Fast Detection of XML Structural Similarity

Sergio Flesca, Giuseppe Manco, Elio Masciari, Luigi Pontieri, and Andrea Pugliese, *Student Member*, *IEEE*

Abstract—Because of the widespread diffusion of semistructured data in XML format, much research effort is currently devoted to support the storage and retrieval of large collections of such documents. XML documents can be compared as to their structural similarity, in order to group them into clusters so that different storage, retrieval, and processing techniques can be effectively exploited. In this scenario, an efficient and effective similarity function is the key of a successful data management process. We present an approach for detecting structural similarity between XML documents which significantly differs from standard methods based on graph-matching algorithms, and allows a significant reduction of the required computation costs. Our proposal roughly consists of linearizing the structure of each XML document, by representing it as a numerical sequence and, then, comparing such sequences through the analysis of their frequencies. First, some basic strategies for encoding a document are proposed, which can focus on diverse structural facets. Moreover, the theory of Discrete Fourier Transform is exploited to effectively and efficiently compare the encoded documents (i.e., signals) in the domain of frequencies. Experimental results reveal the effectiveness of the approach, also in comparison with standard methods.

Index Terms—Web mining, mining methods and algorithms, XML/XSL/RDF, text mining, similarity measures.

1 Introduction

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 this 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 textual formats (HTML, XML, email messages, etc.).

In this work, we address the problem of evaluating structural similarity among Web documents and, in particular, among XML documents. As a matter of fact, XML is rapidly becoming the standard in the management of Web data [1], thus calling for suitable data management techniques and tools. Actually, most commercial databases provide tools for delivering and storing XML data, whereas several research prototypes have been developed which use XML as a logical data model [2], [3]. The definition of efficient indexing techniques based on structural similarity [4], [5], [6] can be an effective support for document storage and retrieval.

More interestingly, the computation of structural similarity is crucial to some clustering applications which can be of great value to the management of Web data:

- Many techniques for the extraction of relevant information from semistructured data sources [7], [8], [9] require a preprocessing phase in which Web sources are grouped according to their structural similarity. In such a context, the efficiency and effectiveness of the extraction techniques are strictly related to the structural homogeneity of the discovered clusters.
- A relevant problem in the research on integrating heterogenous sources is to detect groups of sources which provide the same kind of information. Again, effective clustering schemes based on structural similarity are quite helpful in this setting.
- In the context of personalized content delivery, clustering the pages of a Web site according to both structure and content can help to better understand and present the information provided by the site.

The problem of comparing semistructured documents has recently gained increasing attention, even though from different perspectives: for example, in the context of change detection [10], [11], [12], or with the purpose of characterizing a document with regard to a given DTD [13]. Apart from their effectiveness in the application domains considered here, most of these methods are based on the concept of edit distance [14] and use graph-matching algorithms to calculate a (minimum cost) edit script that contains the updates necessary to transform a document into another. These techniques are generally computationally expensive, i.e., at least $O(N^2)$, where N is the number of elements in the two documents.

Our aim and strategy are completely different. The basic intuition exploited in this work is that an XML document has a "natural" interpretation as a time series (namely, a discrete-time signal), in which numeric values summarize some relevant features of the elements enclosed within the document. We can get an example evidence of this observation by simply indenting all the tags in a given document according to their nesting level. Indeed, the sequence of indentation marks (as they appear within the document rotated by 90 degrees) can be looked at as a time series whose shape roughly describes the document's structure.

Hence, a key tool in the analysis of time-series data is the use of the Discrete Fourier Transform (DFT): Some useful

- *G. Manco, E. Masciari, and L. Pontieri are with the Institute of High Performance Computing and Networks (ICAR-CNR), Via Bucci 41c, I87036 Rende (CS), Italy. E-mail: {manco, masciari, pontieri}@icar.cnr.it.*
- *S. Flesca and A. Pugliese are with the University of Calabria, Via Bucci 41c, I87036 Rende (CS), Italy. E-mail: {flesca, apugliese}@si.deis.unical.it.*

Manuscript received 8 Aug. 2003; revised 29 Jan. 2004; accepted 13 May 2004; published online 17 Dec. 2004.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDE-0146-0803.

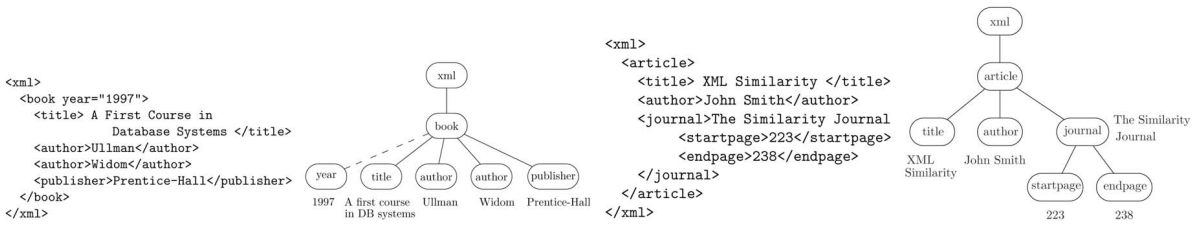


Fig. 1. Example *book* and *article* XML documents and corresponding trees.

properties of the Fourier transform, such as energy concentration or invariance under shifts, enable us to analyze and manipulate signals in a very powerful way. As a matter of fact, the exploitation of the Fourier transform to check similarities among time series is not new (see, e.g., [15], [16], [17]). The main contribution of our approach is the systematic development of effective encoding schemes for XML documents, in a way that makes the use of the Fourier Transform extremely profitable.

The choice of comparing the frequency spectra follows both effectiveness and efficiency issues. Indeed, the exploitation of DFT leads to abstract from structural details which, in most application contexts, should not affect the similarity estimation (such as, e.g., different numbers of occurrences of a shared element or small shifts in the actual positions where it appears). This eventually makes the comparison less sensitive to minor mismatches. Moreover, a frequency-based approach allows us to estimate the similarity through simple measures (e.g., vector distances) which are computationally less expensive than techniques based on the direct comparison of the original document structures.

To summarize, we propose representing the structure of an XML document as a time series, in which each occurrence of a tag in a given context corresponds to an impulse. By analyzing the frequency spectra of the signals, we can hence state the degree of (structural) similarity between documents. It is worth noticing that the overall cost of the proposed approach is only $O(N \log N)$, where N is the maximum number of tags in the documents to be compared. Moreover, it exhibits a satisfactory effectiveness even when compared with more complex tree-edit approaches.

The rest of the paper is organized as follows: In Section 2, we introduce some basic notations, discuss the problem of comparing the structure of XML documents, and give a sketch of our approach. Section 3 illustrates how the structure of an XML document can be encoded into a time series, and presents some methods for accomplishing such a task. Our strategy for comparing two encoded documents is then defined and analyzed in Section 4. Section 5 describes several experiments we performed, on both real and synthesized data, to validate our approach. Finally, in Section 6, we sketch some issues which could be faced in future work on this topic.

2 Problem Statement and Overview of the Proposal

We begin by presenting the basic notation for XML documents that will be used hereafter. An XML document is characterized by *tags*, i.e., terms enclosed between angled brackets. Tags define the structure of an XML document and provide the semantics of the information enclosed. A tag is associated with a *tag name* (representing the term enclosed between angled brackets), and can appear in two forms: either as a *start-tag* (e.g., `<author>`), or as an *end-tag* (characterized by the / symbol, like, e.g., in `</author>`). Finally, a *tag instance* denotes the occurrence of a tag within a certain document. In the *book* XML document of Fig. 1, the tag name `author` is associated with the start-tag `<author>` and the end-tag `</author>`. In the same document, two distinct instances of the `<author>` tag occur.

It is required that, in a well-formed XML document, tags are properly nested, i.e., each start-tag has a corresponding end-tag at the same level. Therefore, an XML document can be considered as an ordered tree, where each node (an *element*) represents a portion of the document, delimited by a pair of start-tag and end-tag instances, and denoted by the tag name associated with the instances. Fig. 1 contains example trees for the *book* and *article* documents. The structure of an XML document corresponds to the shape of the associated tree.

In a tree, several types of structural information can be detected, which correspond to different refinement levels: for example, attribute/element labels, edges, paths, subtrees, etc. Defining the similarity among two documents essentially means choosing an appropriate refinement level and comparing the documents according to the features they exhibit at the chosen level [18]. Different choices may result in rather dissimilar behaviors: In particular, comparing simple structural components (such as, e.g., labels or edges) allows for an efficient computation of the similarity, but typically produces loose-grain similarity values. On the other hand, complex structural components would make the computation of similarity inefficient, and hence unpractical.

Consider, for example, the documents represented in Fig. 2. If a comparison of nodes or edges is exploited, documents *book1* and *book2* appear to be similar, even though the subtrees rooted at the `book` element appear with different frequencies. Accounting for frequencies does not always help: for example, if the order of appearance of the subtrees of the `xml` element in *book3* were changed, the resulting tree still should have the same number of nodes, edges and even paths.

In principle, approaches based on tree-edit distance [14], [19] can better quantify the difference between XML trees; however, they turn out to be too expensive in many applications contexts, as they are generally quadratic with regard to document sizes. Finally, notice that solutions based on detecting local substructures [20], [21], [22] to be used as features may be even hard to handle, for they show two main disadvantages: first, they may imply ineffective representations of the trees in high-dimensional spaces, and, second, costly feature extraction algorithms are required.

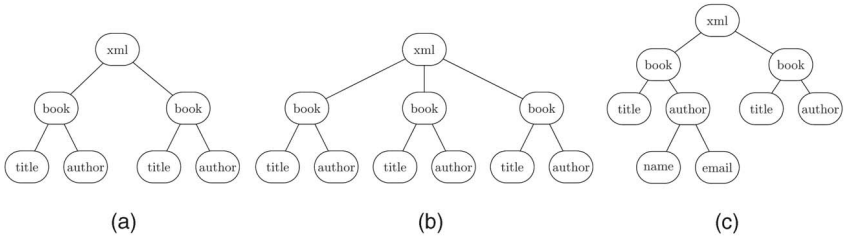


Fig. 2. (a) *book1* and (b) *book2* have the same elements, but with different cardinality. By contrast, (c) *book3* induces a different structure for the `author` element.

In our opinion, an effective notion of structural similarity should take into account a number of issues. First of all, it is important to notice that each document may induce a definition of the elements involved. Thus, an appropriate comparison between two documents should rely on the comparison of such definitions: the more different they are, the more dissimilar are the documents. With reference to documents in Fig. 2, *book1* and *book3* provide a different definition of the `author` element: Indeed, *book1* induces the following definition:

$$< \texttt{!ELEMENT author EMPTY} >,$$

whereas *book3* induces

$$< \texttt{!ELEMENT author (EMPTY}| \\ (\texttt{name}, \texttt{email})) > \\ < \texttt{!ELEMENT name EMPTY} > \\ < \texttt{!ELEMENT email EMPTY} >.$$

Furthermore, even though a similar definition can be devised for an element, the occurrence of a repetition marker in such a definition (such as * or +) can be useful to characterize the differences. For example, the comparison between *book1* and *book2* can be accomplished by observing that both documents comply with

$$< \texttt{!ELEMENT xml (book)}* >,$$

but differ in the number of occurrences of the book element (which is indeed marked by a * symbol).

Clearly, the above features should be considered with a different degree of importance. In particular, a mismatch in the definition of an element should have higher importance than a mismatch in the number of occurrences of an element. In this perspective, *book1* is more similar to *book2* than to *book3*.

