*>
<!ELEMENT index (title, entries)>
<!ELEMENT figure (caption?)>
<!ELEMENT table (title, feet?)>
<!ELEMENT abstract (#PCDATA)>
<!ELEMENT affiliation (#PCDATA)>
 ...
<!ELEMENT vita (#PCDATA)>
```

Fig. 2. DTDs of the *IEEE* dataset

```
<!ELEMENT dblp (article | inproceedings | book |
          incollection)+>
<!ENTITY %field "author | editor | publisher |
         title | booktitle | journal | series">
<!ELEMENT article (%field;)*>
<!ELEMENT inproceedings (%field;)*>
<!ELEMENT proceedings (%field;)*>
<!ELEMENT book (%field;)*>
<!ELEMENT incollection (%field;)*>
<!ELEMENT author (#PCDATA)>
<!ELEMENT editor (#PCDATA)>
...
<!ELEMENT series (#PCDATA)>
```

Fig. 3. DTD of the *DBLP* dataset

articles, respectively, whereas the classification by content organizes the articles by the following 8 topic-classes: "Computer", "Graphics", "Hardware", "Artificial Intelligence", "Internet", "Mobile", "Parallel", and "Security". Moreover, 14 hybrid classes are identified according to these structural and content classes.

The second evaluation data set is a subset of the DBLP archive,[2] a digital bibliography on computer science which contains citations on journal articles, conference papers, books, book chapters, and theses. *DBLP* is comprised of 3,000 documents which correspond to 5,884 transactions and 8,231 items.

DBLP is characterized by a small average depth (3), whereas the number of leaf nodes is 13,209 and the maximum fan out is 20. According to its element type definition (Figure 3), *DBLP* exhibits short text descriptions (e.g., author names, paper titles, conference names), and a moderate structural variety which corresponds to 4 main structural categories, namely 'journal articles' (article), 'conference papers' (inproceedings), 'books' (book), and 'book chapters' (incollection). Also, 6 topical classes are identified in *DBLP*, which are 'multimedia', 'logic programming', 'web

[1] http://www.inex.otago.ac.nz/data/documentcollection.asp

[2] http://dblp.uni-trier.de/xml/

dataset	*# of clusters*	*# of nodes*	*F-measure (avg)*
IEEE	8	1	0.593
		3	0.523
		5	0.485
		7	0.421
		9	0.376
DBLP	6	1	0.764
		3	0.702
		5	0.662
		7	0.612
		9	0.547

TABLE I
CLUSTERING RESULTS WITH $f \in [0..0.3]$ (CONTENT-DRIVEN SIMILARITY)

dataset	*# of clusters*	*# of nodes*	*F-measure (avg)*
IEEE	14	1	0.564
		3	0.497
		5	0.451
		7	0.404
		9	0.356
DBLP	16	1	0.772
		3	0.721
		5	0.676
		7	0.614
		9	0.558

TABLE II
CLUSTERING RESULTS WITH $f \in [0.4..0.6]$ (STRUCTURE/CONTENT-DRIVEN SIMILARITY)

dataset	*# of clusters*	*# of nodes*	*F-measure (avg)*
IEEE	2	1	0.618
		3	0.542
		5	0.497
		7	0.433
		9	0.386
DBLP	4	1	0.988
		3	0.934
		5	0.882
		7	0.819
		9	0.716

TABLE III
CLUSTERING RESULTS WITH $f \in [0.7..1]$ (STRUCTURE-DRIVEN SIMILARITY)

and adaptive systems', 'knowledge based systems', 'software engineering', and 'formal languages'. If both content and structure information are taken into account, 16 classes are identified.

Note that the DTDs shown in Figure 2 and Figure 3 are given only for purposes of description of the data structures, whereas they were not used during the evaluation since our approach does not require the availability of XML schemas.

C. Cluster validity measures

To assess the quality of clustering solutions for the datasets, we exploited the availability of reference classifications for XML documents. The objective was to evaluate how well a clustering fits a predefined scheme of known classes (natural clusters). For this purpose, we resorted to the well-known *F-measure* [18], which is defined as the harmonic mean of values that express two notions from Information Retrieval, namely *Precision* and *Recall*. F-measure ranges within $[0, 1]$, where higher values refer to better quality results. Since we perform tree tuple decomposition of XML documents and then transactional modeling, the evaluation process take into account the set $\mathcal{S}$ of XML transactions.

