Clustering XML Documents: a Distributed Collaborative Approach

(Extended Abstract)

Sergio Greco, Francesco Gullo, Giovanni Ponti,
Andrea Tagarelli, and Giuseppe Agapito

Dept. of Electronics, Computer and Systems Sciences, University of Calabria, Italy
email: {greco,fgullo,gponti,tagarelli,agapito}@deis.unical.it

Abstract. This paper addresses the problem of clustering of XML documents in a distributed collaborative framework. XML documents are first decomposed based on semantically cohesive subtrees, then modeled as transactional data. A distributed centroid-based partitional clustering algorithm is developed to perform in a collaborative fashion. This algorithm is designed to allow each peer in the network to compute a local clustering solution over its own data and to exchange its cluster centroids with other peers. Experimental results are provided on real-world collections of XML documents with a varying number of peers.

1 Introduction

The increasing availability of heterogeneous XML informative sources has raised a number of issues concerning how to represent and manage semistructured data. 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. In this context, a challenge is inferring semantics from XML documents according to the available syntactic information, namely *structure* and *content* features.

The clustering problem finds in text databases a fruitful research area. Since today semistructured text data has become more prevalent on the Web, and XML is the defacto standard for such data, *clustering XML documents* has increasingly attracted great attention. A major issue in XML document clustering is the definition of a representation model that has to be well-suited to handle both structure and content information in XML data. Representing semistructured data has been traditionally addressed by labeled rooted trees. Consequently, dealing with such data has leveraged results from research on tree matching, including a number of algorithms for computing tree edit distances (e.g., [12]). Since the complexity issues relating to edit distances, summarization models have also been proposed to concisely represent XML data while preserving some structural relationships between XML elements (e.g., [11, 4, 13]). In our earlier works [15, 16], we originally introduced an XML representation model that allows for mapping XML document trees into *transactional data*.

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. Traditional clustering techniques assume data is memory-

resident, but this assumption does not hold in many large scale systems. One of the earliest studies on distributed data mining is proposed in [10], where a cooperative agent-based architecture is defined. The problem of document clustering in a distributed peer-to-peer network has been addressed recently. For instance, in [5], the significance of centroid-based partitional clustering like k-Means is leveraged as an efficient approach to distributed clustering of documents. In [8], a collaborative approach to distributed clustering of plain-text documents is presented. The individual local clustering solutions are improved exploiting the distributed environment on the basis of recommendations exchanged by the various peers.

Our proposal is focused on the development of a distributed framework for efficiently clustering XML documents [7]. 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 based on an approach to modeling and clustering XML documents by structure and content [16]. XML documents are treated as transactions of items embedding structure and content information. We resort to the well-known paradigm of *centroid-based partitional clustering* [9] to conceive our distributed, transactional clustering algorithm. 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 that are in charge of computing the "global" centroids. The collaborative clustering strategy of the proposed algorithm is very close to the one proposed in [8]. However, to the best of our knowledge, the method presented in this work addresses for the first time the problem of clustering XML documents in a distributed network taking into account both structure and content information.

We conducted experiments on two large, real-world collections of XML documents. 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.

2 Background on XML Data Modeling

Our approach to modeling XML documents consists of two main steps. The first step is to decompose each document into a number of smaller subtrees that are conceived to be cohesive according to the underlying semantics of the original document. In our approach, this is accomplished by resorting to the notion of *tree tuple*. The second step performs a mapping of the obtained tree tuples to transactional data, where each item represents a combination of XML structure and content information.

Tree tuple resembles the notion of tuple in relational databases, i.e., a function assigning each attribute with a value from the corresponding domain. This notion has also been proposed to extend functional dependencies to the XML setting [1, 6]. Given an XML tree XT, an *XML tree tuple* τ derived from XT is a maximal subtree of XT such that, for each path p in XT, the size of the answer (i.e., the set of nodes resulting from the application of p to τ) is not greater than 1 [16]. 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.

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. 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.

3 Collaborative Clustering of XML Documents in a P2P Network

In this section, we describe how XML tree tuples modeled as transactions can be compared to each other [15, 16] and clustered by applying a collaborative centroid-based partitional algorithm suitably designed for a distributed, P2P environment.

3.1 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.

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. 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. 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.

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

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. Text preprocessing is accomplished by means of language-specific operations such as lexical analysis, removal of stopwords and word stemming [3].

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 $tf.idf$ weighting function [3] 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.

Formally, 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 $\boldsymbol{u}_i$ 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* [14] is used to measure the similarity between TCU vectors.

3.2 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. In [15, 16], we developed a centroid-based partitional clustering algorithm, which is essentially a variant of the k-Means algorithm for the XML transactional domain. 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. 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.

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

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

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.

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

$$sim_J^{\gamma}(tr_1, tr_2) = \frac{|match^{\gamma}(tr_1, tr_2)|}{|tr_1 \cup tr_2|}$$

In the following, we provide a summary of the main characteristics of our *CXK-means* algorithm. Formal details about the scheme of this algorithm can be found in [7].

Data objects are distributed over m nodes and each communicate with all other nodes sending "local" representatives and receiving "global" representatives. Each node N_i is in charge of computing local clusters $C_1^i, \ldots, C_k^i$ and local representatives $c_1^i, \ldots, c_k^i$, but also a subset of the global representatives $c_{m_{i-1}}, \ldots, c_{m_i}$ (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. 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. We define a function that 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. 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 associated with an item e we consider the rank associated with each item (instead of the number of items) having a γ-matching. Tree tuples selected as initial cluster centroids are constrained to come from different XML documents, in order to favor the construction of clusters with low inter-similarity. The termination criterion in *CXK-means* requires the quality of the resulting cluster partition is maximized. An alternative, less expensive exit criterion consists in checking whether clusters are stable, that is checking whether cluster centroids in the current iteration are not changed with respect to the previous iteration.