Finally, in most application domains, it is mandatory to take into account the hierarchical levels where two documents differ. As a matter of fact, from a semantic viewpoint, an element can be considered as a semantic mark of the whole subtree it encloses. In such a case, differences at higher hierarchical levels should be considered of greater weight than differences at lower levels.

By way of example, consider the documents in Fig. 3: The difference between *doc1* and *doc2* stems from the definition of a:

$$doc1: \quad < \texttt{!ELEMENT a (b, c, e)} > \\ doc2: \quad < \texttt{!ELEMENT a (b, c, f)} >.$$

By contrast, document *doc3* exhibits a different definition of the element `xml` with regard to the other two documents:

$$doc1: \quad < \texttt{!ELEMENT xml (a)} > \\ doc3: \quad < \texttt{!ELEMENT xml (d)} >.$$

Even though in both cases a single mismatch is detected, these mismatches have different weights: In particular, the difference in the definition of `xml` appears at a higher hierarchical level, and hence *doc1* is more dissimilar to *doc3* than to *doc2*.

It is worth noticing that schemes based on tree-edit distance, equipped with simple cost models,[1] do not seem adequate in capturing such differences. Indeed, the tree-edit distance between *doc1* and *doc2* is the same as that between *doc1* and *doc3*, since a simple relabeling operation is required in both cases.

Our main objective is the development of an efficient method which is able to approximate the above features at best. Thus, we can state the problem of finding the structural similarity in a set of XML documents as follows: Given a set $D = \{d_1, \ldots, d_n\}$ of XML documents, we aim at building a similarity matrix S, i.e., a matrix representing, for each pair $(d_i,\ d_j)$ of documents in D, an optimal measure of similarity s_{ij}. Here, optimality refers to the capability of reflecting the above described differences. Observe that we do not address here the problem of finding which parts of two documents are similar or not, as, e.g., tree-edit-based techniques do. However, we believe this is not a strong limitation since in many situations, like those mentioned in the Introduction, a scalar similarity value computed in a quick way can be of greater value than more detailed results requiring expensive computations.

In the following, we propose a technique which is essentially based on the idea of associating each document with a time series representing, in a suitable way, both its basic elements and their relationships within the document. More precisely, we can assume a *preorder* visit of the tree-structure of an XML document. As soon as we visit a node of the tree, we emit an impulse encoding the information corresponding to the tag. The resulting signal shall represent the original document as a time series, from which relevant features characterizing a document can be efficiently extracted. As a consequence, the comparison of two documents can be accomplished by looking at their corresponding signals.

The main features of the approach can be summarized as follows:

- Each element is encoded as a real value. Hence, the differences in the values of the sequence provide for an evaluation of the differences in the elements contained by the documents.

1. Schemes which use more complex cost models for update operations exhibit even higher computational complexities [12].

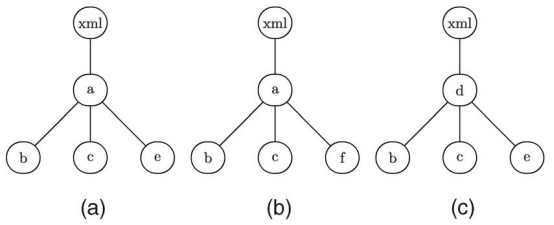


Fig. 3. (a) *doc1* and (b) *doc2* exhibit differences at a leaf node, while *doc1* and (c) *doc3* at a root node.

- The substructures in the documents are encoded using different signal shapes. As a consequence, the analysis of the shapes in the sequences realizes the comparison of the definitions of the elements.
- Context information can be used to encode both basic elements and substructures, so that the analysis can be tuned to handle in a different way mismatches which occur at different hierarchical levels.

In a sense, the analysis of the way the signal shapes differ can be interpreted as the detection of different definitions for the elements involved in the documents. Moreover, the analysis of the frequencies of common signal shapes can be seen as the detection of the differences between the occurrences associated with a repetition marker. In this context, the proposed approach can be seen as an efficient technique, which can satisfactorily evaluate how much two documents are similar with regard to the structural features previously discussed. Notably, the use of time-series for representing complex XML structures, combined with an efficient frequency-based distance function, is the key for quickly evaluating structural similarities: If N is the maximum number of tags in two documents, they can be compared in only $O(N \log N)$ time. In particular, the use of DFT supports the above described notion of similarity: If two documents share many elements having a similar definition, they will be recognized as similar, even when there are repeated and/or optional subelements. Indeed, working on frequency spectra makes the comparison less sensitive to the differences in the number of occurrences of a given element and to small shifts in the actual positions where it occurs in the documents.

Clearly, the effectiveness of the approach relies on the capability of faithfully representing an XML document as a signal. The following section is devoted to investigate such an issue.

3 Document Structure Coding

The structure of an XML document is characterized by elements, which are denoted by start-tags and end-tags. In the following, given an element el of an XML document d, we denote by el_s the start-tag instance of el and with el_e the end-tag instance of el. Since we are only interested in the structure of an XML document, we limit our attention to start-tags and end-tags associated with each element in the document. Furthermore, we deal with element attributes by regarding them as additional (empty) elements: An attribute with name X is treated as a subelement named $ATTRIB@X$. Hence, the structural properties of a document d can be identified by the following sets:

- $tags(d) = \{t | t \text{ is a tag instance in } d\}$;
- $tnames(d) = \{tn | \exists t \in tags(d) \wedge tn \text{ is the name of } t\}$;
- $$stags(d) = \{t | t \in tags(d) \wedge t \text{ is an instance of a start-tag}\};$$
- $$etags(d) = \{t | t \in tags(d) \wedge t \text{ is an instance of an end-tag}\}.$$

For example, the *book* document in Fig. 1 is characterized by the following sets:

$tags(book)$ (τ_1,<xml>),(τ_2,<book>),(τ_3,<ATTRIB@year>),
(τ_4,</ATTRIB@year>″),(τ_5,<title>),(τ_6,</title>),
(τ_7,<author>),(τ_8,</author>),(τ_9,<author>),(τ_{10},</author>),
(τ_{11},<publisher>),(τ_{12},</publisher>),
(τ_{13},</book>),(τ_{14},</xml>)

$tnames(book)$ xml,book,ATTRIB@year,title,author,publisher

$stags(book)$ (τ_1,<xml>),(τ_2,<book>),(τ_3,<ATTRIB@year>),(τ_5,<title>),
(τ_7,<author>),(τ_9,<author>),(τ_{11},<publisher>)

$etags(book)$ (τ_4,</ATTRIB@year>),(τ_6,</title>),
(τ_8,</author>),(τ_{10},</author>),(τ_{12},</publisher>),
(τ_{14},</xml>),(τ_{13},</book>).

Each tag instance in $tags(book)$ is denoted by a pair composed by its unique identifier and its textual representation. Observe also that the definitions of $tags$, $names$, $stags$, and $etags$ can be straightforwardly extended to deal with sets of XML documents.

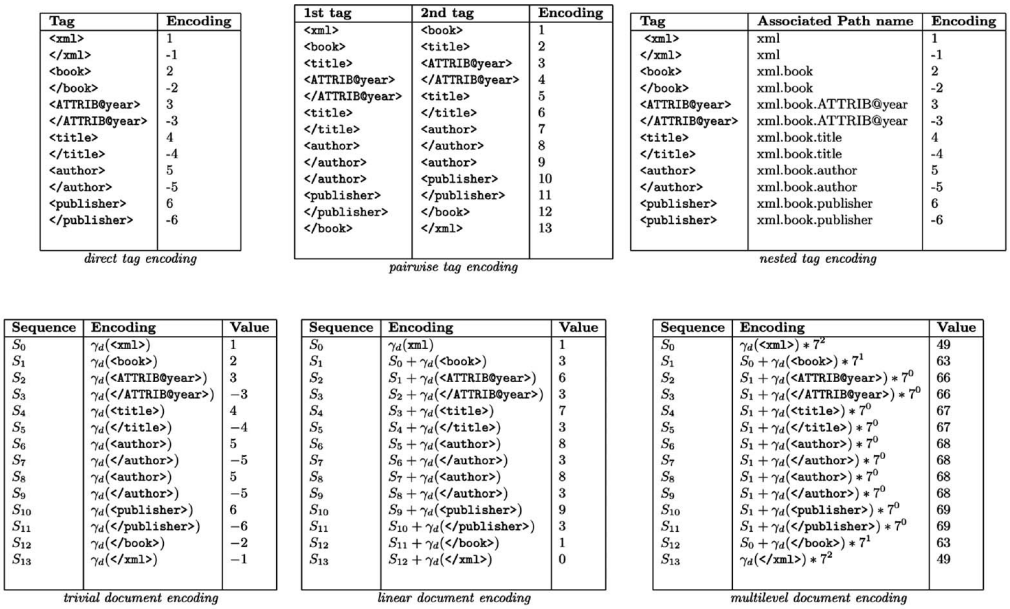
Since XML documents are ordered, it is also necessary to consider the order of appearance of tags within a document. To this end, given an XML document d, we define its *skeleton* ($sk(d)$) as the sequence of all the tag instances appearing within d. The order of $sk(d)$ is defined according to the document order.

Notice that the skeleton is the basic information we need to consider in order to analyze the structural similarity among documents. To this purpose, our technique relies on the effective encoding of the skeleton $sk(d)$ of a document d into a time series properly summarizing its relevant features. We shall analyze several different ways of encoding an XML document according to its skeleton. The resulting encoding schemes are obtained by combining two kinds of encoding functions: A *tag encoding function*, which assigns a real value to each tag instance, and a *document encoding function*, which associates a sequence of reals with the skeleton of a document. In a sense, a tag encoding function corresponds to a *local* analysis of the tags: It considers the general properties of a tag, almost independently of where it appears in the skeleton. On the other side, a document encoding function analyzes the tags from an *overall* perspective, by taking account of the way they are combined within the document at hand.

3.1 Tag Encoding

A tag encoding function is a function that associates a number with each tag instance appearing in the document.

