XML Document Partitioning using Ensemble Clustering

Gianni Costa, Riccardo Ortale
{costa,ortale}@icar.cnr.it

ICAR-CNR,
I87036 Rende (CS), ITALY

Abstract. In this paper we propose a new technique for partitioning XML documents, in which conventional clustering techniques operating on flattened representations of individual aspects of the XML documents are combined to partition the available XML corpus. This offers the potential to divide the problem of catching content and structural regularities into simpler subproblems, in which only individual aspects of the XML documents (i.e., either their content, structure or, even, some simple form of nesting of the former into the latter) are taken into account in isolation to perform multiple clusterings of the same XML corpus, with the ultimate purpose of obtaining a refined partition. The results of a preliminary empirical evaluation reveal the effectiveness of our approach.

1 Introduction

The eXtensible Markup Language[1] (XML) [1] is a standard for representing, storing and sharing data encoded as text files in which pure content can be flexibly arranged into a convenient logical structure.

Partitioning XML documents is useful in several applicative settings, including query processing, data integration and indexing, information retrieval and extraction, Web mining, classification of XML documents into hierarchical topic directories for content management and bioinformatics.

In its most general form, the task of partitioning XML documents [4] consists in separating a collection of XML documents into disjoint subsets called clusters. However, from a knowledge discovery point of view, finding clusters in an XML corpus is problematic for various reasons [2, 4]. First, conventional partitioning methods cannot be applied directly to XML documents and, even their adaptation to the XML domain is not trivial. Also there is a challenging research issue, that in principle involves dealing with various aspects of XML documents, e.g., aligning and matching their (sub)structures, catching content overlap, uncovering mutual semantic relationships among respective textual content and/or structural labels along with accounting for the occurrences of common contents into similar (sub)structures. Simpler models for data representation (such as,

[1] http://www.w3c.org

e.g, the vector space model [18]) do not allow to effectively represent the occurrences of text content within structures, whereas more sophisticated models (such as, e.g, the tensor space model [6]) may be demanding in terms of space and processing time. Yet, the number of XML documents is likely to be very large. Additionally, the dimensionality of XML documents (especially those whose nature is of the text-centric type) is huge, since the hierarchical logical structure exacerbates the curse of dimensionality experienced in text clustering. Yet, evaluating content and structural similarities between volumes of XML documents with a huge dimensionality is likely to be detrimental for the effectiveness, efficiency as well as scalability of the partitioning process.

In this paper we propose an unexplored perspective on XML document partitioning, based on combining multiple clusterings [11, 13, 15, 16], in order to obtain a refined partition that satisfies as much as possible multiple clustering criteria. Clustering combination is an especially interesting method, that allows to tackle XML document partitioning from multiple viewpoints. Although the spectrum of possible combinations is very wide and the idea of combining dedicated clustering approaches is certainly interesting and worthy of a deeper investigation, we believe that the joint exploitation of conventional clustering techniques is more appealing, since this offers the potential to avoid the complexity of designing suitable clustering algorithms, that evaluate the similarity between XML documents by taking into account all of the foresaid aspects of XML resemblance simultaneously. We present an XML partitioning technique, XPEC (*XML Partitioning based on Ensemble Clustering*), whose main novelty is the instantiation of the ensemble clustering method through the original combination of three distinct clusterings, in which consolidated and recent developments from the fields of structural XML clustering, information retrieval as well as (high-dimensional) transactional clustering are used. The three clusterings are operated through conventional partitioning approaches, by separately dividing the underlying collection of XML documents with respect to its structure, content and combination of the former two aspects in isolation. The results of a preliminary comparative evaluation on standard benchmark XML corpora reveals the effectiveness of the devised technique.

2 Preliminaries

In this section we introduce the notation adopted throughout the paper along with some basic concepts.

2.1 Tree-based XML Document Representation

The structure and content of the generic XML document with no references [1] can be modeled in terms of a suitable XML tree representation, that refines the traditional notion of *rooted labeled tree* to also catch content and its nesting into structure.

