Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 60 (2007) 222–234
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Exploiting structural similarity for effective Web information extraction

Sergio Flesca [a], Giuseppe Manco [b,*], Elio Masciari [b], Luigi Pontieri [b], Andrea Pugliese [a]

[a] *DEIS, Univ. della Calabria, Via P. Bucci 41/C, 87036 Rende, Italy*
[b] *ICAR-CNR, Via P. Bucci 41/C, 87036 Rende, Italy*

Available online 17 February 2006

Abstract

In this paper, we propose a classification technique for Web pages, based on the detection of structural similarities among semistructured documents, and devise an architecture exploiting such technique for the purpose of information extraction. The proposal significantly differs from standard methods based on graph-matching algorithms, and is based on the idea of representing the structure of a document as a time series in which each occurrence of a tag corresponds to an impulse. The degree of similarity between documents is then stated by analyzing the frequencies of the corresponding Fourier transform. Experiments on real data show the effectiveness of the proposed technique.

Keywords: Semistructured data; Wrapping; WWW tools

1. Introduction

The huge amount of information available on the Web offers new perspectives for on-line applications, which can be profitably exploited for various purposes. Information extraction agents can be developed for investigating and collecting data from Web sites, to exploit them for business purposes. Typical scenarios include, e.g., competitors monitoring, automatic news filtering, product finding and price comparison, etc. In order to make Web information really usable, it is convenient to manage it through enterprise information systems. When it is a priori known which pages the desired information must be collected from, it is possible to use ad hoc HTML to XML wrappers [1–4], to extract information from sets of HTML pages having a similar structure. The extracted information, encoded in XML, can be exploited by the system to help decision makers, or simply to offer new services to customers. Thus, the use of HTML wrappers allows

* Corresponding author. Tel.: +39 0984 831728; fax: +39 0984 839054.
E-mail addresses: flesca@deis.unical.it (S. Flesca), manco@icar.cnr.it, giuseppe.manco@icar.cnr.it (G. Manco), masciari@icar.cnr.it (E. Masciari), pontieri@icar.cnr.it (L. Pontieri), apugliese@deis.unical.it (A. Pugliese).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2006.01.001

for making high-quality semistructured data available for various purposes, with the major advantage of the low human effort needed to extract the desired information. Provided with several sets of similarly structured HTML pages, the wrapper designer must generate an HTML/XML wrapper for each set. Once these wrappers have been generated, they continuously extract information, and the extraction process must be monitored only for handling possible extraction exceptions. A main issue arises when it is not a priori known where interesting information is located, so it is necessary to crawl the Web [5–7]. In this case, pages collected by crawlers are not necessarily structured in similar ways and, as a consequence, they cannot be automatically handled by wrappers. Moreover, most of the currently available tools only allow the extraction of textual information; a relevant human effort is needed to restructure the available data and detect significant information.

A main problem when characterizing the structure of Web documents is the need to refer to a precise application context. Indeed, even if tags are the basis for detecting the structure of a document, they only express its syntactic structure, disregarding the semantics of the contained data. Finding heterogeneous representations of semantically similar information is a very common situation in the Web: different tags and different combinations could be used to represent similar information sources. Worst, similar markup tags could be used to structure different information sources. Such a problem is even more critical when documents are coded in HTML, a language more suitable for presentation issues than information coding since it provides little semantic expressiveness.

However, to the best of our knowledge, most wrapper languages [3,2,4] only use the syntactic structure of Web pages to define how to extract information, while only a few [1] have (very limited) semantic facilities. Thus, we mainly concentrate on syntactic similarity, as defined by the formatting structure of HTML. Despite the limited number of HTML tag names and the lack of explicit semantics, we believe (and the experimental results confirm this) that such a simple approach can be successful in recognizing homogeneous groups of data-intensive HTML documents. As a matter of fact, we experienced that recognizing syntactically homogeneous groups of documents is sufficient for inducing and selecting the most suitable wrappers. Indeed, a wrapper is able to process only pages that exhibit similar syntactic structures, at least in the portions containing the relevant data to be extracted. Obviously, other portions of the pages, for instance those containing advertisements, can exhibit very different syntactic structures; however, the irrelevant parts of the pages are usually smaller. Furthermore, irrelevant parts are likely to have different structure even in unrelated pages.