Definition 1. *Given a set D of XML documents, a function $\gamma : tags(D) \rightarrow \mathbb{R} - \{0\}$ is a* tag encoding function *for D. γ*



Tag	Encoding
<xml>	1
</xml>	-1
<book>	2
</book>	-2
<ATTRIB@year>	3
</ATTRIB@year>	-3
<title>	4
</title>	-4
<author>	5
</author>	-5
<publisher>	6
</publisher>	-6

direct tag encoding

1st tag	2nd tag	Encoding
<xml>	<book>	1
<book>	<title>	2
<title>	<ATTRIB@year>	3
<ATTRIB@year>	</ATTRIB@year>	4
</ATTRIB@year>	<title>	5
<title>	</title>	6
</title>	<author>	7
<author>	</author>	8
</author>	<author>	9
</author>	<publisher>	10
<publisher>	</publisher>	11
</publisher>	</book>	12
</book>	</xml>	13

pairwise tag encoding

Tag	Associated Path name	Encoding
<xml>	xml	1
</xml>	xml	-1
<book>	xml.book	2
</book>	xml.book	-2
<ATTRIB@year>	xml.book.ATTRIB@year	3
</ATTRIB@year>	xml.book.ATTRIB@year	-3
<title>	xml.book.title	4
</title>	xml.book.title	-4
<author>	xml.book.author	5
</author>	xml.book.author	-5
<publisher>	xml.book.publisher	6
<publisher>	xml.book.publisher	-6

nested tag encoding

Sequence	Encoding	Value
S_0	γ_d(<xml>)	1
S_1	γ_d(<book>)	2
S_2	γ_d(<ATTRIB@year>)	3
S_3	γ_d(</ATTRIB@year>)	−3
S_4	γ_d(<title>)	4
S_5	γ_d(</title>)	−4
S_6	γ_d(<author>)	5
S_7	γ_d(</author>)	−5
S_8	γ_d(<author>)	5
S_9	γ_d(</author>)	−5
S_{10}	γ_d(<publisher>)	6
S_{11}	γ_d(</publisher>)	−6
S_{12}	γ_d(</book>)	−2
S_{13}	γ_d(</xml>)	−1

trivial document encoding

Sequence	Encoding	Value
S_0	γ_d(xml)	1
S_1	$S_0 + \gamma_d$(<book>)	3
S_2	$S_1 + \gamma_d$(<ATTRIB@year>)	6
S_3	$S_2 + \gamma_d$(</ATTRIB@year>)	3
S_4	$S_3 + \gamma_d$(<title>)	7
S_5	$S_4 + \gamma_d$(</title>)	3
S_6	$S_5 + \gamma_d$(<author>)	8
S_7	$S_6 + \gamma_d$(</author>)	3
S_8	$S_7 + \gamma_d$(<author>)	8
S_9	$S_8 + \gamma_d$(</author>)	3
S_{10}	$S_9 + \gamma_d$(<publisher>)	9
S_{11}	$S_{10} + \gamma_d$(</publisher>)	3
S_{12}	$S_{11} + \gamma_d$(</book>)	1
S_{13}	$S_{12} + \gamma_d$(</xml>)	0

linear document encoding

Sequence	Encoding	Value
S_0	γ_d(<xml>) $* 7^2$	49
S_1	$S_0 + \gamma_d$(<book>) $* 7^1$	63
S_2	$S_1 + \gamma_d$(<ATTRIB@year>) $* 7^0$	66
S_3	$S_1 + \gamma_d$(</ATTRIB@year>) $* 7^0$	66
S_4	$S_1 + \gamma_d$(<title>) $* 7^0$	67
S_5	$S_1 + \gamma_d$(</title>) $* 7^0$	67
S_6	$S_1 + \gamma_d$(<author>) $* 7^0$	68
S_7	$S_1 + \gamma_d$(</author>) $* 7^0$	68
S_8	$S_1 + \gamma_d$(<author>) $* 7^0$	68
S_9	$S_1 + \gamma_d$(</author>) $* 7^0$	68
S_{10}	$S_1 + \gamma_d$(<publisher>) $* 7^0$	69
S_{11}	$S_1 + \gamma_d$(</publisher>) $* 7^0$	69
S_{12}	$S_0 + \gamma_d$(</book>) $* 7^1$	63
S_{13}	γ_d(</xml>) $* 7^2$	49

multilevel document encoding

Fig. 4. Sample tag and document encodings for the *book* document of Fig. 1.

is said to be symmetric *iff for each document* $d \in D$ *and for each element* $el \in d$, $\gamma(el_e) = -\gamma(el_s)$. *Moreover,* γ *is* invariant *if for each document* $d \in D$ *and for each element* $el \in d$, $\gamma(el_e) = \gamma(el_s)$. *Finally, it is* plain *if* $\gamma(el_e) \neq \gamma(el'_s)$, *for each* el', el.

We can assign a number n to each tag instance in several different ways: for instance, by generating it randomly, or using a hash function. However, a suitable tag encoding function should at least ensure *injectivity* with regard to tag names (i.e., tag instances which do not share the same tag name are associated with different numbers): For obvious reasons, collisions may correspond to loss of information. A further desirable property is the capability to *contextualize* a given tag, i.e., to capture information about its possible neighbors.

In the following, we explore three basic types of encoding functions which correspond to different ways of taking account of the information carried by a tag.

Direct Encoding. The simplest tag encoding function we consider is named *direct tag encoding* (denoted by γ_d) and can be obtained as specified below. Given a set D of XML documents, we build a sequence of distinct tag names $[tn_1, tn_2, \cdots, tn_k]$ by considering a (randomly chosen) linear order on $tnames(D)$. Given an element el, the direct encoding simply assigns to each start tag instance el_s the position j of the tag name tn_j of el_s in the sequence ($\gamma_d(el_s) = j$). We complete the above definition by distinguishing among the possible encoding strategies for end-tags: symmetric, invariant, and plain. As an example, consider the *book* document shown in Fig. 1. A sample direct symmetric encoding is shown in Fig. 4, with reference to the tags in the document *book*.

It is worth noticing that, in general, in the proposed encoding, the code associated with a given tag is independent of its neighborhood. Thus, two different instances of the same element are encoded in the same way. In this setting, the value associated with each tag has no semantic meaning, and every linear order on $tnames(D)$ is a suitable direct encoding. However, it is possible to define more effective approaches that, instead, exploit context information. For example, two instances of an `author` element should be considered different if the first appeared within a `book` element and the second within an `article` element. Hence, it is interesting to consider a tag also with regard to the possible contexts where it can appear.

Pairwise Encoding. A simple extension of the direct encoding strategy consists of assigning a value to each tag by relating such value to the subsequent tag in the document. In order to define such an encoding, referred to as *Pairwise encoding*, we need some further notation. We denote by $cpairs(D)$ the pairs of tag instances $\langle t_i, t_{i+1}\rangle$ appearing consecutively in a document $d \in D$. Again, we associate an integer number with each pair in $cpairs(D)$, by considering a randomly chosen linear order. Hence, given a pair of tag instances t_i, t_{i+1} appearing consecutively in a document d, the *Pairwise* tag encoding function $\gamma_{pw}(t_i)$ associates with t_i the number $n_{\langle t_i, t_{i+1}\rangle}$ representing the position of $\langle t_i, t_{i+1}\rangle$ in the sequence. An example encoding is shown in Fig. 4.

Observe that, in the pairwise encoding, the associated numbers are still randomly chosen. Differently from the previous encoding scheme, however, the contextualization of each tag allows to differentiate tag instances having the same tag name, according to their context of appearance.

Nested Encoding. The last strategy we propose stresses this peculiarity, by encoding a tag on the basis of its *path name* (i.e., the sequence of the tag names denoting the path from the root to the tag itself). Consider a set of documents D, and let us denote by $pnames(D)$ the set of path names associated with the elements appearing in any document $d \in D$. Again, we use a sequence of path names $[pn_1, pn_2, \cdots, pn_k]$ obtained by considering a randomly generated linear order on $pnames(D)$, and associate each path name pn_i with its position i in the sequence.

Given a start-tag el_s appearing in a document d with corresponding path name pn_i, the *Nested* tag encoding function $\gamma_{pt}(el_s)$ is defined by associating el_s with i. Again, we distinguish between symmetric, invariant, and plain versions of this encoding. An example for such an encoding is shown in Fig. 4.

3.2 Document Encoding

A document encoding is a function that associates an XML document with a time series. Practically, we are interested in representing the structure of the document itself as a sequence of signal samples. The overall signal's shape corresponds to the structure of the document.

Definition 2. *Let D be the set of all the possible XML documents. A document encoding enc is a function that associates each $d \in D$ with a sequence of real numbers, i.e., $enc(d) = h_0, h_1, \cdots, h_n$. enc is said to be* without structural loss *(WSL) iff for each pair of documents d_1, d_2, $enc(d_1) = enc(d_2)$ implies that $sk(d_1) = sk(d_2)$.*

Of course, *WSL* is a desirable property since this implies that we do not lose information about the document structure by considering its encoding. Indeed, if an encoding satisfies this property, given any document d, we can reconstruct its structure from its encoding $enc(d)$. In the following, we propose three main document encoding functions that can be used to represent the document structure. As we shall see, these functions exploit a tag encoding function to identify suitable sequences. All these encodings are *WSL* under certain assumptions on the tag encoding function adopted.

The simplest encoding function applies a tag encoding function to each tag appearing in the skeleton of the document.

Definition 3. *Let d be an XML document with $sk(d) = [t_0, \cdots, t_n]$ and γ a tag encoding function. A* trivial encoding *of d (denoted by $tenc(d)$) is a sequence $[S_0, S_1, \cdots, S_n]$, where $S_i = \gamma(t_i)$.*

Example 1. The trivial encoding of the *book* document shown in Fig. 1 using the symmetric version of the direct tag encoding function previously introduced is shown in Fig 4. Observe that, by looking at a single impulse, the above encoding does not provide any information about the position of the corresponding tag in the XML tree.

A problem with the above encoding is that each element e of the time series associated with a document d encodes the sole information corresponding to a single tag instance t. However, more informative schemes can be defined, for instance, by looking at the tag instances that occur before t in the document. In practice, we can define the element e in the time series as the *linear* combination of all the encodings of the tags that appear before t in the document.

Definition 4. *Let d be an XML document with $sk(d) = [t_0, \cdots, t_n]$ and γ a tag encoding function. A* linear encoding *of d ($lenc(d)$) is a sequence $[S_0, S_1, \cdots, S_n]$, where $S_0 = \gamma(t_0)$ and $S_i = \sum_{k \le i} \gamma(t_k)$.*

Example 2. The linear encoding of the document shown in Fig. 1 using the symmetric version of the direct tag encoding is shown in Fig. 4. Notice that the linear encoding constructs the encoding signal d in an incremental way, by adding the contribution of each impulse to the overall signal.

In the previously discussed document encoding strategies, the contribution of a tag instance t to the document encoding does not depend on the nesting level of the instance. As discussed in Section 2, the containment relationship between elements of an XML document can be considered relevant as well. Hence, an alternate strategy can be defined, which weights the encoding of each tag t with a factor related to its nesting level l_t. The following encoding is defined to give more relevance to the differences at higher levels in the trees.

Definition 5. *Let D be a set of XML documents, d a document in D with $sk(d) = [t_0, \cdots, t_n]$, and γ a tag encoding function for D. Moreover, let $maxdepth(D)$ represent the maximum depth of any document in D, B a fixed value and $nest_d(t)$ the set of tag instances associated with the ancestors of the element with tag instance t. A* multilevel encoding *of d ($mlenc(d)$) is a sequence $[S_0, S_1, \cdots, S_n]$, where*

$$S_i = \gamma(t_i) \times B^{maxdepth(D)-l_{t_i}} + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times B^{maxdepth(D)-l_{t_j}}.$$

We usually set B as the number of distinct symbols encoded by γ (e.g., $B = |tnames(D)| + 1$ in the case of invariant γ_d). In this way, we avoid "mixing" the contributions of different nesting levels and can reconstruct the path from the root to any tag by only considering the corresponding value in the encoded sequence. In fact, the summation on the right-hand side of the above formula can be interpreted as the integer whose B-base representation is the sequence of the tag codes in $\{\gamma(t_j) \mid t_j \in nest_d(t_i)\}$, ordered by increasing nesting levels of the corresponding tags. Notice that such a property is stronger than *WSL*, and is not mandatory for guaranteeing injectivity in the encoding function.

Example 3. The multilevel encoding of the *book* document, obtained using the symmetric version of the direct tag encoding and setting $B = 7$, is shown in Fig. 4.