*An **XML tree** is a rooted, labeled, tree, represented as a tuple* $\mathbf{t} = (\mathbf{V_t}, r_\mathbf{t}, \mathbf{E_t}, \lambda_\mathbf{t})$*, whose elements have the following meaning.* $\mathbf{V_t} \subseteq \mathbb{N}$ *is a set of nodes and* $r_\mathbf{t} \in \mathbf{V_t}$

is the root node of $\mathbf{t}$*, i.e. the only node with no entering edges.* $\mathbf{E_t} \subseteq \mathbf{V_t} \times \mathbf{V_t}$ *is a set of edges, catching the parent-child relationships between nodes of* $\mathbf{t}$*. Finally,* $\lambda_t : \mathbf{V_t} \mapsto \Sigma$ *is a node labeling function, where* Σ *is an alphabet of node tags (i.e., labels).*

Notice that the elements of XML documents are not distinguished from their attributes in an XML tree: both are mapped to nodes in the corresponding XML-tree representation.

Let $\mathbf{t}$ be a generic XML tree. Nodes in $\mathbf{V_t}$ can be divided into two disjoint subsets: the set $\mathbf{L_t}$ of *leaves* and the set $\mathbf{V_t} - \mathbf{L_t}$ of *inner nodes*. An inner node has at least one child. A leaf is instead a node with no children, that can contain only textual information.

A root-to-leaf path $p_l^{r_\mathbf{t}}$ in $\mathbf{t}$ is a sequence of nodes encountered along the path from the root $r_\mathbf{t}$ to a leaf node l in $\mathbf{L_t}$, i.e., $p_l^{r_\mathbf{t}} =< r_\mathbf{t}, \ldots, l >$. Notation $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}})$ denotes the sequence of labels that are associated in the XML tree $\mathbf{t}$ to the nodes of path $p_l^{r_\mathbf{t}}$, i.e., $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}) =< \lambda_\mathbf{t}(r_\mathbf{t}), \ldots, \lambda_\mathbf{t}(l) >$. The set of all root-to-leaf paths in $\mathbf{t}$ is denoted as $paths(\mathbf{t}) = \{p_l^{r_\mathbf{t}} | l \in \mathbf{L_t}\}$.

Let l be a leaf in $\mathbf{L_t}$. The set $terms(l) = \{w_1, \ldots, w_h\}$ is a model of the content items provided by l. Elements w_i (with $i = 1 \ldots h$) are as many as the distinct terms in the context of l. Notation $\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).w_h$ indicates an enriched path and will be used to explicitly represent the nested occurrence of the content item w_h in the structural context of the labeled root-to-leaf path $p_l^{r_\mathbf{t}}$. The content of an XML tree $\mathbf{t}$ is denoted as $terms(\mathbf{t}) = \cup_{l \in \mathbf{L_t}} terms(l)$. The set of all enriched paths in $\mathbf{t}$ is instead indicated as $paths^{(e)}(\mathbf{t}) = \cup_{l \in \mathbf{L_t}, w \in terms(l)} \{\lambda_\mathbf{t}(p_l^{r_\mathbf{t}}).w\}$.

Hereinafter, the notions of XML documents and XML tree will be used interchangeably.

2.2 Problem Statement

Our approach combines multiple clusterings of a same XML corpus to eventually find a refined partition. Formally, let $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ be a forest of N XML trees. Assume that $\mathbf{P} = \{\mathcal{P}_1, \ldots, \mathcal{P}_h\}$ is an ensemble of h separate clusterings of $\mathcal{D}$, that can in principle be performed by different algorithms operating on distinct feature sets. The generic $\mathcal{P}_k \in \mathbf{P}$ is a set of l_h clusters, i.e., $\mathcal{P}_k = \{C_1^{(k)}, \ldots, C_{l_h}^{(k)}\}$, such that $\mathcal{D} = \cup_{i=1}^{l_h} C_i^{(k)}$. Ensemble clustering is exploited to build one partition $\mathcal{P} = \{C_1, \ldots, C_l\}$ of l nonempty clusters from $\mathbf{P}$, such that any two XML trees within the same generic cluster $C_i \in \mathcal{P}$ share meaningful content and structural regularities, whereas any two XML trees belonging to as many distinct clusters $C_i, C_j \in \mathcal{P}$ exhibit no relevant similarities.