The technique proposed in this paper represents the structure of a document as a tree of elements. The tagging structure in well-formed XML documents naturally induces such a kind of representation; HTML pages on the Web are instead often ill-formed, e.g., browsers do not always force the use of end tags. However, most HTML parsers are able to parse ill-formed documents and still represent them as trees. In order to improve the classification process of Web pages data mining techniques can be profitably exploited to classify documents made available by a Web crawler. Indeed, the capability of automatically recognizing whether the contents of a Web source can be suitably processed by an available wrapper facilitates the task of extracting relevant information. Moreover, the capability of automatically detecting and collecting similarly structured pages which do not fit to any available wrapper model, but which may, in principle, contain significant information, can help the expert in building ad hoc wrappers for them.

1.1. Main contribution

In this paper, we address the problem of integrating and enhancing crawling and wrapping systems in order to avoid (or reduce) the human effort necessary to deal with the potentially huge amount of pages found by crawlers. The contribution of this work is twofold:

(1) we present a technique for HTML document categorization, which allows for both classifying found pages w.r.t. the a of available wrapper, and identifying new sets of similarly structured pages, for which new wrappers can be defined;
(2) we propose an architecture for the extraction of information from the Web and its storage into an enterprise information system that exploits the categorization technique.

The proposed architecture is flexible (any wrapper or crawler can be exploited, as shown in the next section) and adopts an efficient and effective technique for measuring the structural similarity between semi-structured documents. This technique represents the structure of a document as a time series in which each occurrence of a tag corresponds to a given impulse. By analyzing the frequencies of the corresponding Fourier transform, we state the degree of (structural) similarity between documents. The efficiency of this approach (it works in $O(N\log(N))$) is compelling when compared to other approaches defined in the literature [8,9]. Moreover, the technique is particularly attractive for its effectiveness. As a matter of fact, the use of the Fourier transform to check similarities among time series is not completely new (see, e.g., [10]), and was proven successful. The main contribution of our approach is the systematic development of effective encoding strategies for Web documents, in a way that makes the use of the Fourier transform extremely profitable.

2. Wrapping and crawling the Web through structural document categorization

The possibility of automatically processing Web pages allows reducing the costs of extracting relevant information. In this section we sketch an architecture where crawling and wrapping systems are integrated to reduce the human efforts needed to extract semistructured information from the Web. The architecture exploits document categorization algorithms to detect structurally similar pages and to select (or build) wrapper programs suitable to their processing.

The proposed architecture, shown in Fig. 1, is devoted to the extraction of interesting information from the Web and to its storage into an enterprise information system. The module which is responsible for finding interesting data is called *Web crawler*; it continuously crawls the Web yielding new potentially interesting pages. Once such pages have been found, the *page classifier* module classifies them w.r.t. the available wrapper programs. *Wrappers* are software modules that convert data that is implicitly contained in Web documents into semi-structured data. Each class of similarly structured pages is then forwarded to a wrapper program, that translates and stores the information they contain.

Obviously, not all the pages found by a crawler can be properly classified. In the proposed architecture, information from unclassified pages can be manually extracted, but such pages can be also used to build new wrappers. To this purpose, this set of pages is forwarded to the wrapper designer, that processes them using the *wrapper designer suite*. During the wrapper design process, document categorization techniques are exploited to automatically identify clusters of similarly-structured pages, that can be handled by the same wrapper. The output of this process is a new set of wrapper definitions that can be used both for classifying new interesting pages and for automatically extracting information.

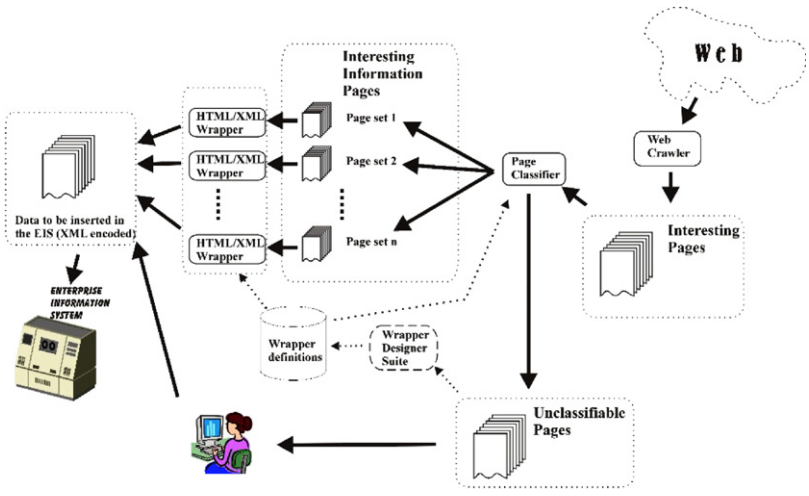


Fig. 1. Architecture of the information extraction system.