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 1. 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 2. 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 3. Clustering results with $f \in [0.7..1]$ (structure-driven similarity)

4 Experimental Evaluation

Evaluation methodology and assessment criteria. 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 three types of clustering correspond to different settings of the parameters f and γ, which control the XML transaction similarity function. We varied f within [0..1] with step 0.1, and γ within [0.5..1) with step 0.05. To assess the impact of the network size on the clustering task in terms of both effectiveness and efficiency, we performed experiments by varying the number of nodes in the network from 1 to 19. Data objects were equally partitioned over the nodes. To evaluate 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*, which is defined as the harmonic mean of *precision* and *recall* [2].

Data description We used two real word document collections for the evaluation that are particularly suited for each of the three types of clustering. The *IEEE* data set refers to the IEEE collection version 2.2, that 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. We kept most of the logical structure elements and removed the stylistic markups.In our XML transactional domain, the *IEEE* collection has 211,909 transactions and 135,869 items.The second evaluation data set is a subset of the DBLP archive,[2] a digital bibliography on computer science that 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.

Results Tables 1–3 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 the sake of

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

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

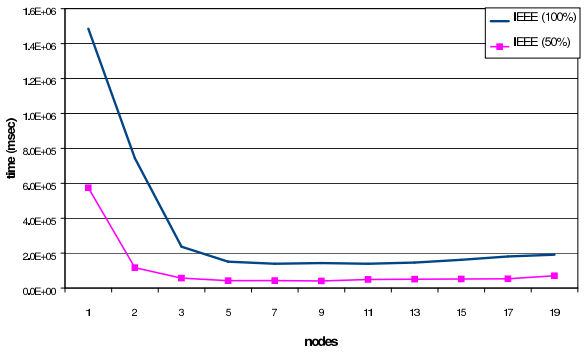

Fig. 1. Clustering time performances on *IEEE* varying the number of nodes and the dataset size

presentation, here results are shown for a maximum number of nodes equal to 9. 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 [16].

As it is reasonable to expect, the centralized case (i.e., one node) led to an upper bound in terms of clustering quality for the collaborative approach. While our focus is not on the evaluation of the centralized case—the interested reader can find details in [16]— we observed how clustering accuracy decreases as the number of nodes increases, regardless of the dataset and the type of clustering. This can be explained since a higher number of nodes corresponds to a lower 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, this performance degradation remained relatively acceptable for a distributed environment.

Figure 1 shows how the time performance in structure/content-driven clustering varied by by increasing the number of nodes on two subsets of *IEEE*; similar results (not shown due to the space limit of this paper) were obtained on *DBLP*. We observed that *CXK-means* takes major advantages w.r.t. a centralized setting in terms of runtime behavior. 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 not be negligible anymore. Indeed, as we can see in Figure 1, 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.

5 Conclusion

We presented a distributed collaborative 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 in a distributed P2P environment. We developed 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 peers on the network. Each peer yields a local clustering solution over its own set of XML data, and exchanges the cluster centroids with other nodes. This sort of recommendation is used to compute global centroids, thus finally obtaining an overall clustering solution in a collaborative way. Experimental evidence has shown that the distributed collaborative approach outperforms the corresponding centralized clusterign setting in terms of runtime behavior, paying a limited loss of accuracy.

References

1. M. Arenas and L. Libkin. A Normal Form for XML Documents. *ACM Transactions on Database Systems*, 29(1):195–232, 2004.
2. 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)*, pages 16–22, 1999.
3. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. ACM Press Books. Addison Wesley, 1999.
4. 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)*, pages 137–148, 2004.
5. M. Eisenhardt, W. Muller, and A. Henrich. Documents by Distributed P2P Clustering. In *GI Jahrestagung (2)*, pages 286–291, 2003.
6. 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)*, pages 238–253, 2003.
7. F. Gullo, G. Ponti, A. Tagarelli, and S. Greco. Collaborative XML Document Clustering. In *Proc. 1th International Workshop on Distributed XML Processing, in conjunction with the 38th IACC Int. Conf. on Parallel Processing (ICPP 2009)*, 2009.
8. K Hammouda and M. Kamel. Collaborative document clustering. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 211–214, 2006.
9. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall advanced reference series. Prentice-Hall, 1988.
10. R. Kargupta and I. Hanzaoglu andB. Stafford. Distributed data mining using an agent based architecture. In *Proc. European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 211–214, 1997.
11. W. Lian, D. W. Cheung, N. Mamoulis, and S.-M. Yiu. An Efficient and Scalable Algorithm for Clustering XML Documents by Structure. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):82–96, 2004.
12. A. Nierman and H. V. Jagadish. Evaluating Structural Similarity in XML Documents. In *Proc. ACM SIGMOD Int. Workshop on the Web and Databases (WebDB)*, pages 61–66, 2002.
13. 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)*, pages 466–477, 2002.
14. A. Strehl, J. Ghosh, and R. Mooney. Impact of Similarity Measures on Web-page Clustering. In *Proc. AAAI Workshop on AI for Web Search*, pages 58–64, 2000.
15. A. Tagarelli and S. Greco. Toward Semantic XML Clustering. In *Proc. SIAM Int. Conf. on Data Mining (SDM)*, pages 188–199, 2006.
16. A. Tagarelli and S. Greco. Semantic Clustering of XML Documents. *ACM Transactions on Information Systems*, 28(1), 2010.