We develop a specific instance of the general ensemble-clustering method, in which $\mathbf{P}$ consists of three clusterings of $\mathcal{D}$, i.e., $h = 3$ and $\mathcal{P}$ is a partition of $\mathcal{D}$.

2.3 XML Features and Ensemble Clusterings

The tree-based representation of XML documents enables the identification of the sets of XML features addressed in the context of the foresaid three clusterings, which are respectively denoted as $\mathcal{S}^{(p)}$, $\mathcal{S}^{(w)}$ and $\mathcal{S}^{(pw)}$. $\mathcal{S}^{(p)}$ is set of all

distinct root-to-leaf paths in $\mathcal{D}$. $\mathcal{S}^{(w)}$ is a collection of relevant terms for the individual XML documents, i.e., $\mathcal{S}^{(w)} \subset \cup_{\mathbf{t} \in \mathcal{D}} terms(\mathbf{t})$. $\mathcal{S}^{(pw)}$ is a space of relevant enriched paths, i.e., $\mathcal{S}^{(pw)} \subset \cup_{\mathbf{t} \in \mathcal{D}} paths^{(e)}(\mathbf{t})$.

The availability of the sets of XML features $\mathcal{S}^{(p)}$, $\mathcal{S}^{(w)}$ and $\mathcal{S}^{(pw)}$ allows to project the XML trees of $\mathcal{D}$ into suitable corresponding spaces.

More precisely, a space $\mathcal{F}^{(p)}$ is associated with the set $\mathcal{S}^{(p)}$ of structural features, i.e., $\mathcal{F}^{(p)} \triangleq \{\mathcal{F}_{\mathbf{p}} | \mathbf{p} \in \mathcal{S}^{(p)}\}$. The generic element of $\mathcal{F}^{(p)}$ is a boolean attribute, whose value indicates the presence (or absence) of the corresponding root-to-leaf path $\mathbf{p}$ within the individual XML trees of $\mathcal{D}$. Hence, the occurrence of the individual root-to-leaf paths within each XML tree can be explicitly represented by modeling the XML trees as transactions. Assume that $\mathbf{x}^{(\mathbf{t},p)}$ is the transactional representation of an XML tree $\mathbf{t}$ over $\mathcal{F}^{(p)}$. The value of each attribute $\mathcal{F}_{\mathbf{p}}$ within $\mathbf{x}^{(\mathbf{t},p)}$ is true if $\mathbf{p}$ is a root-to-leaf path of $\mathbf{t}$, otherwise it is false. Therefore, $\mathbf{x}^{(\mathbf{t},p)}$ can be modeled as a proper subset of $\mathcal{F}^{(p)}$, namely $\mathbf{x}^{(\mathbf{t},p)} \triangleq \{\mathcal{F}_{\mathbf{p}} \in \mathcal{F}^{(p)} | \mathbf{p} \in paths(\mathbf{t})\}$, with the meaning that the structural features explicitly present in $\mathbf{x}^{(\mathbf{t},p)}$ take value true, whereas the others assume value false. The transactional representation of $\mathcal{D}$ is denoted as $\mathbf{D}^{(p)} = \{\mathbf{x}^{(\mathbf{t},p)} \subset \mathcal{F}^{(p)} | \mathbf{t} \in \mathcal{D}\}$. To deal with the peculiarities of the transactional setting, $\mathbf{D}^{(p)}$ is partitioned through a clustering scheme called Generate-Clusters, that was proposed in [7]. Generate-Clusters is an effective and parameter-free algorithm for transactional clustering, that scales to process volumes of high-dimensional transactions and automatically partitions them into a natural number of clusters. Generate-Clusters operates by progressively partitioning the available transactions, on the basis of the gain in quality of the resulting clusters with respect to the whole transactional database.