Notice that, in the proposed architecture, the processes of crawling, classifying and wrapping Web pages are kept separate, and no particular assumption is made about them. Therefore, it is possible to integrate any kind of crawling technique [5–7] and wrapper generation system [3,1,2,4].

The complexity of wrapper generation systems is strongly related to the level of structuring of the Web pages they deal with. Usually, a wrapper is designed for a specific set of Web pages exhibiting inherently similar features. Such features define the context in which the relevant data to extract is located. A typical example is a set of HTML pages containing details (e.g., price, description, picture, etc.) about a given set of products to be purchased online. If such pages refer to the same product category, or if they are extracted from the same service provider, it is likely that the information they provide appears structured in similar ways (e.g., each product is represented as a row in a table, whose first cell contains either the product name or its picture). Thus, in order to design a wrapper for extracting pricing data, all the pages under consideration must have a similar structure. The page classifier and the wrapper design suite are mainly based on structural document categorization, which can be summarized as follows.

Let $\mathbf{w}$ be a wrapper, and $\mathscr{T}_{\mathbf{w}} = \{p_1, \ldots, p_m\}$ be the set of Web pages used to generate $\mathbf{w}$ (*training set*). For a well-defined wrapper, it is assumed that the structural similarity between each pair p_i and p_j is high. The tasks performed by the page classifier and the wrapper generation suite can be described as follows. Given a set $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ of available wrappers, a new page p is associated with $\mathscr{T}_{\mathbf{w}_i}$ if (a) the structural similarity between p and each $q \in \mathscr{T}_{\mathbf{w}_i}$ is considered acceptable, i.e., it is higher than a given threshold, and (b) no other set $\mathscr{T}_{\mathbf{w}_j}$ exhibits a higher structural similarity. If no $\mathbf{w}_i$ exists such that p can be associated with $\mathbf{w}_i$, p is labelled as *unclassified*. A set $\mathscr{U}$ of unclassified pages is worth further consideration if it is possible to define a partition of $\mathscr{U}$ in k clusters, where each cluster $\mathscr{C}_i$ can be exploited as a training set for a new wrapper $\mathbf{w}_{\mathscr{C}_i}$. Notice that the first task can be efficiently accomplished by means of k-nearest neighbor techniques, whereas the second one is mainly a clustering problem, for which similarity-based approaches can be defined. Therefore, a major issue is a proper definition and evaluation of the similarity among Web documents according to their structure.

3. Detecting structural similarity among HTML documents

Intuitively, two documents are said to have a similar structure if they correspond in the type of elements they contain and in the way these elements are combined. Observe that, even if it is easy to detect whether the structure of two documents is exactly the same, this information is rarely useful for our aims. Indeed we would like to quantify the similarity between the structures of two documents, also emphasizing the differences that are more relevant. For instance, we consider as similar two documents that have the same features with different regularities. That is, two HTML documents are similar if it is possible to identify similar substructures, even if they appear with different frequencies.

Much attention has been devoted to the problem of detecting structural similarity between complex objects. Several methods for detecting XML similarity [8,11] have been recently proposed. All these methods are based on the concept of *edit distance*, and employ graph-matching algorithms to calculate (minimum cost) edit scripts that 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 this section, we propose a different approach, which is essentially based on the idea of associating each document with a time series representing its structure, and checking the structural similarity by looking at the corresponding time series. As we shall see, this approach is both efficient and effective. The approach was initially designed to detect structural similarities between XML documents [12]; when dealing with HTML documents, different issues arise that need to be tackled. In the following, we briefly recall our technique for encoding XML documents and measuring their similarity. Further details on the encoding techniques for XML and on the associated similarity measures can be found in [12].

