Approximate Subtree Identification in Heterogeneous XML Documents Collections

Ismael Sanz[1], Marco Mesiti[2], Giovanna Guerrini[3], and Rafael Berlanga Llavori[1]

[1] Universitat Jaume I, Castellón, Spain
{berlanga,Ismael.Sanz}@uji.es
[2] Università di Milano, Italy
mesiti@dico.unimi.it
[3] Università di Pisa, Italy
guerrini@di.unipi.it

Abstract. Due to the heterogeneous nature of XML data for internet applications exact matching of queries is often inadequate. The need arises to quickly identify subtrees of XML documents in a collection that are similar to a given pattern. In this paper we discuss different similarity measures between a pattern and subtrees of documents in the collection. An efficient algorithm for the identification of document subtrees, approximately conforming to the pattern, by indexing structures is then introduced.

1 Introduction

The importance of tree-modeled data has spectacularly grown with the emergence of semistructured and XML-based databases. Nowadays, interoperability between systems is achieved through the interchange of XML files, which can represent a great variety of information resources: semi-structured objects, database schemas, concept taxonomies, ontologies, etc. As their underlying data structures use to be tree based, there is a great interest in designing mechanisms for retrieving subtrees according to user requests. In the case of heterogeneous XML collections, these mechanisms must also be approximate, that is, they must retrieve document subtrees that best fit the user requests.

Document heterogeneity poses several challenges to these retrieval systems. Firstly, they must face the problem of dealing with vocabulary discrepancies in the element names (e.g., synonymy, polysemy, etc.). Two elements should match also when their tags are similar relying on a given Thesaurus. Secondly, they must deal with the structural heterogeneity produced by the different DTDs or Schemas behind the stored XML documents. The common assumption of simply weakening the father-children relationships (e.g., the `address` element can be a child or a descendant of the `person` element) is not enough. They should consider the possibility that the father-children relationship is inverted (e.g., the `person` element is a child/descendant of the `address` element) or that the two elements appear as siblings. Moreover, as schemas can also include optional and complex components, structural heterogeneity can appear even for a single schema collection. For these reasons, a user query could be answered by a set of subtrees presenting different structures as well as slight variations in their labels.

In this paper, we stress the structure and tag heterogeneity of XML document collections that can lead to search a very large amount of documents exhibiting weak similarity to a given pattern and we propose an approach for identifying the portions of documents that are similar to the pattern. A pattern is an abstract representation of a user request whose structural constraints are not taken into account in identifying the portions of the target documents in which the nodes of the pattern appear. The structural similarity between the pattern and the identified portions of the target is evaluated as a second step, allowing to rank the identified portions and producing the result.

The proposed approach is thus highly flexible and allows one to choose the desired structural and tag constraints to be taken into account. The problem is however how to perform the first step efficiently, that is, how to efficiently identify *fragments*, i.e. portions of the target containing labels similar to those of the pattern, without relying on strict structural constraints. Our approach employs an ad-hoc data structure, an *inverted index* of the target and a *pattern index* extracted on the fly from the inverted index relying on the pattern labels. By the inverted index, nodes in the target with labels similar to those of the pattern are identified and organized in the levels in which they appear in the target. Fragments are generated by considering the ancestor-descendant relationship among such vertices. Moreover, identified fragments are combined in *regions*, allowing for the occurrence of nodes with labels not appearing in the pattern, if the region shows a higher structural similarity with the pattern than the fragments it is originated from. However, some heuristics are needed to avoid considering all the possible ways of merging fragments into regions and for the efficient computation of similarity.

In the paper, we formally define the notions of fragments and regions and propose the algorithms allowing their identification, relying on our pattern index structure. The use in the approach of different structural similarity functions taking into account different structural constraints (e.g. ancestor-descendant and sibling order) is discussed. The practical applicability of the approach is finally demonstrated by providing experimental results. The contribution of the paper can thus be summarized as follows: (*i*) characterization of different similarity measures between a pattern and regions in a collection of heterogeneous tree structured data; (*ii*) specification of an approach for the efficient identification of regions by specifically tailored indexing structures; (*iii*) realization of a prototype and experimental validation of the approach.

The remainder of the paper is organized as follows. Section 2 formally introduces the notions of pattern, target, fragments, and regions the approach relies on. Section 3 is devoted to discussing different approaches to measure the similarity between the pattern and a region identified in the target. Section 4 discusses how to efficiently identify fragments and regions. Section 5 presents experimental results while Section 6 compares our approach with related work. Finally, Section 7 concludes the paper.