Information retrieval methods [3, 5] are instead adopted to represent the relevance of the features from $\mathcal{S}^{(w)}$ and $\mathcal{S}^{(pw)}$ in the context of the individual XML trees. In particular, the vector space model is chosen to represent the XML trees as vectors in the high-dimensional spaces constituted by the elements of either $\mathcal{S}^{(w)}$ or $\mathcal{S}^{(pw)}$ and the *TFIDF* weighting scheme [17] is employed to quantify feature relevance. Formally, let $\overrightarrow{x}^{(\mathbf{t},w)}$ and $\overrightarrow{x}^{(\mathbf{t},pw)}$ be the vector representations of the generic XML tree $\mathbf{t}$ in $\mathcal{D}$ over $\mathcal{S}^{(w)}$ and $\mathcal{S}^{(pw)}$, respectively. $\overrightarrow{x}^{(\mathbf{t},w)}$ and $\overrightarrow{x}^{(\mathbf{t},pw)}$ have as many entries as the elements of the corresponding feature sets $\mathcal{S}^{(w)}$ and $\mathcal{S}^{(pw)}$. The i-th entries of $\overrightarrow{x}^{(\mathbf{t},w)}$ and $\overrightarrow{x}^{(\mathbf{t},pw)}$, denoted as $\overrightarrow{x}_i^{(\mathbf{t},w)}$ and $\overrightarrow{x}_i^{(\mathbf{t},pw)}$, indicate the *TFIDF* value of the respective elements of $\mathcal{S}^{(w)}$ and $\mathcal{S}^{(pw)}$, namely $\mathcal{S}_i^{(w)}$ and $\mathcal{S}_i^{(pw)}$. Specifically, $\overrightarrow{x}_i^{(\mathbf{t},w)} = TFIDF(\mathbf{t}, \mathcal{S}_i^{(w)})$ and $\overrightarrow{x}_i^{(\mathbf{t},pw)} = TFIDF(\mathbf{t}, \mathcal{S}_i^{(pw)})$. The *TFIDF* value of the generic element of $\mathcal{S}^{(w)}$ or $\mathcal{S}^{(pw)}$ is defined as the product of two factors, i.e., $TFIDF(\mathbf{t}, \mathcal{S}_i^{(\bullet)}) = TF(\mathbf{t}, \mathcal{S}_i^{(\bullet)}) \cdot IDF(\mathbf{t}, \mathcal{S}_i^{(\bullet)})$. In turn, $TF(\mathbf{t}, \mathcal{S}_i^{(\bullet)})$ is the occurrence frequency of $\mathcal{S}_i^{(\bullet)}$ in $\mathbf{t}$. $IDF(\mathbf{t}, \mathcal{S}_i^{(\bullet)})$ indicates the inverse document frequency of $\mathcal{S}_i^{(\bullet)}$ in $\mathcal{D}$, i.e., $IDF(\mathbf{t}, \mathcal{S}_i^{(\bullet)}) = log \frac{1+|\mathcal{D}|}{|\mathcal{D}_i|}$, where $\mathcal{D}_i$ is a subset of $\mathcal{D}$, such that for each XML tree $\mathbf{t}' \in \mathcal{D}_i$ it holds that $\mathcal{S}_i^{(\bullet)} \in terms(\mathbf{t}')$ (if $\bullet$ is a placeholder for w) or $\mathcal{S}_i^{(\bullet)} \in paths^{(e)}(\mathbf{t}')$ (if $\bullet$ is a placeholder for pw). Notation $\mathbf{D}^{(w)}$ and $\mathbf{D}^{(pw)}$

is used to indicate the vector representations of $\mathcal{D}$, i.e., $\mathbf{D}^{(w)} = \{\overrightarrow{x}^{(\mathbf{t},w)} | \mathbf{t} \in \mathcal{D}\}$ and $\mathbf{D}^{(pw)} = \{\overrightarrow{x}^{(\mathbf{t},pw)} | \mathbf{t} \in \mathcal{D}\}$.

A globally-optimized repeated-bisecting method, equipped with the cosine distance, is leveraged to partition both $\mathbf{D}^{(w)}$ and $\mathbf{D}^{(pw)}$. In particular, we exploited the implementation available in the *Cluto Toolkit* [12] of the foresaid repeated-bisecting method, which will be henceforth referred to as RBR.