3.3 Encoding Schemes

In the following, we shall analyze the encoding functions proposed above, with the main objective of measuring their impact in detecting dissimilarities among documents. We shall concentrate on the following combinations of tag and document encoding functions (referred to as encoding schemes):

- *Trivial encoding*, consisting of the adoption of the Direct Plain Tag encoding and the Trivial Document encoding,
- *Nested encoding*, in which we adopt the Nested Invariant Tag encoding combined with Trivial Document encoding,
- *Linear encoding*, in which we combine Direct Symmetric Tag encoding and Linear Document encoding,
- *Multilevel encoding*, in which we use Direct Invariant Tag encoding and Multilevel Document encoding, and
- *Pairwise Multilevel encoding*, in which we adopt the Pairwise Tag encoding combined with the Multilevel Document encoding.

The idea underlying the above combinations is the following. Trivial and Nested encoding schemes allow for evaluating the effectiveness of a tag encoding function in linearizing the document structure. Notice that linearization is mainly accomplished by a preorder visit of the tree, so that differences in precision of such encodings allow us to evaluate the differences between different tag encoding functions. Linear and Multilevel encodings, on the other

hand, allow the evaluation of the effectiveness of a document encoding function according to the amount of context information taken into account. Essentially, a Linear encoding takes into account the order of the document, while a Multilevel encoding considers the nesting level of the tag instances within the document. Such encodings can be expected to behave differently on documents exhibiting different features: In particular, a Linear encoding can be effective in large documents exhibiting a low degree of nesting, while Multilevel encodings are expected to work fine with documents exhibiting a higher degree.

Finally, the Pairwise encoding summarizes all the above features, but in addition, performs a look-ahead of the tag instances in a document. Hence, it is expected to resume the peculiarities of an XML document with high precision. Experiments reported in Section 5 are aimed at validating the above considerations.

It turns out that all the proposed encoding schemes preserve the document structure, as the following result states.

Theorem 1. *The proposed encoding schemes are* WSL.

Proof. We prove the statement for the multilevel encoding scheme only (the other cases are straightforward adaptations). Given two documents d_1 and d_2, consider the skeletons $sk(d_1) = [t_1, t_2, \ldots, t_n]$ and $sk(d_2) = [p_1, p_2, \ldots, p_n]$, and the associated sequences $mlenc(d_1) = [x_1, x_2, \ldots, x_n]$ and $mlenc(d_2) = [y_1, y_2, \ldots, y_n]$. We show that if $sk(d_1) \neq sk(d_2)$, then $mlenc(d_1) \neq mlenc(d_2)$. Let k be the minimum index s.t. $t_k \neq p_k$. Two cases can occur: either both t_k and p_k are start-tags, or one of them (say p_k) is an end-tag. In the former case, we have:

$$x_k = \gamma_d(t_k) \times B^{M-l_{t_k}} + \sum_{t_i \in nest_{d_1}(t_k)} x_i,$$
$$y_k = \gamma_k(p_k) \times B^{M-l_{p_k}} + \sum_{p_j \in nest_{d_2}(p_k)} y_j,$$

where $M = maxdepth(\{d_1, d_2\})$. Observe that $l_{t_k} = l_{p_k}$ and, since $t_i = p_i$ for each $i < k$, it holds that

$$\sum_{t_i \in nest_{d_1}(t_k)} x_i = \sum_{p_j \in nest_{d_2}(p_k)} y_j.$$

As a consequence, since by definition $\gamma_d(t_k) \neq \gamma_d(p_k)$, we obtain that $x_k \neq y_k$ for every $B > 0$.

For the latter case, observe that, by definition of direct invariant tag encoding,

$$y_k = \gamma_d(p_k) \times B^{M-l_{p_k}} + \sum_{p_j \in nest_{d_2}(p_k)} y_j = \sum_{t_i \in nest_{d_1}(t_k)} x_i.$$

The latter equality follows from $l_{p_k} = l_{t_k} - 1$ and $t_i = p_i$ for each $i < k$. Since $\gamma_d(t_k) > 0$, we obtain $x_k \neq y_k$. □

4 Comparing Documents Using DFT

Faced with the above definitions, we can now detail the similarity measures for XML documents, sketched in the Introduction. As already mentioned, we can assume that we are visiting the tree-structure of an XML document d (using a preorder visit) starting from an initial time t_0. We also assume that each tag instance occurs after a fixed time interval Δ. The total time spent to visit the document is $N\Delta$, where N is the size of $tags(d)$. During the visit, as we find a tag, we produce an impulse which depends on a particular tag encoding function e and on the overall structure of the document (i.e., the document encoding function enc).

As a result of the above physical simulation, the visit of the document produces a signal $h_d(t)$, which usually changes its intensity in the time interval $[t_0, t_0 + N\Delta)$. The intensity variations are directly related to the opening/closure of tags:

$$h_d(t) = \begin{cases} [enc(d)](k) & \text{if } t_0 + k\Delta \leq t < t_0 + (k+1)\Delta \\ 0 & \text{if } t < t_0 \text{ or } t \geq t_0 + N\Delta. \end{cases}$$

Fig. 5a, for example, shows the signals related to the skeletons of the documents reported in Fig. 2, encoded using the *multilevel* encoding scheme, and assuming that for these signals $\Delta = 1$.

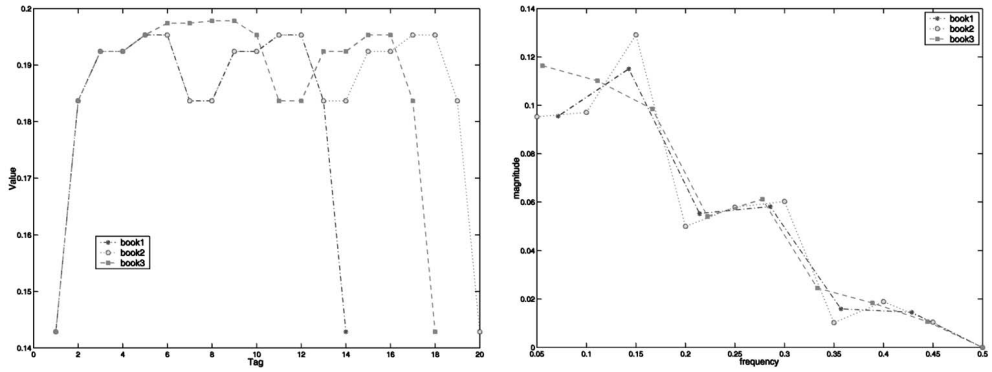
Comparing such signals, however, might be as difficult as comparing the original documents. Indeed, comparing documents having different lengths, requires the combination of both resizing and alignment operations. Moreover, the intensity of a signal strongly depends on the encoding scheme adopted, which can in turn depend from the context (as in the case, e.g., of the multilevel encoding scheme).

A traditional approach for comparing sequences is known in literature as *Time Warping* [23], which mainly consists of considering every possible stretching and narrowing of the two signals, and choosing the best matching. Essentially, time-warping corresponds to a tree-edit over sequences. Hence, it is quite expensive (quadratic in complexity) and, in most cases, the resulting structural similarity of two documents does not necessarily correspond to a similar shape of the associated signals. Consider again Fig. 5a, representing the documents of Fig. 2. Observe that all the signals have different shapes. Notwithstanding, the difference among the signals can be summarized as follows:

- Each `book` element is associated with a unique subsequence within the signals associated with *book1* and *book2*. Nevertheless, the number of occurrences of the subsequences are different.
- *book3* has two different subsequences associated with the `book` elements. Moreover, the first subsequence is different from the ones in *book1* and *book2*.

A comparison in the time domain (accomplished using the time-warping distance) will result in a higher similarity between *book1* and *book3* than between *book1* and *book2*. Nevertheless, each different subsequence triggers a different contribution in the frequency domain, thus allowing for detecting the above described dissimilarities.

To better understand how the differences between two documents reflect on the frequency spectra of their associated encodings, we can always consider these differences separately and exploit the linearity property of the Fourier transform. Consider, e.g., the trees reported in Figs. 6a and 6b, and their encodings obtained using linear or multilevel schemes. The difference between the two trees consists of the name of the root node of subtree `2`. This results in different magnitudes of the portions of the signals corresponding to subtree `2`; more precisely, the difference signal has a rectangular shape and k nonzero values (where k is the cardinality of the subtree). As the Fourier transform



(a) (b)

Fig. 5. (a) Example Signals corresponding to the *book1*, *book2*, and *book3* documents in Fig. 2 and (b) nonzero frequency components of their Discrete Fourier Transform.

of the difference signal has significant frequency components for low frequencies only (in a range whose size is inversely proportional to k) [24], [25], this difference mainly affects low frequencies. Consider now the trees reported in Figs. 6a and 6c. The different number of subtree repetitions produces two signals having almost the same frequency components, but the second one has a higher associated energy. Finally, the trees shown in Figs. 6a and 6d have a nodes containing completely different subtrees (2 and 3). Thus, in the resulting signals, the portions associated with these subtrees have different lengths and values, and their associated transforms will be completely different.

All differences discussed above are reflected in the frequency spectra, and can hence be detected without the need of resorting to edit distance-based algorithms. Thus, to compare the structural properties of two signals associated with two different XML documents, it is particularly convenient to examine their DFT transforms, since they reveal much about the distribution and relevance of signal frequencies. Indeed, some useful properties of such a transform [25], namely, the concentration of the energy into few frequency coefficients, its invariance of the amplitude under shifts, and especially its efficient computation, make it particularly attractive for the problem at hand. Given a document d, we denote as $\mathtt{DFT}(enc(d))$ the Discrete Fourier Transform of the time series resulting from the encoding. In particular, such a transform is represented as a vector whose components corresponding to frequencies in the interval $[-.5, .5]$ (obtained choosing the sampling period $\Delta = 1$). By looking at the DFTs of the signals in Fig. 5a, we can understand how the difference in the structure of these documents are reflected in their transforms. Indeed, the available frequency components summarize the relevant features of the documents and, consequently, the amplitudes associated with them allow to compare such features. In particular, consider Fig. 5b. Here, the zero-frequency components (representing the mean intensity of the signal) as well as the (redundant) negative ones, are omitted. Observe that the DFT transforms of *book1* and *book2* are quite similar. We claim that the similarities in the structure of the two documents are reflected by the nonzero components of the spectra.

In order to compare two documents d_i and d_j, hence, we can exploit the properties of the corresponding transforms. In particular [15], [16], a possibility is to exploit that, by Parseval's theorem, energy (total power) is an invariant in the transformation (and, hence, the information provided by the encoding remains unchanged in the transform). However, a more effective discrimination can exploit the difference in the magnitude of frequency components: In a sense, we are interested 1) in abstracting from the length of the document and 2) in knowing whether a given subsequence (representing a subtree in the XML document) exhibits a certain regularity, no matter where the subsequence is located within the signal. In particular, we aim at considering as (almost) similar documents exhibiting the same subtrees, even if they appear at different positions. Now, as the encoding guarantees that each relevant subsequence is associated with a group of frequency components, the comparison of their magnitudes allows the detection of similarities and differences between documents. Observe that measuring the energy of the difference signal would result in a low similarity value. On the other side, if the phases of the documents' transforms are disregarded, documents are more likely to be considered as similar.

A problem with such an approach can be seen in the fact that, with variable-length sequences, the frequency coefficients may not correspond and thus become incomparable. However, this problem can be tracked by forcing the computation of the DFT on M fixed frequencies, where M is a large number (typically, $M \geq \max(N_i, N_j)$, where $N_i = |tags(d_i)|$ and $N_j = |tags(d_j)|$). The choice of M has strong implications on the interpretation of the corresponding signal in the time domain: In particular, it corresponds to a zero-padding [25]. As a matter of fact, a zero-padding preserving the frequency components of the original signals can be obtained by choosing M as the least common multiple of N_i and N_j. This clearly increases the complexity of the overall approach, as the size of the sequences is notably increased.