Table 1. Notations

Symbol	Meaning
$root(T)$	The root of T
$\mathcal{V}(T)$	The set of vertices of T (i.e., V)
$\|V\|$,$\|T\|$	The cardinality of $\mathcal{V}(T)$
$label(T)$	The label associated with the root of T
$label(v)$,$label(V)$	The label associated with a node v and the nodes in V
$\mathcal{P}(v)$	The parent of vertex v
$desc(v)$	The set of descendants of v ($desc(v) = \{u \| (v,u) \in E^*\}$)
$level(T)$	The depth of T
$level(v)$	The level in which v appears in T
$pre(v)$,$post(v)$	The pre/post-order traversal rank of v
$nca(v,u)$	The nearest common ancestor of v and u
$Dist(v,u)$	The number of vertices from v to u in a pre-order traversal
$d(v)$,d^{max}	$d(v) = Dist(root(T), v)$, $d^{max} = max_{v \in \mathcal{V}(T)} d(v)$

2 Pattern, Target, Fragment and Region Trees

Trees. Following standard notations, a *tree* T is a pair (V, E), where V is a finite set of *vertices* ($root(T) \in V$ is the tree root) and E is a binary relation on V that satisfies the following conditions: (i) the root has no parent; (ii) every node of the tree except the root has exactly one parent; (iii) all nodes are reachable via edges from the root, i.e. $(root(T), v) \in E^*$ for all nodes in V (E^* is the Klein closure of E). u is the *parent* of v if an *edge* (u, v) belongs to E. A node labelling function *label* assigns to each node in V a label in a set $\mathcal{L}$. Given a tree $T = (V, E)$, Table 1 reports functions/symbols used throughout the paper. In using the notations, the tree T is not explicitly reported whenever it is clear from the context. Otherwise, it is marked as a subscript of the operator.

Document order is determined by a pre-order traversal of the document tree [1]. In a pre-order traversal, a tree node v is visited and assigned its pre-order rank $pre(v)$ before its children are recursively traversed from left to right. A post-order traversal is the dual of the pre-order traversal: a node v is assigned its post-order rank $post(v)$ after all its children have been traversed from left to right. Each node v is thus coupled with a triple $(pre(v), post(v), level(v))$ as shown in Figure 1(a). For the sake of clarity, node triples are reported in the figure only when they are relevant for the discussion. The distance $Dist(v, u)$ between two vertices u, v in the tree is specified as the number of vertices traversed moving from v to u in the pre-order traversal.

Pre- and post- ranking can be used to efficiently characterize the descendants u of v. A node v is a descendant of u, $v \in desc(u)$, iff $pre(v) < pre(u) \wedge post(u) < post(v)$. Given a tree $T = (V, E)$ and two nodes $u, v \in V$, the *nearest common ancestor* of u and v, $nca(u, v)$, is the common ancestor of u and v whose distance to u (and to v) is smaller than the distance to any other common ancestor. Note that $nca(u, v) = nca(v, u)$ and $nca(u, v) = v$ if u is a descendant of v.

Two labels in a tree are *similar* if they are identical, or synonyms relying on a given Thesaurus, or *syntactically similar* relying on a string edit function [2].

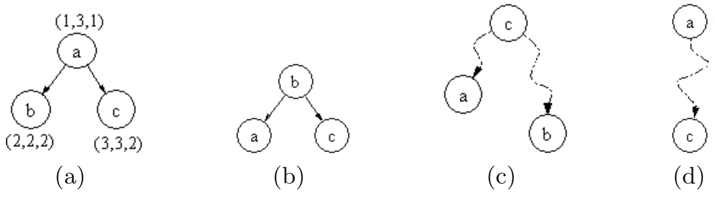


Fig. 1. (a) Pre/Post order rank, a matching fragment with a different order (b), missing levels (c), and missing elements (d)

Let l_1, l_2 be two labels, $l_1 \simeq l_2$ iff: (1) $l_1 = l_2$ or (2) l_1 is a synonym of l_2, or (3) l_1 and l_2 are syntactically similar. Given a label l and a set of labels L, we introduce the operator *similarly belongs*, $\propto$: $l \propto L$ iff $\exists n \in L$ s.t. $l \simeq n$.

Pattern and Target Trees. A pattern is a tree representing a collection of navigational expressions on the target tree (e.g., Xpath expressions in XML documents) or simply a set of labels for which a “preference” is specified on the hierarchical or sibling order in which such labels should occur in the target.

Example 1. Consider the pattern in Figure 1(a). Possible matches are reported in Figure 1(b,c,d). The matching tree in Figure 1(b) contains similar labels but at different positions, whereas the one in Figure 1(c) contains similar labels but at different levels. Finally, the matching tree in Figure 1(d) presents a missing element and both the elements appear at different levels. ○

The target is a set of heterogeneous documents in a source. The target is conveniently represented as a tree whose root is labelled `db` and whose subelements are the documents of the source. This representation relies on the common model adopted by native XML databases (e.g., eXist, Xindice) and simplifies the adopted notations. A target is shown in Figure 2(a).

Definition 1. *(*Target*). Let $\{T_1, \ldots, T_n\}$ be a collection of trees, where $T_i = (V_i, E_i)$, $1 \leq i \leq n$. A* target *is a tree $T = (V, E)$ such that:*

– $V = \cup_{i=1}^{n} V_i \cup \{r\}$, *and* $r \notin \cup_{i=1}^{n} V_i$,
– $E = \cup_{i=1}^{n} E_i \cup \{(r, root(T_i)), 1 \leq i \leq n\}$,
– $label(r) =$ `db`. □

Fragment and Region Trees. The basic building blocks of matchings in our approach are *fragments*. Given a pattern P and a target T, a fragment F is a subtree of T, in which only nodes with labels similar to those in P are considered. Two vertices u, v belong to the same fragment F for a pattern P, iff their labels as well as the label of their common ancestor *similarly belong* to the labels of the pattern. A fragment should belong to a single document of the target.