3 XML Partitioning based on Ensemble Clustering

The pseudo code of the XPEC approach is shown in Fig. 1. It works by projecting the individual XML trees of the input forest $\mathcal{D}$ into three distinct spaces of clustering features, separately partitioning the resulting representations and then combining such clusterings into one partition. More precisely, four different stages can be identified. At the first stage (lines 1-4), the XML trees are projected (at line 2) into a space of structural features, which results (at line 3) in the transactional representation $\mathbf{D}^{(p)}$ of the original forest $\mathcal{D}$. GENERATECLUSTERS is subsequently used (at line 4) to group the transactions within $\mathbf{D}^{(p)}$ in the partition $\mathcal{P}^{(p)}$. At the second stage (lines 5-8), the vector representation $\mathbf{D}^{(w)}$ of the original forest $\mathcal{D}$ results (at line 7) from the projection (at line 6) of the XML trees into a space of content items. The partition $\mathcal{P}^{(w)}$ is discovered (at line 8) by RBR in $\mathbf{D}^{(w)}$. At the third stage (lines 9-12), the XML trees are projected (at line 10) into a space of enriched paths to yield the vector representation $\mathbf{D}^{(pw)}$ (at line 11). The latter is partitioned by RBR (at line 12) to form the partition $\mathcal{P}^{(pw)}$. It is worth to underline that RBR requires the number of clusters to find as an input parameter. This is not specified in Fig. 1 for the sake of readability. However, in the experimental evaluation, the number of clusters discovered in both $\mathcal{P}^{(w)}$ and $\mathcal{P}^{(pw)}$ is reported in Fig. 1. Finally, at the forth stage (line 13), $\mathcal{P}^{(p)}$, $\mathcal{P}^{(w)}$ and $\mathcal{P}^{(pw)}$ are suitably combined to find one refined partition of $\mathcal{D}$. The COMBINE procedure used at line 13 of Fig. 1 is detailed in Fig. 2. Essentially, it can be explained as another round of transactional clustering in the space of boolean features induced by the input clusterings $\mathcal{P}_1, \ldots, \mathcal{P}_h$, according to the scheme for the combination of multiple clusterings presented in [15].

Further details on the clustering features used in the former three stages of XPEC are provided below.

3.1 Clustering Features

XPEC relies on the XCPC approach [9] to manipulate the XML documents by their structure alone. This is obtained (at line 1 of Fig. 1) by defining the set $\mathcal{S}^{(p)}$ of relevant structural features as the collection of all distinct root-to-leaf paths in $\mathcal{D}$, i.e., $\mathcal{S}^{(p)} = \cup_{\mathbf{t} \in \mathcal{D}} paths(\mathbf{t})$.

The set $\cup_{\mathbf{t} \in \mathcal{D}} terms(\mathbf{t})$ is the whole space of content items from $\mathcal{D}$. We resort to feature selection in order to distill a subset $\mathcal{S}^{(w)}$ of relevant content items from $\cup_{\mathbf{t} \in \mathcal{D}} terms(\mathbf{t})$, that maintain their original semantic meaning (as opposed to feature extraction methods, that instead would identify artificial features with