Given a set $\mathcal{S} = \{tr_1, \ldots, tr_m\}$ of XML transactions, let $\mathit{\Gamma} = \{\Gamma_1, \ldots, \Gamma_H\}$ be the reference classification of the objects in $\mathcal{S}$, and $\mathbf{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_K\}$ be the output partition yielded by a clustering algorithm. *Precision* of cluster $\mathcal{C}_j$ with respect to class Γ_i is the fraction of the objects in $\mathcal{C}_j$ that has been correctly classified, whereas *Recall* of cluster $\mathcal{C}_j$ with respect to class Γ_i is the fraction of the objects in Γ_i that has been correctly classified. Formally,

$$P_{ij} = \frac{|\mathcal{C}_j \cap \Gamma_i|}{|\mathcal{C}_j|}\ ,\ R_{ij} = \frac{|\mathcal{C}_j \cap \Gamma_i|}{|\Gamma_i|}\ ,\ F_{ij} = \frac{2P_{ij}R_{ij}}{(P_{ij} + R_{ij})}$$

In order to score the quality of $\mathcal{C}$ with respect to $\mathit{\Gamma}$ by means of a single value, the overall F-measure $F(\mathbf{C}, \Gamma)$ is computed using the weighted sum of the maximum F_{ij} score for each class Γ_i.

$$F(\mathbf{C}, \Gamma) = \frac{1}{|\mathcal{S}|} \sum_{i=1}^{H} |\Gamma_i| \max_{j \in [1..K]} F_{ij}$$

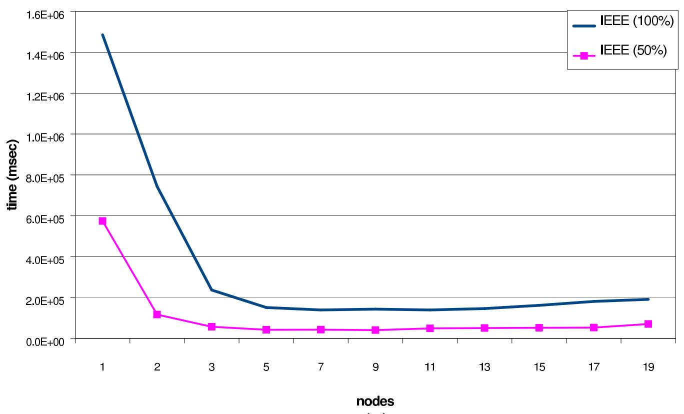
D. Results

Tables I–III show the average clustering performance obtained on the various data sets by *CXK-means* varying the number of nodes and the type of clustering setting (i.e., structure-, content-, and structure/content-driven clustering). For each dataset and clustering setting, results refer to multiple (10) runs of the algorithm and were measured by averaging the F-measure scores over the range of f values specific of the clustering setting. As far as parameter γ, the best setting was found to be close to high values (typically above 0.85), for each dataset and type of clustering [2].

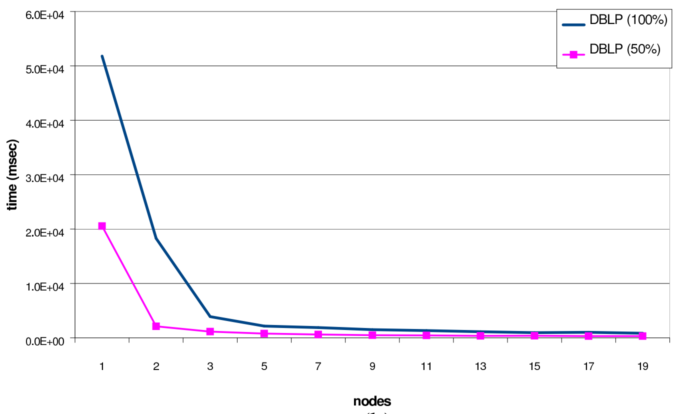
As it is reasonable to expect, the centralized case (i.e., one node) corresponds to an upper bound in terms of clustering quality for the collaborative distributed clustering. While our focus is not on the evaluation of the centralized case—the interested reader can find details in [2], [1]—it can be noted how the clustering accuracy decreases as the number of nodes increases, regardless of the dataset and the type of clustering. This is explained since, the higher the number of nodes, the lower the distribution ratio of the transactions over the nodes; as a consequence, each node produces, at each step of the distributed algorithm, a local clustering solution over a too small portion of data, which cannot really represent the final overall solution. However, such a performance degradation remains relatively acceptable for a distributed environment, which is partly due to our model of cluster centroid in achieving good quality summaries for the clusters.

Figure 4 shows time performances on the two evaluation datasets by increasing the number of nodes and varying the dataset size. Both plots in the figure refer to structure/content-driven clustering experiments (i.e., $f \in [0.4..0.6]$). Here a noteworthy remark is that the performance of *CXK-means* takes major advantages w.r.t. a centralized setting in terms of runtime behavior. In fact, a higher number of nodes in the network leads to more parallelism, which results in a drastic reduction of the overall time needed for the clustering task, in spite of a relatively moderate decrease of the clustering quality.