Definition 2. *(*Fragment Node Set*). A* fragment node set *of a target T with respect to a pattern P is one of the maximal subsets V of $\mathcal{V}(T)$ for which $root(T) \notin V$ and $\forall u, v \in V, label(u), label(v), label(nca(u, v)) \propto label(\mathcal{V}(P))$.* □

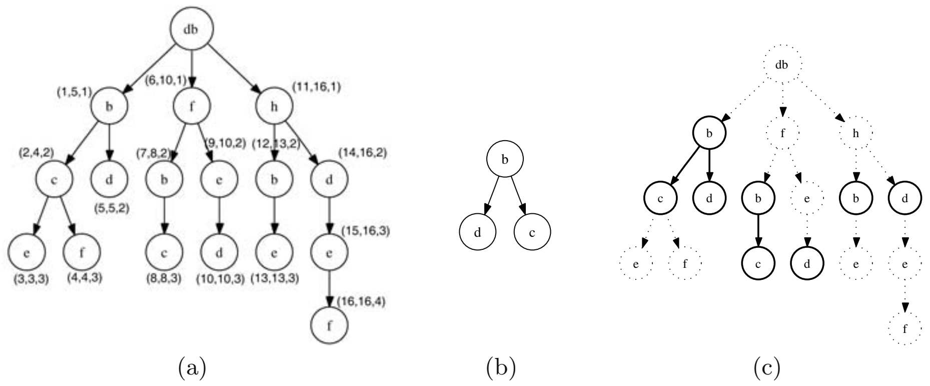


Fig. 2. (a) A target, (b) a pattern, and in bold (c) the corresponding fragments

The tree structure of each fragment is derived from the ancestor-descendant relationship existing among vertices in the target.

Definition 3. *(*Fragment Set*). Let $FN_P(T)$ be the collection of fragment node sets of target T with respect to a pattern P. Every set of vertices $V_F \in FN_P(T)$ defines a* fragment *as the tree $F = (V_F, E_F)$ such that:*

1. *For each $v \in V_F$, $nca(root(F), v) = root(F)$;*
2. *$(u, v) \in E_F$ if u is an ancestor of v in T, and there is no vertex $w \in V_F$, $w \neq u, v$ such that $w \in desc(u)$ and $v \in desc(w)$.* □

Example 2. Consider the pattern in Figure 2(b) and the target in Figure 2(a). The corresponding five fragments are shown in bold in Figure 2(c). ○

Starting from fragments, *regions* are introduced as combinations of fragments rooted at the nearest common ancestor in the target. Two fragments can be merged in a region only if they belong to the same document. In other words, the common root of the two fragments is not the `db` node of the source.

Example 3. Consider as tree T the tree rooted at node $(6, 10, 1)$ in Figure 2(a). Its left subtree contains elements `b` and `c`, whereas its right subtree contains element `d`. T could have a higher similarity with the pattern tree in Figure 2(b) than its left or right subtrees. Therefore, the need arises of combining fragments in regions to return subtrees with higher similarities. ○

Definition 4. *(*Regions*). Let $F_P(T)$ be the set of fragments btw a pattern P and a target T. The corresponding set of regions $R_P(T)$ is defined as follows.*

- *$F_P(T) \subseteq R_P(T)$;*
- *For each $F = (V_F, E_F) \in F_P(T)$ and for each $R = (V_R, E_R) \in R_P(T)$ s.t. $label(nca(root(F), root(R))) \neq$ `db`, $S = (V_S, E_S) \in R_P(T)$, where:*
  - *$root(S) = nca(root(F), root(R))$,*
  - *$V_S = V_F \cup V_R \cup \{root(S)\}$,*
  - *$E_S = E_F \cup E_R \cup \{(root(S), root(F)), (root(S), root(R))\}$.* □

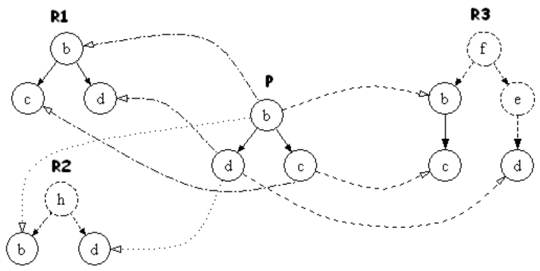


Fig. 3. Identification of different mappings

Figure 3 contains the three regions R_1, R_2, R_3 obtained from the fragments in Figure 2(c). This definition of regions allows one to identify as regions all possible combinations of fragments in a document of the target. This number can be exponential in the number of fragments. In Section 4.2 we will discuss the locality principle to reduce the number of regions to consider.

3 Similarity of a Region w.r.t. a Pattern

In this section we present an approach for measuring the similarity between a pattern and a region. We first identify the possible matches between the vertices in the pattern and the vertices in the region having similar labels. Then, the hierarchical structure of nodes is taken into account to choose, among the possible matches, those that are structurally more similar.

3.1 Mapping Between a Pattern and a Region

A mapping between a pattern and a region is a relationship among their elements that takes the tags used in the documents into account. Our definition differs from the mapping definition proposed by other authors ([3, 4]). Since our focus is on heterogeneous structured data, we do not consider the hierarchical organization of the pattern and the region in the definition of the mapping. We only require that the element labels are similar.