XPEC($\mathcal{D}$)
Input: a forest $\mathcal{D} = \{\mathbf{t}_1, \ldots, \mathbf{t}_N\}$ of XML trees;
Output: a partition $\mathcal{P}$ of $\mathcal{D}$;
1: let $\mathcal{S}^{(p)}$ be a set of relevant structural features in $\mathcal{D}$;
2: let $\mathbf{x}^{(\mathbf{t}_i,p)} \leftarrow \{\mathcal{F}_{\mathbf{p}} \in \mathcal{F}^{(p)} | \mathbf{p} \in \mathcal{S}^{(p)}\}$ for each $i = 1, \ldots, N$;
3: let $\mathbf{D}^{(p)} \leftarrow \{\mathbf{x}^{(\mathbf{t},p)} \subseteq \mathcal{F}^{(p)} | \mathbf{t} \in \mathcal{D}\}$;
4: $\mathcal{P}^{(p)} \leftarrow$ GENERATE-CLUSTERS($\mathbf{D}^{(p)}$);
5: let $\mathcal{S}^{(w)}$ be a set of representative content items from $\mathcal{D}$;
6: let $\overrightarrow{x}_i^{(\mathbf{t},w)} \leftarrow TFIDF(\mathbf{t}, \mathcal{S}_i^{(w)})$ for each $\mathbf{t} \in \mathcal{D}$ and $i = 1, \ldots, |\mathcal{S}^{(w)}|$;
7: let $\mathbf{D}^{(w)} \leftarrow \{\overrightarrow{x}^{(\mathbf{t},w)} | \mathbf{t} \in \mathcal{D}\}$;
8: $\mathcal{P}^{(w)} \leftarrow$ RBR($\mathbf{D}^{(w)}$);
9: let $\mathcal{S}^{(pw)}$ be a set of representative enriched paths from $\mathcal{D}$;
10: let $\overrightarrow{x}_i^{(\mathbf{t},pw)} \leftarrow TFIDF(\mathbf{t}, \mathcal{S}_i^{(pw)})$ for each $\mathbf{t} \in \mathcal{D}$ and $i = 1, \ldots, |\mathcal{S}^{(pw)}|$;
11: let $\mathbf{D}^{(pw)} \leftarrow \{\mathbf{x}^{(\mathbf{t},pw)} | \mathbf{t} \in \mathcal{D}\}$;
12: $\mathcal{P}^{(pw)} \leftarrow$ RBR($\mathbf{D}^{(pw)}$);
13: $\mathcal{P} \leftarrow$ COMBINE($\mathcal{P}^{(p)}, \mathcal{P}^{(w)}, \mathcal{P}^{(pw)}$);
14: RETURN $\mathcal{P}$;

Fig. 1. The scheme of the XPEC algorithm

COMBINE($\mathcal{P}_1, \ldots, \mathcal{P}_h$)
Input: multiple separate clusterings $\mathcal{P}_1, \ldots, \mathcal{P}_h$ of $\mathcal{D}$;
L1: /* form one partition $\mathcal{P}$ of $\mathcal{D}$, by combining $\mathcal{P}^{(1)}, \ldots, \mathcal{P}^{(h)}$ through the scheme in [15], which is sketched by the below lines;
L2: $\mathcal{S} \leftarrow \{\mathcal{F}_{C_m^{(l)}} | l = 1, \ldots, h \wedge m = 1, \ldots, |\mathcal{P}^{(l)}|\}$;
L3: $\mathbf{x}^{(\mathbf{t}_i)} \leftarrow \{\mathcal{F}_{C_m^{(l)}} \in \mathcal{S} | \mathbf{t}_i \in C_m^{(l)}\}$ for each $i = 1, \ldots, N$;
L4: $\mathbf{D} \leftarrow \{\mathbf{x}^{(\mathbf{t}_i)} \subseteq \mathcal{S} | \mathbf{t}_i \in \mathcal{D}\}$;
L5: $\mathcal{P} \leftarrow$ GENERATE-CLUSTERS($\mathbf{D}$);
L6: RETURN $\mathcal{P}$;

Fig. 2. The ensemble clustering procedure used at line 13 of Fig. 1

less clear meaning). More precisely, the mean *TFIDF* value [19] of each content item over all the XML documents within $\mathcal{D}$ is used to quantify its relevance to the clustering process.

The task of clustering by both content and structure is functional to drive the combination of the former two clusterings, respectively performed with respect to content and structure in isolation. A simple form of content nesting into structure, namely the enriched paths, is targeted in this clustering process. Focusing on extended paths exacerbates the curse of dimensionality faced when dealing with content only. Therefore, the whole set of enriched paths $\cup_{\mathbf{t} \in \mathcal{D}} paths^{(e)}(\mathbf{t})$ is distilled (at line 9 of Fig. 1) into a smaller set $\mathcal{S}^{(pw)}$. Again, this is accomplished by exploiting the average *TFIDF* value [19] of the enriched paths. $\mathcal{S}^{(pw)}$ is then grown through the addition of constrained rarely occurring enriched paths. Details about the constraints on the occurrence frequency of the added rare enriched paths are provided in Section 4.

4 Evaluation

A preliminary empirical evaluation was conducted to assess the behavior of XPEC. The effectiveness of XPEC is comparatively evaluated against an established competitor. We also expect to gain some preliminary insight into the more general question of the actual advantages deriving from approaching the problem of XML partitioning through the ensemble clustering method. All tests are performed on a Linux machine, with an Intel Core 2 Duo cpu, 4Gb of memory and 2Ghz of clock speed.