3.1. Encoding XML documents

An XML document is structured as a tree of elements, where each element embeds a piece of information. In principle, we would like to flatten the tree structure into a time series which perfectly summarizes the

relevant features of the original document. Assuring an injective flattening is not sufficient here: since we directly compare two time series, we would like to give greater weight to the more relevant structural characteristics.

We begin by fixing some notation. Given an XML document d, we denote by $tags(d)$ the *tag set* of the document d, i.e., the set of all the tags occurring within d; moreover, $tnames(d)$ denotes the set of all the distinct tag names appearing in d. Furthermore, for an element el of d, we denote by el_s the starting tag of el and by el_e the ending tag of el. Given a tag t with tag name tn, the *type* of t is its tag name tn if t is a start tag or $/tn$ if t is an end tag. The *skeleton* of d (denoted by $sk(d)$) is the sequence of tags appearing within d, the sequence $[t_0, t_1, \ldots, t_n]$ such that $t_i \in sk(d) \iff t_i \in tags(d)$ and t_i precedes t_j within d if and only if $i < j$. Intuitively, the skeleton of an XML document represents a description of the sole document structure. For a tag $t \in sk(d)$, we define $nest_d(t)$ as the set of the start tags el_s in d occurring before t and for which there is no end tag el_e matching el_s and appearing before t. The *path name* of an element el is defined as the concatenation of the names of the element that enclose it in d. We also denote by l_t the nesting level of the tag t, i.e., $l_t = |nest_d(t)|$. Finally, for a given set D of documents, $maxdepth(D)$ denotes the maximum nesting level of tags appearing in a document $d \in D$.

We define a document encoding as a combination of a *tag encoding function* and a *document encoding function*. Intuitively, a tag encoding function provides a numerical encoding of a tag by looking at its "internal" (local) properties, whereas a document encoding function aims at encoding a sequence of tags by looking at its overall features.

3.1.1. Tag encoding functions

Given a set D of XML documents, a function γ from $tags(D)$ to $\mathbb{R}$ is a *tag encoding function* for D. We can assign a number n to each tag in several different ways: for instance, by generating it randomly, or using a hash function. Obviously, a good tag encoding function should at least ensure to be injective w.r.t. tag names. The encoding functions presented in the following differ in the way in which they capture information about each tag's neighbors.

The simplest tag encoding function we consider is named *Direct tag encoding* (γ_d). Given a set D of XML documents, we build a sequence of distinct tag names $[tn_1, tn_2, \ldots, tn_k]$ by considering a (randomly chosen) linear order on $tnames(D)$. Given an element el, the direct encoding simply associates each start tag el_s with the position n of the tag name tn of el in the sequence ($\gamma_d(el_s) = n$).

A simple extension of the direct encoding consists in assigning a value to each tag by relating such value to the subsequent one. We denote by $cpairs(D)$ the pairs of types of tags $\langle tn_i, tn_{i+1}\rangle$ such that there exists a pair of tags $\langle t_i, t_{i+1}\rangle$, resp. of type tn_i, tn_{i+1}, that appear consecutively in a document $d \in D$. We associate an integer number $P_{\langle tn_i, tn_{i+1}\rangle}$ with each pair of types of tags $\langle tn_i, tn_{i+1}\rangle$ by considering a randomly chosen linear order on $cpairs(D)$. Given a pair of tags t_i, t_{i+1} (resp. of type tn_i, tn_{i+1}) appearing consecutively in a document d, the *Pairwise tag encoding* function ($\gamma_{pw}(t_i)$) associates with t_i the number $P_{\langle tn_i, tn_{i+1}\rangle}$.

Finally, we can encode a tag on the basis of its path name. Consider a set of documents D, and let $pnames(D)$ be the set of path names associated with the elements appearing in a document $d \in D$. Again, we use a sequence of path names $[pn_1, pn_2, \ldots, pn_k]$ obtained by considering a randomly generated linear order on $pnames(D)$, and we associate each path name pn_i with its position i (denoted as $pos(pn_i)$) in the sequence. Given a start tag el_s appearing in a document d with corresponding path name pn, the *Nested tag encoding* function $\gamma_{pt}(t)$ is defined by associating el_s with $pos(pn)$.