However, when the number of nodes grows up, the collaborative clustering algorithm also needs a higher number of iterations to converge. This fact affects negatively the network traffic (i.e., the centroid exchange) which might be not negligible anymore. Indeed, as we can see in Figure 4 for both datasets, after a drastic reduction of the runtime due to the use of just a few nodes, the runtime remains roughly constant for a certain range, then it starts to slightly increase when the number of nodes becomes significantly higher. In particular, time performances on *IEEE* tend to stabilize for six and four nodes, respectively in the case of full and halved dataset; on *DBLP*, time performances tend to stabilize for a



(a)



(b)

Fig. 4. Clustering time performances varying the number of nodes and the dataset size: (a) *IEEE*, (b) *DBLP*

smaller number of nodes (four and two, respectively) which is due to a smaller size of *DBLP* w.r.t. *IEEE*, in terms of both transactions and vocabulary of terms.

Another important remark is that, as the dataset size is halved, the minimum number of nodes to bring down the clustering times tends to decrease. This suggests that the advantage of the collaborative distributed approach w.r.t. the centralized one tends to become less significant as the dataset size is reduced.

V. Conclusion and Future Work

We presented a collaborative distributed framework for clustering XML documents; to the best of our knowledge, this is the first collaborative approach to clustering XML documents by structure and content. XML documents are modeled in a transactional domain which is well-suited to extract XML structure as well as content features. We implemented our approach in a distributed, centroid-based partitional clustering algorithm, where cluster centroids are used to describe portions of the document collection and can conveniently be exchanged with other nodes on the network. Each node yields a local clustering solution over its own set of XML data, and exchanges the cluster centroids with other nodes. These recommendations in form of centroids are used to compute global centroids, thus the overall clustering solution is computed in a collaborative way.

We plan to extend our collaborative framework to deal with semantic information of both structural and content type from XML data [2], [1]. Also, more experiments are needed to make the evaluation robust from both effectiveness and scalability viewpoints. In particular, it would be interesting to know on various application domains, how the collaborative clustering behavior varies in function of different data distributions.

References

[1] A. Tagarelli and S. Greco, "Toward Semantic XML Clustering," in *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2006, pp. 188–199.

[2] ——, "Semantic Clustering of XML Documents," *ACM Transactions on Information Systems*, 2009. To appear.

[3] A. Jain and R. Dubes, *Algorithms for Clustering Data*, ser. Prentice-Hall advanced reference series. Prentice-Hall, 1988.

[4] A. Nierman and H. Jagadish, "Evaluating Structural Similarity in XML Documents," in *Proc. ACM SIGMOD Int. Workshop on the Web and Databases (WebDB)*, 2002, pp. 61–66.

[5] W. Lian, D. Cheung, N. Mamoulis, and S. Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," *IEEE Trans. Knowledge Data Eng.*, vol. 16, no. 1, pp. 82–96, 2004.

[6] G. Costa, G. Manco, R. Ortale, and A. Tagarelli, "A Tree-based Approach to Clustering XML Documents by Structure," in *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2004, pp. 137–148.

[7] N. Polyzotis and M. Garofalakis, "Structure and Value Synopses for XML Data Graphs," in *Proc. of the Int. Conf. on Very Large Data Bases (VLDB)*, 2002, pp. 466–477.

[8] A. Doucet and M. Lehtonen, "Unsupervised Classification of Text-Centric XML Document Collections," in *INEX Workshop*, 2006.

[9] A. M. Vercoustre, M. Fegas, S. Gul, and Y. Lechevallier, "A Flexible Structured-based Representation for XML Document Mining," in *INEX Workshop*, 2005, pp. 443–457.

[10] L. Candillier, I. Tellier, and F. Torre, "Transforming XML Trees for Efficient Classification and Clustering," in *INEX Workshop*, 2005, pp. 469–480.

[11] R. Kargupta and I. H. andB. Stafford, "Distributed data mining using an agent based architecture," in *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 1997, pp. 211–214.

[12] M. Eisenhardt, W. Muller, and A. Henrich, "Documents by Distributed P2P Clustering," in *GI Jahrestagung (2)*, 2003, pp. 286–291.

[13] K. Hammouda and M. Kamel, "Collaborative document clustering," in *Proc. SIAM Int. Conf. on Data Mining (SDM)*, 2006, pp. 211–214.

[14] M. Arenas and L. Libkin, "A Normal Form for XML Documents," *ACM Trans. Database Systems*, vol. 29, no. 1, pp. 195–232, 2004.

[15] S. Flesca, F. Furfaro, S. Greco, and E. Zumpano, "Repairs and Consistent Answers for XML Data with Functional Dependencies," in *Proc. Int. XML Database Symposium (XSym)*, 2003, pp. 238–253.

[16] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*, ser. ACM Press Books. Addison Wesley, 1999.

[17] A. Strehl, J. Ghosh, and R. Mooney, "Impact of Similarity Measures on Web-page Clustering," in *Proc. AAAI Workshop on AI for Web Search*, 2000, pp. 58–64.

[18] B. Larsen and C. Aone, "Fast and effective text mining using linear-time document clustering," in *Proc. ACM Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, 1999, pp. 16–22.