The *Sigmod* corpus[2] was chosen as benchmark dataset. It consists of 140 XML documents complying to two different structural class DTDs, namely `IndexTermsPage` and `OrdinaryIssuePage`. Originally, this corpus was used to measure the effectiveness of the approaches to XML clustering at grouping XML documents by structure alone (e.g., in) [2, 8]. However, given the reduced number of structural classes, this is not a challenging task. Therefore, in the following tests, we consider a classification of the *Sigmod* corpus into 5 classes [14]. These were obtained for the purpose of adding complexity to the experimental evaluation, by using expert knowledge to group the underlying XML documents on the basis of their structure and content-based similarity.

On *Sigmod*, the effectiveness of XPEC is compared against the effectiveness of state-of-the-art competitor XCFS [14], whose performances on the chosen XML corpus are reported from [14]. In addition, we report the effectiveness of the clusterings combined by XPEC, for the twofold purpose of providing additional baselines and understanding whether their combination is actually an improvement with respect to the individual clusterings. These are respectively denoted as SO (which stands for the clustering of XML documents based on their structure alone, as it is obtained by applying the GENERATE-CLUSTERS [7] procedure to the structural features selected in Section 3.1, that is equivalent to the approach in [9]), CO (which stands for the clustering of XML documents based on their content alone) and SC (which stands for the clustering of XML documents with respect to the enriched paths).

Effectiveness is measured in terms of *purity*. The latter measure was used in the context of the Mining Track at INEX 2007 [10] and is widely exploited in the literature. Purity is an external quality measure, that assumes knowledge of a predefined classification of the XML documents into a certain number k of natural classes. Therefore, we study the effectiveness of XPEC over collections of XML documents with known class labels and analyze the correspondence between the discovered and natural structures. The available class labels are hidden to XPEC. A partition $\mathcal{P} = \{C_1, \ldots, C_l\}$ of $\mathcal{D}$ can be summarized into a contingency table m, where columns represent discovered clusters and rows represent natural classes. Each entry m_{ij} indicates the number of XML documents in $\mathcal{D}$, that are assigned to cluster C_j and actually belong to the natural class $\mathbf{C}_i$, with $1 \leq i \leq k$. Intuitively, each cluster C_j corresponds to the class $\mathbf{C}_i$ that is best represented in C_j, i.e., such that m_{ij} is maximal. For any cluster C_j, the index $h(j)$ of the class with maximal m_{ij} is defined as $h(j) = max_i\ m_{ij}$. Purity for

[2] kindly provided by Professor Richi Nayak (personal communication)

a cluster C_j is a measure of the degree to which C_j contains XML documents primarily from $\mathbf{C}_{h(j)}$ [4]. Formally, $Purity(C_j) = \frac{|\mathbf{C}_{h(j)}|}{|C_j|}$. Macro-averaged purity and micro-averaged purity extend the notion of purity for a single cluster to a whole partition $\mathcal{P}$. Precisely, macro-averaged purity for partition $\mathcal{P}$ is defined as

$$Macro\text{-}averaged\ purity(\mathcal{P}) = \frac{1}{h} \sum_{C \in \mathcal{P}} Purity(C)$$

Macro-averaged purity is an arithmetic mean, that assigns a same weight to each cluster. Instead, micro-averaged purity weights each cluster by a weight proportional to the size of the cluster itself, i.e.,

$$Micro\text{-}averaged\ purity(\mathcal{P}) = \frac{\sum_{C \in \mathcal{P}} |C| \cdot Purity(C)}{N}$$

Obviously, micro-averaged purity is more strongly influenced by larger clusters. Both micro-averaged purity and macro-averaged purity are measured in our experiments. Larger values of such measures are indicative of higher partitioning effectiveness.

Table 1 shows the results of a comparative evaluation of the effectiveness of XPEC at partitioning XML documents. The empirical evidence reveals that XPEC is the best performer on *Sigmod* corpus, whose XML documents exhibit differentiated structures. Also the combination of SO, CO and SC results into an actual improvement of performance with respect to the individual clusterings.

