Distance-based Clustering of XML Documents[⋆]

Francesco De Francesca, Gianluca Gordano,
Riccardo Ortale, and Andrea Tagarelli

DEIS, University of Calabria, 87036 Rende (CS) – Italy
e-mail: {ortale, tagarelli}@si.deis.unical.it

The increasing relevance of the Web as a mean for sharing information around the world has posed several new interesting issues to the computer science research community. The traditional approaches to information handling are ineffective in the new context: they are mainly devoted to the management of highly structured information, like relational databases, whereas Web data are semistructured and encoded using different formats (HTML, XML, and so on).

In such context, we address the problem of clustering structurally similar Web documents, and in particular XML documents. This problem has several interesting applications, related, e.g., to the management of Web data. For example, the detection of structural similarities among documents can help in solving the problem of recognizing different sources providing the same kind of information [2], or in the structural analysis of a Web site.

In this paper we propose a novel methodology for clustering XML documents, focusing on the notion of *XML cluster representative*, i.e., a prototype XML document subsuming the most relevant features of the set of XML documents within the cluster. In particular, we devise a technique to compute a representative of a set of XML documents, which is capable of capturing all the structural specificities within the represented documents. To this purpose, the notion of *structural matching* between the trees associated to two XML documents is exploited. Structural matchings allow to both identify the structural similarities between two XML documents and to build a representative around these similarities. We also investigate the exploitation of *merging* and *pruning* strategies for refining XML document trees into effective cluster representatives.

1 Preliminaries

Depending on the specific application domain, the notion of tree matching can be defined in a variety of ways. Here it is exploited in the context of XML trees, i.e. trees resulting from a hierarchy of XML tags, to the purpose of highlighting *common skeletons*. Given two XML documents, a common skeleton is a substructure belonging to both the XML trees: precisely, it is defined as a collection of tags which all exhibit the same name, depth level and parent node. Some definitions at the basis of our approach are provided next.

A tree t is a tuple $t = \langle r_t, V_t, E_t, \delta_t \rangle$ where $V_t \subseteq \mathbb{N}$ is the set of nodes, $E_t \subseteq V_t \times V_t$ is the set of edges, r_t is the root node of t, and $\delta_t : V_t \mapsto \Sigma$ is a

[⋆] This work was partially supported by the National Research Council project SP2: "Strumenti, ambienti e applicazioni innovative per la società dell'informazione - Legge 449/97-99".

node labelling function where Σ is an alphabet of node labels. In addition, we denote by $depth_t(v)$ the depth of a node v in t.

Definition 1 (meaningful matching). *Given two XML trees t_1 and t_2, and $v \in V_{t_1}$, $w \in V_{t_2}$, a* meaningful matching *(henceforth called $m(v,w)$) holds if the following conditions inductively hold: $v = r_{t_1}$, $w = r_{t_2}$ and $\delta_{t_1}(v) = \delta_{t_2}(w)$; otherwise, $\delta_{t_1}(v) = \delta_{t_2}(w)$, $depth_{t_1}(v) = depth_{t_2}(w)$ and $m(a,b)$ holds, where $(a,v) \in E_{t_1}$ and $(b,w) \in E_{t_2}$.*

Note that, in general, multiple matchings may occur when a node in a tree has a meaningful matching with more than one node of a different tree. Formally, given two trees t_1 and t_2, a node $v \in V_{t_1}$ has a *multiple matching* if $\exists w', w'' \in V_{t_2}$ s. t. both $m(v,w')$ and $m(v,w'')$ hold.

The definition below introduces a criterion exploited to overcome the multiple matching problem. The idea is avoiding to take into account those meaningful matchings which are not effectively indicative of strong structural similarities. The relevance of a meaningful matching is evaluated through the weighting function $w_m : V \times V \mapsto \mathbb{N}$, for a set V of nodes.

Definition 2 (match weighting function). *Given two nodes $v \in V_{t_1}$ and $w \in V_{t_2}$, the weight associated with $m(v,w)$ is computed as: $w_m(v,w) = 0$ if $\delta_{t_1}(v) \neq \delta_{t_2}(w)$, $w_m(v,w) = 1 + sum_{v,w}$ otherwise.*

where $sum_{v,w}$ is recursively defined as the sum of the relevance degrees associated to the best matchings between the descendants of v and w. Function w_m weights the prominence of a meaningful matching in such a way that the higher its value, the more relevant the matching itself.

The set of meaningful matchings represent the common paths between two trees. A tree containing only these common paths is defined as a *matching tree*, since it resembles the intersection of the original trees.

Definition 3 (matching tree). *A tree t is a* matching tree *(henceforth called t_m) for two trees t_1, t_2 if $\forall v \in V_t$, $\exists! v_1 \in V_{t_1}$ and $\exists! v_2 \in V_{t_2}$ s. t. $m(v_1, v_2)$ holds. A matching tree t_m is* optimal *if its size is maximal, i.e., if for each matching tree $u_m \neq t_m$, we have $|t_m| \geq |u_m|$.*

2 Problem Statement

Suitable clustering algorithms for semistructured documents were extensively studied in the current literature [5]. Hierarchical methods are widely known as providing clusters with a better quality, especially for text datasets [1, 6]. Our *XRep* algorithm is an adaptation of the agglomerative hierarchical algorithm to our problem. Initially each XML tree is placed in its own cluster, and a matrix containing the pair-wise tree distance is computed. Next, the algorithm walks into an iterative step in which the least dissimilar clusters (evaluated on the basis of their cluster representatives) are merged. As a consequence, the distance matrix is updated to reflect this merge operation. The overall process is stopped when an optimal partition is reached.