Definition 5. *(*Mapping *M). Let P be a pattern, and R a region subtree of a target T. A mapping M is a partial injective function between the vertices of P and those of R such that $\forall x_p \in \mathcal{V}(P), M(x_p) \neq \perp \Rightarrow label(x_p) \simeq label(M(x_p))$.*$\Box$

Example 4. Figure 3 reports the pattern P in the center and the three regions, R_1, R_2, R_3, obtained from the pattern in Figure 2(a). Dashed lines represent the mappings among the vertices of the pattern and of each region. $\bigcirc$

Several mappings can be determined between a pattern and a region thus measures are required to evaluate the "goodness" of a mapping. The similarity between a pattern and a region depends on the similarity between the vertices having similar labels in the two structures. Possible similarity measures $Sim(x_p, x_r)$ between pairs of vertices will be introduced in next section.

Definition 6. *(*Evaluation of a Mapping *M). Let M be a mapping between a pattern P and a region R. The evaluation of M is:*

$$\mathcal{E}val(M) = \frac{\sum_{x_p \in \mathcal{V}(P) s.t. M(x_p) \neq \perp} \mathcal{S}im(x_p, M(x_p))}{max(|\mathcal{V}(P)|, |\mathcal{V}(R)|)} \qquad \square$$

Once the evaluation of mappings is computed, we define the similarity between a pattern and a region as the maximal evaluation so obtained.

Definition 7. *(*Similarity between a Pattern and a Region*). Let $\mathcal{M}$ be the set of mappings between a pattern P and a region R. Their similarity is defined as:*

$$\mathcal{S}im(P, R) = max_{M \in \mathcal{M}} \mathcal{E}val(M) \qquad \square$$

3.2 Similarity Between Matching Vertices

In this section we present three approaches for computing the similarity between a pair of matching vertices. In the first approach their similarity is one just because we have identified a match in the region. This definition of similarity does not take the structures of the pattern and of the region into account, but just the occurrence of a match. In the second approach, we consider the level at which x_p and $M(x_p)$ appear in the pattern and region structure, respectively. Whenever they appear in the same level, their similarity is equal to the similarity computed by the first approach. Otherwise, their similarity linearly decreases as the number of levels of difference increases. Since two vertices can be in the same level, but not in the same position, a third approach is introduced. Relying on the pre-order traversal of the pattern and the region, the similarity is computed by taking the distance of vertices x_p and $M(x_p)$ with respect to their roots into account. Thus, in this case, the similarity is the highest only when the two vertices are in the same position in the pattern/region.

Definition 8. *(*Similarity between Matching Vertices*). Let P be a pattern, R be a region in a target T, $x_p \in \mathcal{V}(P)$, and $x_r = M(x_p)$. The similarity $\mathcal{S}im(x_p, x_r)$ can be computed as follows:*

1. Match-based similarity*:* $\mathcal{S}im_M(x_p, x_r) = 1$*;*
2. Level-based similarity*:* $\mathcal{S}im_L(x_p, x_r) = 1 - \frac{|level_P(x_p) - level_R(x_r)|}{max(level(P), level(R))}$*;*
3. Distance-based similarity*:* $\mathcal{S}im_D(x_p, x_r) = 1 - \frac{|d_P(x_p) - d_R(x_r)|}{max(d_P^{max}, d_R^{max})}$. $\square$

Example 5. Let x_p be the vertex tagged `d` in the pattern P in Figure 3 and x_r^1, x_r^2, x_r^3 the corresponding vertices in the regions R_1, R_2, R_3. Table 2(a) reports the similarity of x_p to the corresponding vertices in the three regions. Since in each region a vertex tagged `d` appears, the match-based similarity is always 1. The level-based similarity is 1 both for region R_1 and R_2 because the vertex tagged `d` in that regions appears at the same level it appears in the pattern. By

Table 2. (a) Similarity of matching vertices (b) Similarity of a pattern with regions

(a)

	$\mathcal{S}im_M$	$\mathcal{S}im_L$	$\mathcal{S}im_D$
x_r^1	1	1	$\frac{2}{3}$
x_r^2	1	1	$\frac{2}{3}$
x_r^3	1	$\frac{2}{3}$	$\frac{1}{5}$

(b)

	$\mathcal{S}im_M$	$\mathcal{S}im_L$	$\mathcal{S}im_D$
R_1	1	1	$\frac{1+\frac{2}{3}+\frac{2}{3}}{3} = \frac{7}{9}$
R_2	$\frac{2}{3}$	$\frac{1+\frac{1}{2}}{3} = \frac{1}{2}$	$\frac{\frac{2}{3}+\frac{2}{3}}{3} = \frac{4}{9}$
R_3	$\frac{3}{5}$	$\frac{3 \cdot \frac{2}{3}}{5} = \frac{2}{5}$	$\frac{\frac{4}{5}+\frac{1}{5}+1}{5} = \frac{2}{5}$

contrast, the level-based similarity between x_p and x_r^3 is $\frac{2}{3}$ because x_r^3 is at the third level while x_p is at the second level (thus one level of difference) and the maximal number of levels in P and R_3 is 3 $(1-\frac{1}{3})$. The distance-based similarity between x_p and x_r^1 and x_r^2 is the same because in regions R_1 and R_2 vertex `d` has distance 3 while in the pattern it has distance 2. Moreover, the maximal distance in the pattern/region is the same (3). Things are different for region R_3. Indeed, vertex `d` has distance 5 (which is the maximal) whereas the distance in the pattern is 1. Their distance-based similarity is thus $1 - \frac{5-1}{5}$.
Table 2(b) reports the similarity of P with each region. ○