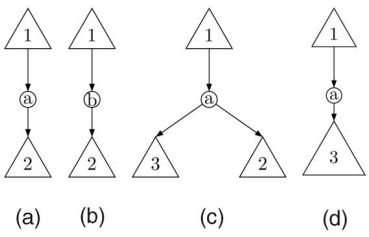


Fig. 6. Sample differences in a documents. All differences are located in a corresponding node of the associated trees.

A viable approximation can be the interpolation of the missing coefficients starting from the available ones. It is worth noticing that the approximation error due to interpolation is inversely proportional to $\min(N_{d_i}, N_{d_j})$: The more elements that are available in a document d, the better the DFT approximates the (continuous) Fourier Transform of the signal $h_d(t)$ and, consequently, the higher is the degree of reliability of interpolation. As a practical consequence, the approach is expected to exhibit good results with large documents, providing poorer performances with small documents.

As a final remark, we observe that, for real-valued sequences, the property $|H(f)| = |H(-f)|$ holds, and that if $h(t)$ is normalized [17] (that is, it has 0-mean), then $H(0) = 0$. This allows us to concentrate only on a subset of the frequency components. To summarize, the overall computation of the dissimilarity between documents can be defined as follows.

Definition 6. *Let d_1, d_2 be two XML documents, and enc a document encoding function, such that $h_1 = enc(d_1)$ and $h_2 = enc(d_2)$. Let DFT be the Discrete Fourier Transform of the (normalized) signals. We define the* Discrete Fourier Transform *distance of the documents as the approximation of the difference of the magnitudes of the DFT of the two encoded documents:*

$$dist(d_1, d_2) = \left(\sum_{k=1}^{M/2} \left(\left| [\tilde{\mathtt{DFT}}(h_1)](k) \right| - \left| [\tilde{\mathtt{DFT}}(h_2)](k) \right| \right)^2 \right)^{\frac{1}{2}},$$

where $\tilde{\mathtt{DFT}}$ is an interpolation of DFT to the frequencies appearing in both d_1 and d_2, and M is the total number of points appearing in the interpolation, i.e., $M = N_{d_i} + N_{d_j} - 1$.

Example 4. Let us consider again the documents of Fig. 2. Notice that the three documents are very similar, although a difference can be detected for each pair under consideration. Adopting the multilevel encoding scheme, we obtain $dist(book1, book2) = .057$ and $dist(book1, book3) = .05772$. As discussed above, these values correspond to the expected values of document structural similarity, as *book1* and *book2* only differ in the number of book elements, whereas *book1* and *book3* have differently structured book elements.

An interesting property is that the above defined distance function is a metric, as proven by the following statement.

Theorem 2. *$dist(d_1, d_2)$ is a metric distance.*

Proof. It suffices to show that triangular inequality holds, since the other properties directly follow from the definition. But, this is trivial, by observing that for any function $f : \mathbb{R}^N \mapsto \mathbb{R}$ and for any distance d defined as $d(\mathbf{x}, \mathbf{y}) = \|f(\mathbf{x}) - f(\mathbf{y})\|$, every vectors $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are such that

$$d(\mathbf{x}, \mathbf{y}) = \|f(\mathbf{x}) - f(\mathbf{y})\| \leq \|f(\mathbf{x}) - f(\mathbf{z})\| + \|f(\mathbf{y}) - f(\mathbf{z})\| = d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z}).$$

□

Alternative ways of comparing documents can be defined. For example, we can choose to compare only a given number of points of the transforms, in the style of [15], [16]. However, it is important to stress here that we aim at comparing to graph-based approaches. To this purpose, it is important to remark that the complexity of the computation of the above distance is mainly influenced by the computation of the DFT.

In the Introduction, we claim that our approach is more efficient than traditional approaches used in literature. Indeed, we can measure the cost of computing a similarity matrix of a set D of documents, by looking at the overall encoding time and transformation time. In particular, the general complexity result can be stated as follows.

Proposition 1. *The encoding schemes defined in Section 3.3, equipped with the distance measure of Definition 6, require $O(N \log N)$ to compare two documents of length N.*

Proof. The proposed encoding schemes require at most two visits of the XML tree. Now, each visit is $O(N)$, since each node within the XML tree represents a tag instance. Moreover, the computation of the Fast Fourier Transform can be done in $O(N \log N)$ [26]. For a given pair of documents, we perform the encoding step and compute the FFT twice. Finally, the computation of the distance function requires M operations, where M is $O(N)$. □

As a consequence, in order to build a similarity matrix for a set of documents D, we require at most $O(|D|^2 N \log N)$, where N is the maximum length of a document in D. Observe that classical techniques based on tree-matching [19] work in at least $O(|D|^2 N^2)$.

5 Evaluation

In this section, we evaluate the proposed approach by performing a set of meaningful experiments on both real and synthesized data. Experiments are mainly devoted to assess the validity of the proposed approach with regard to some prior knowledge about the similarity of selected documents. Here, prior knowledge is modeled as the conformity of a document to its generating DTD: Essentially, documents are grouped in subsets conforming to some previously chosen DTDs, so that each group can be seen as structurally homogenous. Thus, the experiments aim at evaluating the effectiveness of the proposed encoding schemes in recognizing such groups. The evaluation is also accomplished through a comparison with related work on structural similarity detection in XML documents. In particular, we consider a technique recently proposed in [19] and specifically designed to measure structural similarities between XML documents.

To this purpose, a collection of tests is performed, and in each test, some relevant groups of homogeneous documents (*document classes*) are considered. The direct result of each test is a similarity matrix representing the degree of structural similarity for each pair of XML documents in the data set. In order to give an immediate and overall perception of the similarity relationships in the data set, we draw the similarity matrix as an image, where the gray level of each pixel is proportional to the value stored in the corresponding cell of the matrix (i.e., darker pixels correspond to higher similarity values).

A similarity matrix enables simple quantitative analyses, aimed at evaluating the resulting intraclass similarities (i.e., the average of the values computed inside each class), and to compare them with the interclass similarities (i.e., the similarity computed by considering only documents belonging to different classes). To this purpose, values inside

the matrix can be aggregated according to the classes of membership of the related elements: Given a set of documents belonging to n prior classes, a similarity matrix S about these documents can be summarized by an $n \times n$ matrix CS, where the generic element $CS(i,j)$ represents the average similarity between class i and class j:

$$CS(i,j) = \begin{cases} \frac{\sum_{x,y \in C_i, x \neq y} S(x,y)}{|C_i| \times (|C_i|-1)} & \text{iff } i = j \\ \frac{\sum_{x \in C_i, y \in C_j} S(x,y)}{|C_i| \times |C_j|} & \text{otherwise.} \end{cases}$$

The higher the values on the diagonal of the corresponding CS matrix with regard to those outside the diagonal, the higher the ability of the similarity measure to separate different classes.

A further relevant measure is the error rate of a *k*-Nearest Neighbor classifier defined on the basis of the similarity measure. For a given document, we can check whether the dominant class of the k most similar elements allows to correctly predict the actual class of membership. Thus, the total number of documents correctly predicted can be considered as a measure for evaluating the effectiveness of the similarity at hand. Formally, the error $e_k(S)$ of a *k*NN classifier exploiting a similarity matrix S can be defined as

$$e_k(S) = \frac{1}{N} \sum_{i=1}^{N} \gamma_k(i),$$

where N is the total number of documents, and $\gamma_k(i)$ is 0 if the predicted class of the *i*th document coincides with its actual class, and 1 otherwise. Low values of the $e_k(S)$ index correspond to good results.

The above measure can be refined by evaluating the average number of elements, in a range of k elements, having the same class of the document under consideration. Practically, we define q_k as the average percentage of documents in the *k*-neighborhood of a generic document belonging to the same class of that document. Formally:

$$q_k(S) = \frac{1}{N} \sum_{i=1}^{N} \frac{|\mathcal{N}_k(i) \cap Cl(i)|}{min(k, n_i)},$$

where $Cl(i)$ represents the actual class associated with the *i*th document in the collection, $n_i = |Cl(i)|$, and $\mathcal{N}_k(i)$ is the set of k documents having the lowest distances from d_i, according to the similarity measure at hand. In principle, a Nearest Neighbor classifier exhibits a good performance when q_k is high. Furthermore, q_k provides a measure of the stability of a Nearest-Neighbor: High values of q_k make a *k*NN classifier less sensitive to increasing values k of neighbors considered.

The sensitivity of the similarity measure can also been measured by considering, for a given group of documents x, y, z, the probability that x and y belong to the same class and z belongs to a different class, but z is more similar to x than y is. We denote this probability by $\varepsilon(S)$, which is estimated as

$$\varepsilon(S) = \frac{1}{N} \times \sum_{i=1}^{N} \left(\frac{1}{(n_i - 1) \times (N - n_i)} \times \sum_{Cl(j) = Cl(i), j \neq i} \sum_{Cl(k) \neq Cl(i)} \delta_S(i,j,k) \right),$$

where δ_S is 1 if $S(i,j) < S(i,k)$, and 0 otherwise. Once again, low values of $\epsilon(S)$ denote a good performance of the similarity measure under consideration.

In the following experiments, the similarity is computed by exploiting the FFT algorithm described in [27], and whose execution time depends on the length of the transform. In particular, the algorithm is fastest for powers of two. It is almost as fast for lengths that have only small prime factors. It is typically several times slower for lengths that are prime or which have large prime factors. In the latter cases, a modification of the algorithm (in which zero padding to the closest power of two is exploited [26]), can be implemented, thus preserving the quasilinear complexity of the approach.

5.1 Experiments on Synthesized Data

In this section, we illustrate a number of experiments we performed on synthesized data sets by using both our methods and the one proposed in [19]. The experiments are organized in two parts. The first set of tests is meant to analyze the behavior of the methods, by evaluating their effectiveness in recognizing a number of homogeneous, yet rather different, document classes. In the second set of experiments, the methods are examined in a more difficult situation because of the presence of some classes which are very similar to each other. The aim is to evaluate the capability of the methods to separate such groups and still discover high levels of intraclass similarity.

Synthetic data sets were generated with an XML document generator. The generator takes as input a DTD and randomly produces a set of conforming documents, on the basis of suitable statistical models which rule the occurrences of elements marked by operators `*`, `?`, `|`, and `+`. In particular, the `*` operator (and its homologous `+`) is associated with a stochastic variable representing the length of the sequence it may produce, whereas the operators `|` and `?` are both viewed as a Bernoulli experiment which can produce one of two possible outcomes with a given probability. In the experiments described here, the probabilities of all Bernoulli variables are randomly chosen from a uniform distribution, whereas every stochastic variable is modeled by a log-normal function.

The data sets used in this section were generated using the five DTDs shown in Fig. 7. We chose the DTDs according to the requirement that each pair of DTDs must be substantially different, but at the same time it must exhibit some common features. This requirement is useful to evaluate the behavior of the various methods in the absence of completely distinct groups, and to affect in some way the interclass affinities. In Fig. 7, `DTD1` and `DTD2` are comparable in their overall shape, but they have no common tag name. `DTD3` has the same elements of `DTD1`, but with different definitions. Finally, `DTD5` shares tag names with `DTD1` and `DTD3`, and some definitions with `DTD4`.

The quality values obtained by the various encoding schemes are reported in Table 1. For measures e_k and q_k, we considered neighborhoods of size 50, i.e., the size of each class in the data set.