3.1.2. Document encoding functions

Let D be a set of XML documents. A document encoding is a function enc that associates each $d \in D$ with a sequence of real numbers, i.e., $enc(d) = h_0, h_1, \ldots, h_n$. In the following we assume a set of XML documents D, a document $d \in D$ with $sk(d) = [t_0, \ldots, t_n]$ and a tag encoding function γ.

A *trivial encoding* of d ($tenc(d)$) is a sequence $[S_0, S_1, \ldots, S_n]$, where $S_i = \gamma(t_i)$. This encoding simply applies γ to each tag appearing in the skeleton of the document.

A *linear encoding* of d ($lenc(d)$) is a sequence $[S_0, S_1, \ldots, S_n]$, where $S_0 = \gamma(t_0)$ and $S_i = \sum_{k \leqslant i} \gamma(t_k)$. Here, each element of the time series associated with a document should encode the information corresponding to more than a single tag. Indeed, it computes a linear combination of the tag encodings.

```
<xml>
    <book year="1997">
        <title> A First Course in Database Systems </title>
        <author> Ullman </author>
        <author> Widom  </author>
        <publisher> Prentice-Hall </publisher>
    </book>
</xml>
```

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. 2. Tag and document encodings.

A *multilevel encoding* of d ($mlenc(d)$) is a sequence $[S_0, S_1, \ldots, S_n]$, where $S_i = \gamma(t_i) \times B^{maxdepth(D)-l_{t_i}} + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times B^{maxdepth(D)-l_{t_j}}$. This encoding function assumes that the contribution of a tag t to the document encoding must depend on the nesting level of the tag. Intuitively, we encode t according to a basis B which takes into account both its nesting level and the path from the root to t. We usually set $B = |tnames(D)| + 1$ to avoid "mixing" the contributions of different nesting levels.

Fig. 2(bottom) shows the application of the above encodings to the simple XML document in Fig. 2(top). It should be noted that the alternative encodings proposed are not competitors in principle, but may prove useful for different types of documents.

3.2. Similarity measures

Encoding a document provides a particular view of the structure of a document, that can be seen as a time series. The series is built through a preorder visit of the tree structure of the document, starting from an initial time t_0 and visiting each tag after a fixed time interval Δ. For each tag, there is an impulse whose amplitude is determined by the document encoding function; as a result of this physical simulation, the visit of the document produces a signal $h_d(t)$ that varies in the time interval $[t_0, t_0 + N\Delta)$.

Comparing two such signals can be as difficult as comparing the original documents. Indeed, (i) comparing documents having different lengths requires costly resizing and alignment operations, and (ii) stretching (or narrowing) signals is not a suitable solution since such operations heavily affect the corresponding document structure.

These drawbacks can be avoided if the signals are compared by examining their *Discrete Fourier Transforms* (DFT) [13,14], which reveal much about the distribution and relevance of signal frequencies. Given a document d, we denote as DFT($enc(d)$) the discrete Fourier transform of the time series resulting from its encoding. To compare two documents, we consider the difference in the magnitude of the corresponding frequency components, that allows (i) to abstract from the length of the document, and (ii) to know whether a

given subsequence (representing a subtree in the XML document) exhibits a certain regularity, no matter where it is located within the signal. Let d_1, d_2 be XML documents, and *enc* be a document encoding, such that $h_1 = enc(d_1)$ and $h_2 = enc(d_2)$. We define the *Discrete Fourier Transform distance* of the documents as an approximation of the squared difference of the magnitudes of the two signals:

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

where $\mathtt{D}\widetilde{\mathtt{F}}\mathtt{T}$ 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). Interpolation in frequency domain is here exploited to allow for comparing sequences with different lengths; this can be seen as an efficient method to approximate a zero-padding [13] operation on the sequences.

It is worth noticing that, when comparing two documents with length N, the proposed techniques require $O(N \log N)$ time, since computing DFT is $O(N \log N)$; this is compelling w.r.t. other approaches, such as graph-based techniques, working in at least $O(N^2)$ time.