Proposition 1. *$\mathcal{S}im_M$,$\mathcal{S}im_L$ are order-irrelevant. $\mathcal{S}im_D$ is order-relevant.* □

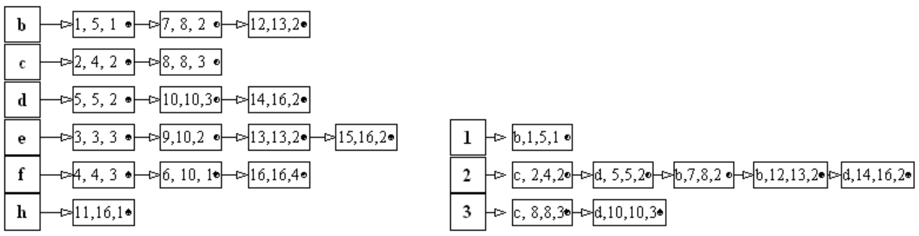
In the previous definition the context of the two vertices is not taken into account. The "context" is formed by vertices in the neighborhood of x_p and x_r that also match. Such vertices can be sibling, ancestor, or descendant vertices of x_p and x_r. Correction factors can be used to tune the similarity obtained by the previous similarity functions. For space constraints we do not present such factors even if they have already been included in our implementation.

4 Fragment and Region Construction

In this section we present an approach for the construction of regions in a target. The first step is to identify fragments F in the target T that satisfy Definition 3. A peculiarity of fragments is that the labels in the path from $root(F)$ and $root(T)$ do not appear in the pattern P. Thus fragments are disjoint subtrees of T. Then, we merge fragments in regions only when the similarity between P and the generated region is greater than the similarity of P with each single fragment. Thus, regions in the target are single fragments or combination of regions with fragments. The identification of the fragments in the target, their merging in regions, and the evaluation of similarity are efficiency performed by exploiting indexing structures.

4.1 Construction of Fragments

Inverted Index for a Target. Starting from the labels $label(T)$ of a target T, the set is normalized with respect to $\simeq$ obtaining $NL(T) = label(T)_{/\simeq}$.



(a) (b)

Fig. 4. (a) Inverted index, (b) pattern index

Each label $l \in NL(T)$ is associated with the list of vertices labeled by l or a similar label, ordered relying on the pre-order rank. For each $v \in \mathcal{V}(T)$, the list contains the 4-tuple $(pre(v), post(v), level(v), \mathcal{P}(v))$. Figure 4(a) depicts the inverted index for the target in Figure 2(a). For the sake of graphical readability, the parent of each vertex is not represented (only a $\cdot$ is reported).

Pattern Index. Given a pattern P, for every node v in P, all occurrences of nodes u in the target tree such that $label(v) \simeq label(u)$ are retrieved, and organized level by level in a *pattern index*. The number of levels of the index depends on the the levels in T in which vertices occur with labels similar to those in the patter. For each level, vertices are ordered according to the pre-order rank. Figure 4(b) contains the pattern index for the pattern in Figure 2(b) evaluated on the target in Figure 2(a).

Identification of Fragments from the Pattern Index. Once the pattern index is generated, the fragments are generated through a visit of the structure and the application of the *desc* function. Each node v in the first level of the pattern index is the root of a fragment because, considering the way we construct the pattern index, no other vertices can be the ancestor of v. Possible descendants of v can be identified in the underlying levels. Given a generic level l of the pattern index, a vertex v can be a root of a fragment iff for none of the vertices u in previous levels, v is a descendant of u. If v is a descendant of a set of vertices U, v can be considered the child of the vertex $u \in U$ s.t. $Dist(v, u)$ is minimal.

The developed algorithm visits each vertex in the pattern index only once by marking in each level the vertices already included in a fragment. Its complexity is thus linearly proportional to the number of vertices in the pattern index. Figure 5 illustrates fragments $F_1, \ldots, F_5$ obtained from the pattern index of Figure 4.

Proposition 2. *Let P be a pattern and T be a target. Let K be the maximal size of a level in the pattern index for P. The complexity of fragment construction is $\mathcal{O}(K \cdot |label(P)| \cdot |NL(T)|)$.* □

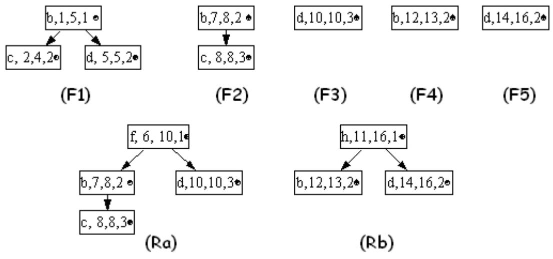


Fig. 5. Construction of fragments and regions

4.2 Construction of Regions

Two fragments should be merged in a single region when, relying on the adopted similarity function, the similarity of the pattern with the region is higher than the similarity with the individual fragments.