Linear and Trivial encoding schemes provide the worst performances. The images reported in Figs. 8a and 8b show that the proposed schemes are able to clearly distinguish the sole classes 2 and 4. Nevertheless, a closer look at the corresponding CS matrices (Tables 2a and 2b), reveals the effectiveness of the encodings. Indeed, average intraclass similarity is generally higher than interclass similarity, thus enabling a hypothetic agglomerative hierarchical clustering to build a dendrogram as follows:

```
<!DOCTYPE DTD1 [
   <!ELEMENT XML (a*) >
   <!ELEMENT a (b,c,d,e*) >
   <!ELEMENT b (f?)  >
   <!ELEMENT c (g|h) >
   <!ELEMENT d EMPTY >
   <!ELEMENT e EMPTY >
   <!ELEMENT f EMPTY >
   <!ELEMENT g EMPTY >
   <!ELEMENT h EMPTY >
]>
<!DOCTYPE DTD4 [
   <!ELEMENT XML ((x,y)*) >
   <!ELEMENT x ((a,w)|z*) >
   <!ELEMENT a EMPTY >
   <!ELEMENT w (c?)  >
   <!ELEMENT c EMPTY >
   <!ELEMENT z (v,c) >
   <!ELEMENT v EMPTY >
   <!ELEMENT y EMPTY >
]>
```

```
<!DOCTYPE DTD2 [
   <!ELEMENT XML (a1*) >
   <!ELEMENT a1 (b1,c1,d1,e1*) >
   <!ELEMENT b1 (f1?)  >
   <!ELEMENT c1 (g1|h1) >
   <!ELEMENT d1 EMPTY >
   <!ELEMENT e1 EMPTY >
   <!ELEMENT f1 EMPTY >
   <!ELEMENT g1 EMPTY >
   <!ELEMENT h1 EMPTY >
]>
   <!DOCTYPE DTD5 [
      <!ELEMENT XML (m*,n) >
      <!ELEMENT m (q*) >
      <!ELEMENT q (x,y) >
      <!ELEMENT x ((a,c)|z*) >
      <!ELEMENT a EMPTY >
      <!ELEMENT c EMPTY >
      <!ELEMENT z EMPTY >
      <!ELEMENT n EMPTY >
      <!ELEMENT y EMPTY >
   ]>
```

```
<!DOCTYPE DTD3 [
   <!ELEMENT XML (h*) >
   <!ELEMENT h (f,g) >
   <!ELEMENT f (d*) >
   <!ELEMENT g (b|c) >
   <!ELEMENT d (a?)  >
   <!ELEMENT b EMPTY >
   <!ELEMENT c EMPTY >
   <!ELEMENT a EMPTY >
]>
```

Fig. 7. Example DTDs for synthesized data.

- First, classes C_4 and C_2, exhibiting the highest intraclass similarities, are recognized.
- After the removal of these classes, the average similarities lower of an order of magnitude, thus allowing for detecting classes C_1, C_3 and, finally, C_5.

The performance of Nested encoding is shown in Fig. 8c. As we can see, the differences among the various classes are now marked with higher precision. This is mainly due to the fact that, in this scheme, each tag contains information about its ancestor tags in the XML tree and, hence, its code resumes, in some way, the structure of the document. Notwithstanding, because of the adoption of a simple document encoding function, this scheme is unable to neatly separate some regions: In particular, class 1 does not exhibit substantial differences from class 2.

Things get significantly better with encoding schemes based on the multilevel technique, which produce the best results. For example, the multilevel scheme exhibits a neat distinction between elements belonging to a class and elements outside that class. Fig. 8d clearly puts in evidence such a behavior. It is interesting to take a closer look at the average values within the similarity matrix in Table 2d, as it shows that both average intrasimilarities and average intersimilarities are particularly high. Such an effect is mainly due to the fact that, in the proposed encoding, the values in the sequence are dominated by the encoding of the path. As a consequence, sequences associated to documents having common ancestors exhibit low variances. The Pairwise Multilevel encoding (shown in Fig. 8e) avoids this drawback and clearly separates the contours around each class. This is clearly shown by the average values of Table 2e. In such a case, the benefits of considering the path of each element are strengthened by the contextualization of each tag, which causes a higher irregularity in the corresponding signals.

TABLE 1
Quality Indices on Synthesized Data

method/index	ε	$e_{k=50}$	$q_{k=50}$
Trivial	0.1803	0.2840	0.6353
Nested	0.0892	0.1880	0.7417
Linear	0.1831	0.1960	0.6361
Multilevel	0	0	1
Pairwise Mult.	0	0	1
Tree-Edit distance	0.1131	0.1040	0.8248

Interestingly, similarities based on multilevel encoding outperform the measure based on Tree-Edit script (shown in Fig. 8f). The latter, indeed, exhibits good performances compared to the schemes based on trivial and linear encoding, but does not recognize class 3 (as, instead, schemes based on contextualization do).

It is useful to examine in more detail the capability of distinguishing different degrees of interclass similarity. To this purpose, the similarity measures corresponding to the various schemes are tested in a more stressing situation, where document classes very similar to each other, i.e., generated from similar DTDs, appear in the data set. In particular, we considered two further DTDs, namely, DTD6 and DTD7 (displayed in Fig. 9), both having a structure very similar to that of DTD5. Notice that DTD5 and DTD6 share similar definitions (in particular, for elements close to the root): Hence, we expect a higher degree of interclass similarity between them. The results of the experiments are summarized in Table 3.

In general, the introduction of the new DTDs causes a global reduction of the quality values for all the encoding schemes. Notwithstanding, the Multilevel and the Pairwise Multilevel encoding schemes are still able to distinguish the classes. Notably, Figs. 10a and 10b show that both the techniques separate each class, and detect a higher interclass similarity among documents in classes DTD5, DTD6, and DTD7, as expected. Tables 4a and 4b also show a higher interclass similarity between DTD5 and DTD6.

By contrast, Tree-Edit distance (Fig. 10c) exhibits a different behavior on the three classes. In general, the average interclass similarity is low, compared to the average interclass similarity with the remaining classes (as shown in Table 4c). Moreover, class DTD6 exhibits a low interclass similarity with regard to DTD5, and, surprisingly, a higher interclass similarity with regard to DTD7 (actually, higher than intraclass similarity).

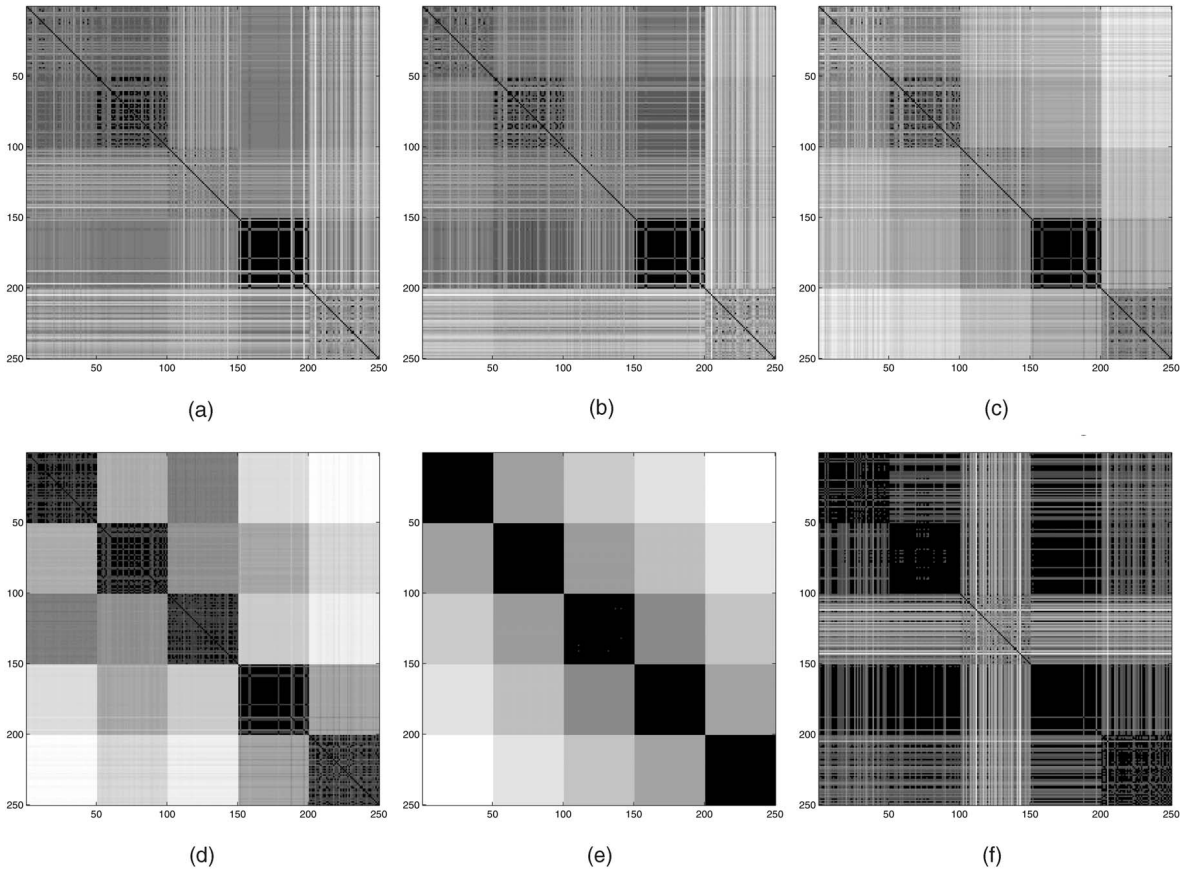


Fig. 8. Similarity matrices on synthesized data. (a) Trivial encoding. (b) Linear encoding. (c) Nested encoding. (d) Multilevel encoding (e) Pairwise encoding. (f) Tree-Edit distance.

TABLE 2
Average Similarity Values

	C_1	C_2	C_3	C_4	C_5
C_1	.067	.075	.032	.040	.023
C_2	.075	.203	.035	.047	.025
C_3	.032	.035	.040	.033	.019
C_4	.040	.047	.033	.668	.029
C_5	.023	.025	.019	.029	.037

(a)

	C_1	C_2	C_3	C_4	C_5
C_1	.095	.057	.043	.063	.019
C_2	.057	.185	.060	.108	.023
C_3	.043	.060	.056	.068	.025
C_4	.063	.108	.068	.683	.023
C_5	.019	.023	.025	.023	.039

(b)

	C_1	C_2	C_3	C_4	C_5
C_1	.164	.162	.098	.086	.058
C_2	.162	.379	.141	.111	.066
C_3	.098	.141	.202	.179	.091
C_4	.086	.111	.179	.840	.134
C_5	.058	.066	.091	.134	.267

(c)

	C_1	C_2	C_3	C_4	C_5
C_1	.965	.642	.815	.482	.393
C_2	.642	.968	.748	.659	.504
C_3	.815	.748	.962	.540	.431
C_4	.482	.659	.540	.978	.682
C_5	.393	.504	.431	.682	.945

(d)

	C_1	C_2	C_3	C_4	C_5
C_1	.898	.121	.066	.050	.036
C_2	.121	.947	.127	.078	.049
C_3	.066	.127	.841	.167	.073
C_4	.050	.078	.167	.994	.115
C_5	.036	.049	.073	.115	.952

(e)

	C_1	C_2	C_3	C_4	C_5
C_1	.286	.112	.028	.142	.060
C_2	.112	.579	.035	.199	.080
C_3	.028	.035	.035	.037	.025
C_4	.142	.199	.037	.777	.094
C_5	.060	.080	.025	.094	.215

(f)

(a) Trivial encoding. (b) Linear encoding. (c) Nested encoding. (d) Multilevel encoding. (e) Pairwise encoding. (f) Tree-Edit distance.