Corpus	Competitor	Clusters	Micro-averaged purity	Macro-averaged purity
Sigmod	XPEC	5	0.83	0.84
	XCFS [14]	5	0.82	0.49
	SO (or, equivalently, [9])	2	0.79	0.80
	CO	5	0.80	0.83
	SC	5	0.80	0.82

Table 1. Comparative evaluation of clustering effectiveness

5 Conclusions and Further Research

We proposed to exploit the combination of multiple clustering for partitioning XML documents, since in principle it allows to decompose the inherently difficult problem of catching structural and content relationships into a number of simpler subproblems. To verify the validity of our intuition, we developed a technique, XPEC, which separately clusters a same corpus of XML documents with respect to their contents, structure and simple nesting of content items into root-to-leaf paths. The three clusterings are combined by solving a categorical clustering problem, as discussed in [15]. The results of a preliminary evaluation over a real-world XML corpus reveal the effectiveness of our approach. As further research, currently, we are exploring alternative patterns of content nesting into structure.

References

1. S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2000.
2. C.C. Aggarwal, N. Ta, J. Wang, J. Feng, and M. Zaki. Xproj: A framework for projected structural clustering of xml documents. In *Proc. of Int. Conf. on Knowledge Discovey and Data Mining*, pages 46 – 55, 2007.
3. C.C.. Aggarwal and C. Zhai, editors. *Mining Text Data*. Springer, 2012.
4. A. Algergawy, M. Mesiti, R. Nayak, and G. Saake. Xml data clustering: An overview. *ACM Computing Surveys*, 43(4):25:1 – 25:41, 2011.
5. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
6. D. Cai, X. He, and J. Han. Tensor space model for document analysis. Technical report UIUCDCS-R-2006-2715, Computer Science Department, University of Illinois at Urbana-Champaign, 2006.
7. E. Cesario, G. Manco, and R. Ortale. Top-down parameter-free clustering of high-dimensional categorical data. *IEEE Transactions on Knowledge and Data Engineering*, 19(12):1607 – 1624, 2007.
8. G. Costa, G. Manco, R. Ortale, and E. Ritacco. Hierarchical clustering of xml documents focused on structural components. *Data and Knowledge Engineering*, 84:26 – 46, 2013.
9. G. Costa and R. Ortale. On effective xml clustering by path commonality: An efficient and scalable algorithm. In *IEEE Int. Conf. on Tools with Artificial Intelligence*, pages 389 – 396, 2012.
10. L. Denoyer and P. Gallinari. Report on the xml mining track at inex 2007. *ACM SIGIR Forum*, 42(1):22 – 28, 2008.
11. A. Gionis, H. Mannila, and P. Tsaparas. Clustering aggregation. *ACM Transactions on Knowledge Discovery from Data*, 1(1):341 – 352, 2007.
12. G. Karypis. Cluto - software for clustering high-dimensional datasets. Available at `http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download`.
13. L.I. Kuncheva, S.T. Hadjitoorov, and L.P. Todorova. Experimental comparison of cluster ensemble methods. In *Int. Conf. on Information Fusion*, pages 1 – 7, 2006.
14. S. Kutty, R. Nayak, and Y. Li. Xcfs: A novel approach for clustering xml documents using both the structure and the content. In *Proc. of ACM Conference on Information and Knowledge Management*, pages 1729 – 1732, 2009.
15. T. Li, M. Ogihara, and S. Ma. On combining multiple clusterings: an overview and a new perspective. *Applied Intelligence*, 33(2):207 – 219, 2010.
16. S. Vega Pons and J. Ruiz-Shulcloper. A survey of clustering ensemble algorithms. *International Journal of Pattern Recognition and Artificial Intelligence*, 25(3):337 – 372, 2011.
17. G. Salton. Developments in automatic text retrieval. *Science*, 253:974 – 980, 1991.
18. G. Salton, A. Wong, and C.S. Yang. A vector space model for information retrieval. *Communications of the ACM*, 18:613620, 1975.
19. B. Tang, M. Shepherd, E. Milios, and M.I. Heywood. Comparing and combining dimension reduction techniques for efficient text clustering. In *Canadian Conference on AI*, pages 292–296, 2005.