Whenever a document in the target is quite big and the number of fragments is high, the regions that should be checked can grow exponentially. To avoid such a situation we exploit the following *locality principle*: merging fragments together or merging fragments to regions makes sense only when the fragments/regions are close. Indeed, as the size of a region tends to be equal to the size of the document, the similarity decreases.

In order to meet such locality principle we evaluate regions obtained by merging adjacent fragments. Operatively, we start from a pair of fragments F, G obtained for a pattern P whose roots have the lowest pre-order rank and identify their common ancestor v. If v is the root of the target, the two fragments cannot be merged. If v is a vertex of the document, the similarity $\mathcal{S}im(P, R)$ is compared with $\mathcal{S}im(P, F)$ and $\mathcal{S}im(P, G)$. If $\mathcal{S}im(P, R)$ is greater than $\mathcal{S}im(P, F)$ and $\mathcal{S}im(P, G)$, F and G are removed and only R is kept. Otherwise, F is kept separate and we try to merge G with the right adjacent fragment.

Example 6. Considering the running example we try to generate regions starting from the fragments in Figure 5. Since the common ancestor between F_1 and F_2 is the root of the target, the two fragments cannot be merged. Since the common ancestor between F_2 and F_3 is a node of the same document, region R_a in Figure 5 is generated. Since the similarity of P with R_a is higher than its similarity with F_2 and F_3, R_a is kept and F_2, F_3 removed. Then, we try to merge region R_a with F_4, but their common ancestor is the root of the target, thus R_a is kept the merging of F_4 and F_5 is evaluated. The region R_b obtained from F_4 and F_5 has an higher similarity than the fragments, thus R_b is kept and F_4 and F_5 are removed. At the end of the process the identified regions are $\{F_1, R_a, R_b\}$. ○

We wish to remark that the construction of regions is quite fast because the target should not be explicitly accessed. All the required information are contained in the inverted indexes. Moreover, thanks to our *locality principle* the number of regions to check is proportional to the number of fragments. Finally, the regions obtained through our process do not present all the vertices occurring

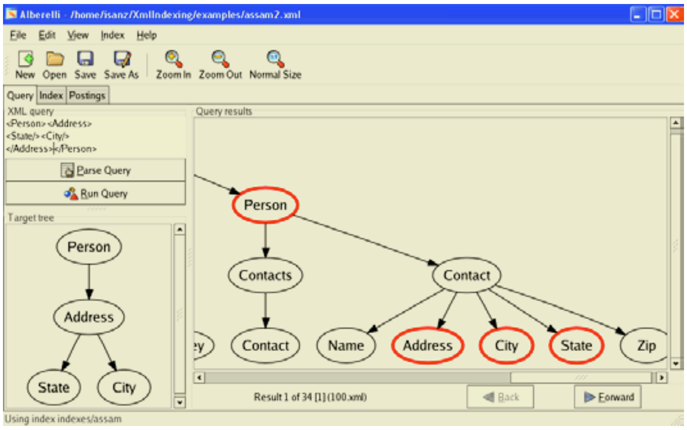


Fig. 6. A screenshot of the GUI prototype on the ASSAM dataset

in the target but only those necessary for the computation of similarity. The evaluation of vertices appearing in the region but not in the pattern is computed through the pre/post order rank of each node.

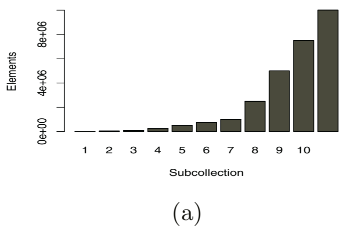
Example 7. Consider the region R_3 in Figure 3 and the corresponding representation R_b in Figure 5. Vertex e is not explicitly present in R_b. However, its lack can be computed by considering the levels of vertex f and vertex d. ○

5 Prototype and Experimental Results

We have developed a prototype of the system, including a indexer module and a query tool written in Python using the Berkeley DB library. A GTK-based graphical user interface is also available; Figure 6 shows a screenshot. Several aspects of the system have been studied: its performance with respect to the dimension of the dataset, its behavior with respect to structural variations, and its effectiveness in a real dataset.

Performance. Two synthetic datasets and test patterns have been developed with different characteristics: Dataset 1 is designed to contain just a few matching results, embedded in a large number of don't-care nodes (around 7500 relevant elements out of 10^7 elements). Dataset 2 has a high proportion of relevant elements (3×10^5 out of 5×10^5). Moreover, to check the performance of pattern identification for a range of dataset sizes, smaller collections have been extracted from our datasets. The characteristics of each dataset are summarized in Figure 7. Results in Figure 8 show that performance is linearly dependent on the size of the result, and don't-care nodes are effectively discarded.