5.2 Experiments on Real Data

This section describes the results of experiments we performed on real XML documents extracted from different collections available on the Internet. The documents used belong to five main classes:

- `Astronomy`: a data set containing 217 documents extracted from an XML-based metadata repository, that describes an archive of publications owned by the *Astronomical Data Center* at NASA/GSFC (http://adc.gsfc.nasa.gov/).

```
<!DOCTYPE DTD5 [
   <!ELEMENT XML (m*,n) >
   <!ELEMENT m (q*) >
   <!ELEMENT q (x,y) >
   <!ELEMENT x ((a,c)|z*) >
   <!ELEMENT a EMPTY >
   <!ELEMENT c EMPTY >
   <!ELEMENT z EMPTY >
   <!ELEMENT n EMPTY >
]>
```

```
<!DOCTYPE DTD6 [
   <!ELEMENT XML (m*, n) >
   <!ELEMENT m (x) >
   <!ELEMENT x ((a,c)|z*) >
   <!ELEMENT a EMPTY >
   <!ELEMENT c EMPTY >
   <!ELEMENT z EMPTY >
   <!ELEMENT n EMPTY >
]>
```

```
<!DOCTYPE DTD7 [
   <!ELEMENT XML (m) >
   <!ELEMENT m (x,n) >
   <!ELEMENT x (z*) >
   <!ELEMENT z EMPTY >
   <!ELEMENT n EMPTY >
]>
```

Fig. 9. DTD5 and two similar DTDs sharing some definitions with it.

- `Forum`: a data set composed of 264 documents containing messages sent by users of a Web forum (http://static.userland.com/).
- `News`: a data set composed of 64 XML documents containing press news from all over the world, daily collected by *PR Web*, a company that provides free online press release distribution (http://www.prweb.com/rss.php).
- `Sigmod`: a data set composed of 51 XML documents containing issues of SIGMOD Record. Such documents were obtained from the XML version of the ACM SIGMOD Web site (http://www.dia.uniroma3.it/Araneus/Sigmod/).
- `Wrapper`: a data set composed of 53 XML documents representing wrapper programs for Web sites, obtained by means of the *LIXTO* system [8].

Like in the case of synthesized data sets, we expect to obtain a first, loose-grained separation of the documents according to their classes. Neverthless, the distributions of the tags within the documents are quite heterogeneous, due to the complexity of the DTDs associated with the classes, and to the semantic heterogeneity of the documents. In particular, wrapper programs may have substantially different forms, as a natural consequence of the structural differences existing among the various Web sites they have been built on. Hence, we expect a finer-grain analysis to be able of detecting such differences.

Table 5 summarizes the quality values obtained using the various encoding schemes. For measures e_k and q_k, we considered neighborhoods of size 51, i.e., the size of the smallest class in the data set. As we can see, the error rate is extremely low in all the encoding schemes, even though schemes exploiting context information exhibit better performances. In general, the results reflect the behavior analyzed in the previous section.

TABLE 3
Quality Indices on Synthesized Data in Presence of `DTD6` and `DTD7`

method/index	ε	$e_{k=50}$	$q_{k=50}$
Trivial	0.1800	0.3114	0.6147
Nested	0.1026	0.3714	0.6093
Linear	0.1956	0.2314	0.5926
Multilevel	0.0284	0.1514	0.8317
Pairwise Mult.	0.0145	0.0600	0.8997
Tree-Edit distance	0.1058	0.1171	0.8025

Fig. 11 shows the similarity matrices of three main encoding schemes, corresponding to three different strategies for taking into account the context where a tag appears. Once again, trivial encoding (Fig. 11a) exhibits the worst performance, although classes can be still separated.

The relevance of context information for estimating structural similarity is highlighted by the performances exhibited by other encoding schemes, which are based on the multilevel document encoding. Figs. 11b and 11c show the corresponding similarity matrices. In general, both the encodings exhibit low interclass similarity values, and quite high intraclass similarity values.

The only problematic class appears to be the `wrapper` one, which exhibits little homogeneity. This result is not so surprising, if the skewed nature of this set of documents is taken into account. Notwithstanding, the similarity matrices exhibit a neat separation from the other classes, and show homogeneos subgroups inside the class. This gives evidence of the effectiveness of the approach in detecting both interclass and intraclass dissimilarity. A further confirmation of the above hypothesis is given by a visual analysis of the DFT transforms of the signals produced by the Multilevel and Pairwise Multilevel schemes, plotted, respectively, in Figs. 12a and 12b. By looking at the Fourier transforms of the documents in the `wrapper` class, it is possible to recognize several clearly distinguishable groups of similar shapes.

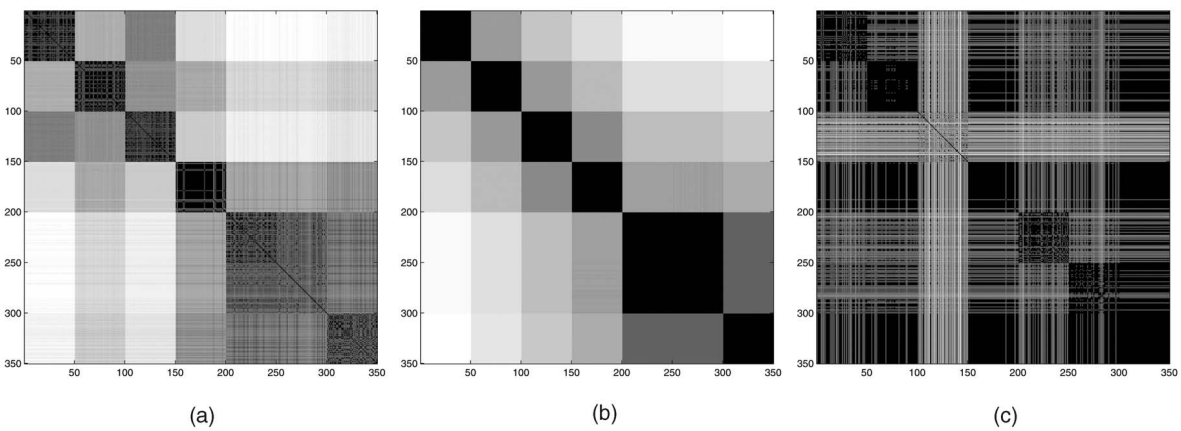


Fig. 10. Similarity matrices in presence of `DTD6` and `DTD7`. (a) Multilevel encoding. (b) Pairwise encoding. (c) Tree-edit distance.

TABLE 4
Average Similarity Values for DTD5, DTD6, and DTD7

	C_5	C_6	C_7
C_1	.393	.394	.417
C_2	.504	.504	.544
C_3	.431	.432	.460
C_4	.682	.680	.751
C_5	.945	.911	.847
C_6	.911	.898	.831
C_7	.847	.831	.936

(a)

	C_5	C_6	C_7
C_1	.037	.037	.035
C_2	.051	.051	.046
C_3	.076	.076	.067
C_4	.120	.120	.098
C_5	.956	.912	.350
C_6	.912	.919	.347
C_7	.350	.347	.996

(b)

	C_5	C_6	C_7
C_1	.060	.085	.148
C_2	.080	.101	.207
C_3	.025	.026	.038
C_4	.094	.013	.319
C_5	.215	.062	.114
C_6	.062	.281	.352
C_7	.114	.352	1.00

(c)

(a) Multilevel encoding. (b) Pairwise encoding. (c) Tree-edit distance.

TABLE 5
Quality Indices on Real Data

method/index	ε	e_k	q_k
Trivial	0.1071	0.0570	0.9270
Nested	0.1124	0.0493	0.9243
Linear	0.0996	0.0247	0.9508
Multilevel	0.0121	0.0015	0.9789
Pairwise Mult.	0.0144	0.0015	0.9795
Tree-Edit distance	0.0458	0.0123	0.9752

5.3 Final Remarks on the Encoding Schemes

We can summarize the differences exhibited by the various encoding schemes as follows: The Trivial scheme does not contextualize the current information and, hence, it has the poorest results. Linear, Nested, and Multilevel schemes introduce context information for each element. However, the contextualization introduced by the Linear encoding scheme is still little effective. Indeed, both trivial and linear encoding do not take into account context information provided by the nesting levels and, hence, are insensitive to such features. On the other hand, the contextualization introduced by the nested scheme seems to be more appropriate to enhance structural differences. This is also proven by the multilevel scheme, which provides the same information as the nested scheme, but in the document encoding instead of the tag encoding.

Linear, Nested, and Multilevel schemes provide mainly "backward" information: They consider, at each point of the XML tree, the path from the node to the root. In a sense, they assume a *depth-first visit* of the tree. The pairwise encoding scheme, on the contrary, provides also "forward" information: At each node, we look at the node that follows the current one. In principle, such a node can be a sibling of the current node: In such a situation, this corresponds to performing a more accurate analysis of the neighborhood of a tag. Combining the resulting information with the information provided by the multilevel encoding yields high quality similarity values.

Compared to the technique which makes use of a tree-edit distance, the above techniques are generally effective in measuring the structural similarity. Indeed, the performances achieved by the Trivial, Nested, and Linear

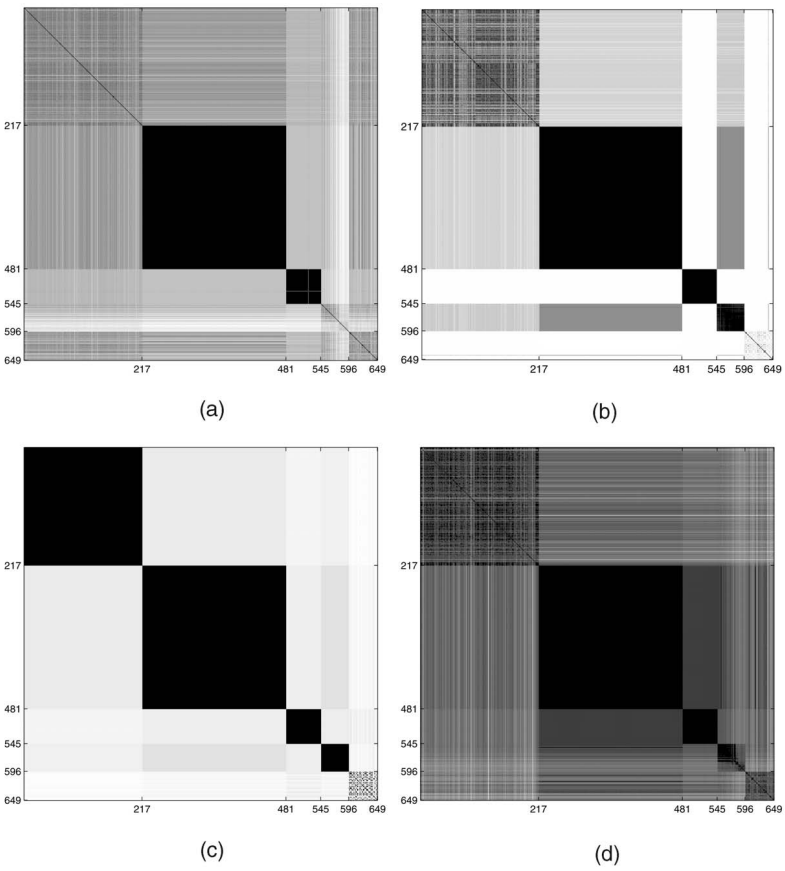


Fig. 11. Similarity matrices on real data. (a) Trivial encoding. (b) Multilevel encoding. (c) Pairwise encoding. (d) Tree-edit distance.