We address the problem of clustering XML documents in a parametric way w.r.t. the concepts of distance measure and cluster representative. Viewed in this respect, we need to investigate the conditions for the optimality of a cluster representative, once a suitable distance measure is given.

The notion of proximity, between two patterns drawn from the same feature space, is essential to the definition of a cluster. We mainly focus on suitably adapting a distance measure originally conceived to deal with strings, namely *Edit distance*. Edit distance between trees is computed as the minimum-cost sequence of operations required to convert one given tree to another [7, 4].

Intuitively, a *representative* of a cluster of XML documents is an XML document which effectively reflects all the structural contents within the cluster. The core of the *XRep* algorithm is the computation of the XML cluster representative through three main steps: *i*) (*Tree matching*) first, the information related to all meaningful matchings between the XML trees are suitably combined in order to build an optimal matching tree; *ii*) (*Tree merging*) the optimal matching tree is successively grown into a merge tree by adding to it the uncommon substructures within the original trees; *iii*) (*Pruning of merge tree*) finally, the merge tree is pruned according to a strategy which aims at removing the least frequent nodes: this allows to obtain a representative as an XML tree.

3 Cluster Representative

3.1 XML tree matching

We propose a dynamic-programming algorithm for building the optimal matching tree from two XML trees. The idea is that, at each level, all meaningful matchings are detected and weighted (by means of function w_m). Weights take into account the relevance of meaningful matchings and, as a consequence, allow to overcome multiple matchings. The bottom-up approach to the construction of the optimal matching tree consists of two steps: *i*) matching matrix computation, *ii*) removal of multiple matchings.

Matching matrix computation. Given two XML trees $t_1 = \langle r_{t_1}, V_{t_1}, E_{t_1}, \delta_{t_1} \rangle$ and $t_2 = \langle r_{t_2}, V_{t_2}, E_{t_2}, \delta_{t_2} \rangle$, all meaningful matchings among nodes at the same level within both trees are captured by a matching matrix M_m. M_m has $|V_{t_1}|$ rows and $|V_{t_2}|$ columns. For each level k, a sub-matrix $M_m(k)$ collects the meaningful matchings among the sets of $|V_{t_1}(k)|$ and $|V_{t_2}(k)|$ nodes at level k respectively belonging to t_1 and t_2. The computation starts from the last level of the XML trees and iterates until the roots r_{t_1} and r_{t_2} are reached. At the generic iteration, it verifies if there are nodes, at the same level in both t_1 and t_2, which exhibit meaningful matchings. In these cases, the weight corresponding to each such meaningful matching becomes the value of the entry associated to the matching nodes.

Removal of multiple matchings. A multiple matching is denoted by the presence of at least two non-zero values in some row and/or column of M_m. A node $v_i \in V_{t_1}$, which corresponds to the i-th row in M_m, can match with a number of nodes belonging to t_2. Let $J_{v_i} = \{j_1, \ldots, j_h\}$ be the set of column indexes corresponding to these nodes. Formally, v_i exhibits a multiple matching if $|J_{v_i}| > 1$. Multiple matchings can be eliminated by simply choosing the best matching for each node $v_i \in V_{t_1}$: such a matching corresponds to the column index $j^*_{v_i} = \arg\max_{j_1,\ldots,j_h}\{M_m(i,j_1),\ldots,M_m(i,j_h)\}$. This technique allows the construction of a *marking vector* $V_m = \{j^*_{v_1},\ldots,j^*_{v_n}\}$, whose generic i-th entry indicates the best matching node in t_2 for v_i (both nodes are at the same level).

An optimal matching tree is efficiently built from vector V_m: it suffices that all nodes $v_i \in V_{t_1}$ such that $V_m[i] = -1$ are removed from t_1. These nodes exhibit no meaningful matchings and, as a consequence, are not taken into account.

3.2 Merging of XML trees

This step aims at building an approximation of the actual cluster representative. Starting from t_2 and the above vector V_m, a merge tree $t_{1,2}$ is suitably formed in order to include nodes which reveal to be either common or uncommon to both trees t_1 and t_2: each node is taken into account at most once. Given a node v_i in t_1 with no meaningful matchings, if the parent node of v_i matches with a node $w_j \in V_{t_2}$, then v_i appears in $t_{1,2}$ as a child of w_j.

Both the techniques for building an optimal matching tree and a merge tree are conceived to work with only two XML documents. We show that this does not determine a loss of generality, since merge trees are associative and, as a consequence, can be associated even to clusters consisting of more than two XML documents. A proof of the associativity of the merging process w.r.t. Edit distance is illustrated in [3].

3.3 Representative computation

Pruning is at the basis of the refining process which turns a merge tree into an effective cluster representative. Precisely, leaf nodes are inspected for removal from the merge tree one at a time. Any performed cut minimizes the distance between the refined merge tree and the original document trees in the cluster. The removal of a leaf node from the refined merge tree is reiterated until its distance from the original cluster trees cannot be further decreased. The pruning technique for building a cluster representative is parametric w.r.t. the notion of distance.

References

1. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* ACM Press Books. Addison Wesley, 1999.
2. S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.
3. F. De Francesca, G. Gordano, R. Ortale, and A. Tagarelli. A general framework for XML document clustering. Technical report, n.8, ICAR-CNR, 2003.
4. Carsten Isert. The editing distance between trees. Technical report, Ferienakademie, for course 2: Bume: Algorithmik Und Kombinatorik, Italy, 1999.
5. A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data.* Prentice-Hall, 1988.
6. M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *Proc. ACM SIGKDD Workshop on Text Mining*, 2000.
7. K. Zhang and D. Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM J. Comput.*, 18(6):1245–1262, 1989.