Effect of Structural Distortions. The second aspect we have evaluated is the effect of structural variations in fragments. In order to test this, we have generated another synthetic dataset, in which we have embedded potential matches of



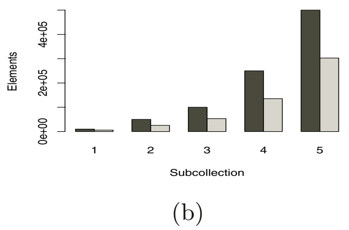
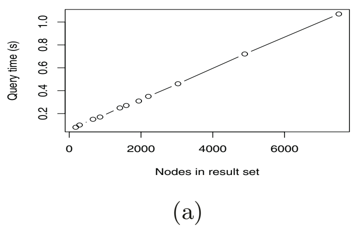


Fig. 7. (a) Total number of elements in each subcollection extracted from Dataset 1 (b) Total number of elements (dark) and number of relevant elements (light) in each subcollection extracted from synthetic Dataset 2



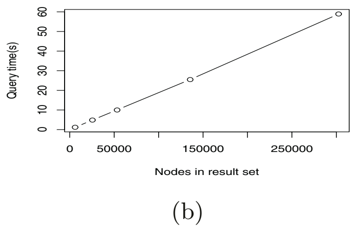
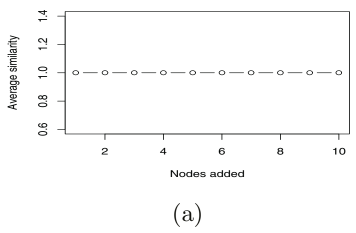


Fig. 8. Execution time in Dataset 1 (left) and Dataset 2 (right)



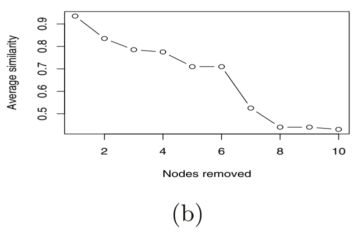


Fig. 9. Change in similarity with the addition and removal of nodes in regions

a 15-element test pattern with controlled distortions of the following kinds: *(1)* addition of n random nodes; *(2)* deletion of n random nodes; *(3)* switching the order of nodes in the same level, with a given probability; *(4)* switching parent and child nodes, with a given probability.

Results in Figure 9 show that added don't-care nodes are ignored by the system, while, predictably, removing relevant nodes in the target does have an effect in the average relevance of results. The results for switching nodes in the same level and for interchanging parent and child nodes are similar to those for nodes addition: fragments with these distortions are successfully retrieved.

Evaluation on the ASSAM Dataset. Preliminar experiments have been conducted on the ASSAM dataset `http://moguntia.ucd.ie/repository/datasets/`, a collection of heterogeneous schemas derived from Web Service descriptions. The original OWL-S schema specifications have been transformed in XML.

Figure 10 shows some sample patterns. The quality of the answers has been evaluated using the traditional information retrieval measures, namely: precision,

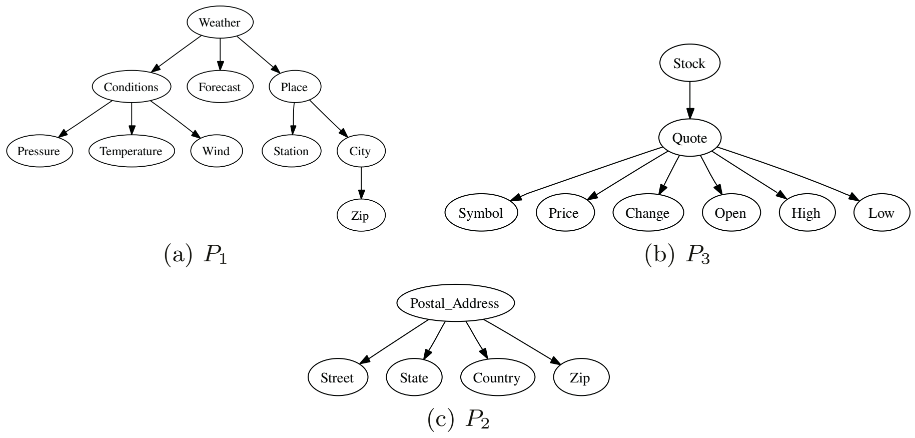


Fig. 10. Sample patterns searched for in the ASSAM dataset

Table 3. Results for the ASSAM datasets

Query	*MinSim*	*Prec.*	*Recall*	F_1
P_1	0.2	1	0.75	0.86
P_2	0.15	0.29	0.7	0.41
P_3	0.2	1	0.8	0.89

recall and the F_1 measure. A threshold $MinSim$ has been considered to discard outliers. $A_{MinSim}(P)$ denotes the results whose Sim_L similarity is greater than $MinSim$. Table 3 shows the results. The first column indicates the pattern, and the second column indicates the similarity threshold used for the answer set. While keeping in mind that these are preliminar tests using a simple, generic distance measure, the results seem encouraging. The low F_1 value for pattern 2 is due to the presence of some of the pattern tags in rather different contexts in the dataset; in a more realistic application, this should be solved by using a more suitable similarity measure.