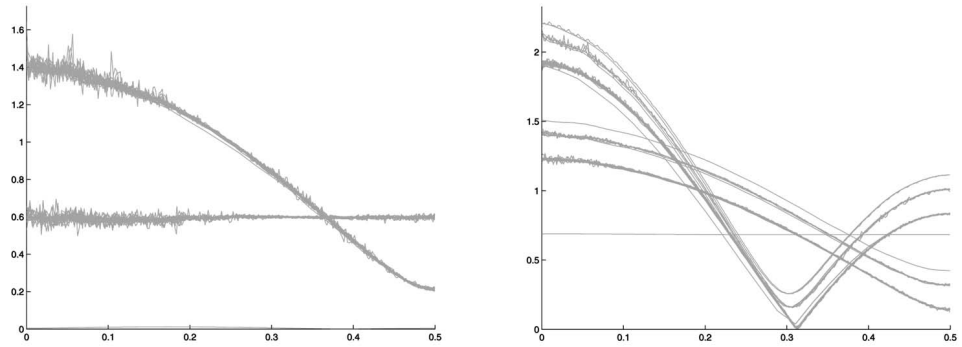


Fig. 12. Frequency spectra of multilevel and pairwise encodings on the wrapper class. (a) Multilevel and (b) pairwise.

encoding schemes are comparable, while schemes based on the the Multilevel document encoding function provide better results.

As a final remark, notice that all the proposed encoding schemes are scalable: The addition of a new document in the collection does not entail a reencoding of the entire collection. The claim is straightforward for the trivial, linear, and nested encoding schemes. Concerning the schemes based on the multilevel document encoding, observe that, by Theorem 1, these schemes are WSL no matter which value of B is adopted. That is, a new document can still be encoded by exploiting the B-base representation initially adopted. Notice that, since in principle the addition of new documents may introduce new tag names, the requirement $B = |tnames(D)| + 1$ may no longer be met, thus triggering a progressive worsening of the quality of results. Detailed studies show that performance worsening is only significant when $|tnames(D)|$ is far greater than B. In particular, when $\frac{B}{|tnames(D)|} \to 0$, the multilevel encoding scheme is expected to behave almost like the linear encoding scheme, whereas the pairwise multilevel scheme is expected to behave like a linear document encoding function combined with a modified (symmetric) version of the pairwise tag encoding.

6 Conclusions and Future Works

In this paper, we addressed the problem of detecting the structural similarity between XML documents. The technique we have proposed is mainly based on the idea of representing an XML document as a time series. Thereby, the structural similarity between two documents can be computed by exploiting the Discrete Fourier Transform of the associated signals. Experimental results showed the effectiveness of our approach, with particular regard to some of the encoding schemes defined in the paper, even when compared with different techniques based on tree-edit.

The current work is subject to further extensions, that we plan to address in future works:

- The structural similarity between documents can be refined exploiting information retrieval techniques. In particular, the combination of the distance measure we propose with traditional techniques, such as Jaccard or Cosine similarity, can be extremely profitable. Also, it is interesting to study the combination with document-content similarity. Documents' contents may contain significant semantic information and, hence, a combined measure can be defined by combining both structural and content similarity.
- In our current implementation, the tag encoding function does not take into account semantic similarities between tags. However, precision could be improved by exploiting tag similarity techniques, through, e.g., suitable ontologies.
- The encoding schemes can be further improved by analyzing different visiting strategies for the trees. An example alternative encoding can be defined by analyzing a tag according to its siblings (instead of its ancestors). In principle, a higher information content associated to a tag should guarantee a more refined translation of a document into a time series. Thus, we expect that the analysis of the corresponding signals allows us to detect finer differences.
- From an application viewpoint, a suitable application domain can be the extraction of information from the Web and its storage into an enterprise information system. Here, crawling and wrapping can be integrated by means of a document categorization technique based on structural similarity, and suitable for both classifying a set of pages with regard to a set of available wrappers and identifying new sets of similarly structured pages for which new wrappers can be defined.

Acknowledgments

This work is a revised, extended version of the paper that appeared in [28]. The authors are very grateful to Andy Nierman for providing them with the executables of the algorithm proposed in [19]. 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."

References

[1] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*. Morgan-Kaufman, 2000.
[2] L. Xyleme, "Xyleme: A Dynamic Warehouse for XML Data of the Web," *Proc. Int'l Database Eng. and Applications Symp. (IDEAS '01)*, pp. 3-7, 2001.

[3] H. Schöning and J. W Wäsch, "Tamino—An Internet Database System," *Proc. Seventh Int'l Conf. Extending Database Technology (EDBT '00),* pp. 383-387, 2000.
[4] J. McHugh and J. Widom, "Query Optimization for XML," *Proc. 25th Very Large Data Bases Conf. (VLDB '99),* pp. 315-326, 1999.
[5] J. McHugh et al., "Indexing Semistructured Data," Stanford Univ., technical report, 1998.
[6] W. Lian, D. Cheung, N. Mamoulis, and S.-M. Yiu, "An Efficient and Scalable Algorithm for Clustering XML Documents by Structure," *IEEE Trans. Knowledge and Data Eng.,* vol. 16, no. 1, Jan. 2004.
[7] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," *Proc. 27th Very Large Data Bases Conf. (VLDB '01),* pp. 109-118, 2001.
[8] R. Baumgartner, S. Flesca, and G. Gottlob, "Visual Web Information Extraction with Lixto," *Proc. 27th Very Large Data Bases Conf. (VLDB '01),* pp. 119-128, 2001.
[9] I. Muslea, S. Minton, and C. A. Knoblock, "Hierarchical Wrapper Induction for Semistructured Information Sources," *Autonomous Agents and Multi-Agent Systems,* vol. 4, nos. 1/2, pp. 93-114, 2001.
[10] G. Cobena, S. Abiteboul, and A. Marian, "Detecting Changes in XML Document," *Proc. 18th Int'l Conf. Data Eng. (ICDE '02),* pp. 41-52, 2002.
[11] Y. Wang, D. DeWitt, and J. Cai, "X-Diff: A Fast Change Detection Algorithm for XML Documents," *Proc. Int'l Conf. Data Eng. (ICDE '03),* pp. 519-530, 2003.
[12] S.S. Chawathe et al., "Change Detection in Hierarchically Structured Information," *Proc. ACM Conf. Management of Data (SIGMOD '96),* pp. 493-504, 1996.
[13] E. Bertino, G. Guerrini, and M. Mesiti, "A Matching Algorithm for Measuring the Structural Similarity between an XML Document and a DTD and Its Applications," *Information Systems,* vol. 29, no. 1, 2004.
[14] K. Zhang and D. Shasha, "Simple Fast Algorithms for the Editing Distance between Trees and Related Problems," *SIAM J. Computing,* vol. 18, no. 6, pp. 1245-1262, 1989.
[15] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient Similarity Search in Sequence Databases," *Proc. Fourth Int'l Conf. Foundations of Data Organization (FODO '93),* pp. 69-84, 1993.
[16] D. Rafiei and A. Mendelzon, "Efficient Retrieval of Similar Time Series," *Proc. Fifth Int'l Conf. of Foundations of Data Organization (FODO '98),* pp. 249-257, 1998.
[17] D. Goldin and P. Kanellakis, "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation," *Proc. First Int'l Conf. Constraint Programming (CP '95),* pp. 137-153, 1995.
[18] Z. Zhang, R. Li, S. Cao, and Y. Zhu, "Similarity Metric for XML Documents," *Proc. Workshop Knowledge Experience and Management (FGWM '03),* 2003, http://km.aifb.uni-karlsruhe.de/ws/LLWA/fgwm/index_eng_html.
[19] A. Nierman and H. Jagadish, "Evaluating Structural Similarity in XML Documents," *Proc. Fifth Int'l Workshop Web and Databases (WebDB '02),* 2002.
[20] H. Kashima and T. Koyanagi, "Kernels for Semi-Structured Data," *Proc. Int'l Conf. Machine Learning (ICML '02),* pp. 291-298, 2002.
[21] X. Yan and J. Han, "GSPAN: Graph-Based Substructure Pattern Mining," *Proc. IEEE Int'l Conf. Data Mining (ICDM '02),* pp. 721-724, 2002.
[22] T. Asai et al., "Efficient Substructure Discovery from Large Semi-Structured Data," *Proc. Second SIAM Int'l Conf. Data Mining (SDM '02),* 2002.
[23] B. Yi, H. Jagadish, and C. Faloutsos, "Efficient Retrieval of Similar Time Sequences under Time Warping," *Proc. 14th Int'l Conf. Data Eng. (ICDE '98),* pp. 201-208, 1998.
[24] B. Porat, *A Course in Digital Signal Processing.* Wiley, 1997.
[25] A. Oppenheim and R. Shafer, *Discrete-Time Signal Processing.* Prentice Hall, 1999.
[26] W. Press et al., *Numerical Recipes in C++.* Cambridge Univ. Press, 2001.
[27] M. Frigo and S.G. Johnson, "FFTW: An Adaptive Software Architecture for the FFT," *Proc. Int'l Conf. Acoustics, Speech, and Signal Processing,* vol. 3, pp. 1381-1384, 1998.
[28] S. Flesca et al., "Detecting Structural Similarities between XML Documents," *Proc. Fifth Int'l Workshop Web and Databases (WebDB 2002),* 2002.

Sergio Flesca received the computer engineering degree (summa cum laude) in 1996, and the PhD degree in computer science from the University of Calabria. He is currently an assistant professor at the University of Calabria, Italy. He has been a visiting fellow in the Computer Science Department, Vienna University of Technology. His current research interests include deductive databases, query languages, Web databases, and semistructured data management.

Giuseppe Manco received the degree in computer science (summa cum laude), in 1994, and the PhD degree in computer science from the University of Pisa. He is currently a senior researcher at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy, and contract professor at the University of Calabria, Italy. He has been a contract researcher at the CNUCE Institute in Pisa, Italy, and a visiting fellow at the CWI Institute in Amsterdam, Netherlands. His current research interests include deductive databases, knowledge discovery and data mining, Web databases, and semistructured data.

Elio Masciari received the PhD degree in computer science engineering from the University of Calabria, Italy. Currently, he is a researcher at the ICAR institute of the Italian National Research Council. His research activity is mainly focused on techniques for the analysis and mining of structured and unstructured data, XML query languages, and XML structural similarity.

Luigi Pontieri received the laurea degree in computer engineering in July 1996, and the PhD degree in system engineering and computer science in April 2001, from the University of Calabria. He is a senior researcher at the High Performance Computing and Networks Institute (ICAR-CNR) of the National Research Council of Italy, and a contract professor at the University of Calabria, Italy. His current research interests include information integration, data warehousing, and data mining.

Andrea Pugliese received the laurea degree (summa cum laude) in computer engineering from the University of Calabria, Italy. Since 2001, he has been a PhD degree student in systems and computer engineering in the DEIS Department at the University of Calabria. His current research interests include grid computing, databases, and semistructured data. Dr. Pugliese is a student member of the ACM and of the IEEE Computer Society.