6 Related Work

In the last decade, there has been a great interest in the *tree embedding* (or *tree inclusion*) problem and its application to semi-structured and XML-databases. Classes of tree inclusion problems are analyzed in [5], providing solutions for the ordered and unordered cases of all of them. The computational cost of the diverse tree-matching primitives varies widely. Simple ordered XPath-like matching has linear complexity, while the unordered version of tree embedding is NP-complete. Flexible and semiflexible matchings have also been proposed in [6] allowing a path in the pattern to match with a path in the target where nodesw appear in a different order. *Ranked tree-matching* approaches have been proposed as well in the same spirit of Information Retrieval (IR) approaches, where approximate answers are ranked on their matching scores. In these approaches, instead of generating all candidate subtrees, the algorithms return a ranked list of "good enough" matches. In [7] a dynamic programming algorithm for ranking query results according to a cost function is proposed. In [8] a data pruning algorithm is proposed where intermediate query results are filtered dynamically during evaluation process. `ATreeGrep` [9] uses as basis an exact matching algorithm, but allowing a fixed number of "differences" in the result.

As these approaches, ours also returns a ranked list of "good enough" subtree matches. However, our approach is highly flexible because it allows choosing the most appropriate structural similarity measures according to the application semantics. All the considered ranked approaches, indeed, enforce at least the ancestor-descendant relationship in pattern retrieval. Moreover, our approach also includes approximate label matching, which allows dealing with heterogeneous tag vocabularies. The proposed tool can thus be used in a wide range of tree-data based applications.

The retrieval of XML documents has also been investigated in the IR area [10]. These approaches mainly focus on the textual part of XML documents, so that the XML structure is used to guide user queries, and to improve the retrieval effectiveness of content indexes. However, current INEX evaluation collections present little heterogeneity in both the tags and structures `http://inex.is.informatik.uni-duisburg.de/2005/`. In the future, the combination of content-based IR techniques with the method proposed in this paper will allow us to extend the range of applications to large text-rich collections with many variations in tag names, structures and content.

Finally, some specific structural similarity measures have been proposed in the areas of Schema Matching [11] and more recently Ontology Alignment. In the former tree/graph-matching techniques allow determining which target schema portions can be mapped to the source ones. In this context, node matching depends on datatypes and domain constraints. However, schemas use to present simple structures that seldom exceed three depth levels (relation-attribute-type). In the latter, the problem consists in finding out which concepts of a target ontology can be associated to concepts of the source ontology, relying on neighborhood and cardinality constraints. However, there are few proposals to state useful tree-based similarity measures that can help this process.

7 Conclusions and Future Work

In this paper we have developed an approach for the identification of subtrees similar to a given pattern in a collection of highly heterogeneous tree structured documents. In this context, the hierarchical structure of the pattern cannot be employed for the identification of the target subtrees but only for their ranking. Peculiarities of our approach are the support for tag similarity relying on a Thesaurus, the use of indexing structures to improve the performance of the retrieval, and a prototype of the system.

As future work we plan to compare different similarity measures in order to identify those more adequate depending on the application context and the heterogeneity of the considered data. Such measures can be collected in a framework of functions that a user can select, compose, and apply depending on her needs. Moreover we plan to consider more sophisticated patterns in which further constraints on vertices and edges can be stated. For instance, an element or a portion of the pattern could be requested to mandatorily appear in the target regions, or the difference between the levels in which two elements appear could be constrained by fixing a threshold. The constraints should then be considered in the similarity evaluation. Finally, we wish to consider subtree identification in a collection of heterogeneous XML Schemas. In this context, the proposed approach should be tailored to the typical schema constraints (e.g., optionality and repeatability of elements, groups, types).

References

1. Grust, T.: Accelerating XPath Location Steps. In: ACM SIGMOD International Conference on Management of Data. (2002) 109–120
2. Wagner, R.A., Fischer, M.J.: The String-to-string Correction Problem. Journal of the ACM **21** (1974) 168–173
3. Nierman, A., Jagadish, H.V.: Evaluating Structural Similarity in XML Documents. In: 5th International Workshop on the Web and Databases. (2002) 61–66
4. Buneman, P., Davidson, S.B., Fernandez, M.F., Suciu, D.: Adding Structure to Unstructured Data. In: 6th International Conference on Database Theory. Volume 1186 of LNCS. (1997) 336–350
5. Kilpeläinen, P.: Tree Matching Problems with Applications to Structured Text Databases. PhD thesis, Dept. of Computer Science, University of Helsinki (1992)
6. Kanza, Y., Sagiv, Y.: Flexible Queries Over Semistructured Data. In: 20th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems. (2001)
7. Schlieder, T., Naumann, F.: Approximate Tree Embedding for Querying XML Data. In: ACM SIGIR Workshop On XML and Information Retrieval. (2000)
8. Amer-Yahia, S., Cho, S., Srivastava, D.: Tree Pattern Relaxation. In: 8th International Conference on Extending Database Technology. Volume 2287 of LNCS. (2002) 496–513
9. Shasha, D., Wang, J.T.L., Shan, H., Zhang, K.: ATreeGrep: Approximate Searching in Unordered Trees. In: 14th International Conference on Scientific and Statistical Database Management. (2002) 89–98
10. Luk, R.W., Leong, H., Dillon, T.S., Chan, A.T., Croft, W.B., Allen, J.: A Survey in Indexing and Searching XML Documents. Journal of the American Society for Information Science and Technology **53** (2002) 415–438
11. Rahm, E., Bernstein, P.A.: A Survey of Approaches to Automatic Schema Matching. The VLDB Journal **10** (2001) 334–350