3.3. From XML to HTML (and back)

Although the above described technique can be directly applied to HTML documents, two main issues may arise when dealing with HTML. First of all, HTML documents (as found on the Web) are not necessarily well-formed. Typically, errors which may appear are the absence of a start/end-tag, and the lack of nesting among tags (which prevents from representing the HTML document as a tree). We solved the above problem by a suitable pre-processing of a document, in a way that makes it XML-compliant. The absence of start/end tag can be easily sanitized, by suitably inserting the missing tag within the document. The lack of nesting among tags, which was solved by repeatedly applying following heuristics, until a well-formed document was obtained:

- if tag a is opened but not closed before the closure of a further tag b, then the latter is moved after the first next closure downward the document,
- if by the converse a is closed but not opened before the opening of b, then the latter is moved before the first preceding opening upward the document.

The second issue is that HTML is a language specifically designed to address presentation issues, rather than semantic issues (like in XML). Thus, HTML provides little expressiveness from a semantic point of view. In the following we concentrate mainly on syntactic similarity, as defined by the formatting structure of HTML tags. Although this may look like a limitation, the experimental analysis shown in the next section will show the effectiveness of this approach.

4. Experimental results

In this section, we present some experiments we performed to assess the effectiveness of the proposed approaches in measuring the structural similarity among HTML documents.

The direct result of each test is a similarity matrix S representing the degree of structural similarity for each pair of documents. The evaluation of the results relies on some a priori knowledge about the data sets used. In fact, the HTML documents were gathered from different sources in the Internet; a group of documents coming from the same source is said to be a *class*. We used about 400 documents, belonging to 16 classes, which can be grouped into four high-level *categories* (each corresponding to a distinct application domain): (1) E-commerce, containing 102 HTML documents and consisting of four classes, named E_1, E_2, E_3 and E_4, corresponding to e-commerce Web sites; (2) Museums, 96 documents in four classes, named M_1, M_2, M_3 and M_4, corresponding to the Web sites of four museums; (3) Newspapers, 111 documents grouped in four classes, named N_1, N_2, N_3, N_4, corresponding to the Web sites of four newspapers; (4) Universities, 94

documents, grouped in four classes, named U_1, U_2, U_3, U_4, taken from corresponding to the Web sites of four universities.

As said before, we refer to a combination of a tag- and a document-encoding function as to an *encoding scheme*. In particular, we consider five schemes: *Trivial*, which combines trivial document encoding with direct tag encoding, *Linear* (linear document and direct tag encodings), *Nested* (trivial document and nested tag encodings), *Multilevel* (multilevel document and direct tag encodings), *Pairwise* (multilevel document and pairwise tag encodings).

4.1. Evaluating similarity measures

In this section, we detail the similarity measures obtained over the data set `Newspapers` described in previous section.

In order to give an immediate feeling of the similarities, we plot the similarity matrix as an image, using a color scale where darker pixels correspond to higher similarity values. Note that, the blocks on the diagonal of the matrix correspond to intra-class similarities, whereas the blocks outside the diagonal represent the inter-class similarities. Obviously, "perfect" results are represented by a similarity matrix having black blocks on the diagonal and white blocks outside.

Average values of intra- and inter-class similarities are instead summarized into a matrix CS, with the objective of supporting a simple quantitative analysis. In particular, given a set of documents belonging to n prior classes and a similarity matrix S defined on those documents, an $n \times n$ matrix CS is produced, where each element is computed as follows:

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

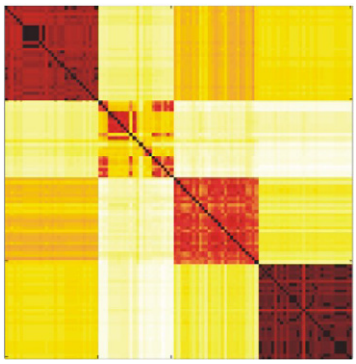
At a first glance at Fig. 3(left), the trivial scheme seems not able to suitably distinguish the classes. In fact, while classes N_1 and N_4 are clearly recognized, the other ones show a quite low internal similarity.

However, the quantitative results shown in Fig. 3(right) reveal that the trivial scheme performs surprisingly well. Indeed, for all classes, the intra-class similarity values are sufficiently higher than the inter-class ones, thus allowing for separating all classes from one another. In particular, adopting an iterative approach, classes N_1 and N_4 can be first extracted, which exhibit the highest intra-class similarity. After the removal of these classes, the average similarities lower of about an order of magnitude, allowing the separation of class N_3. Finally, the second class, with the lowest intra-class similarity, can be identified.

The results shown in Fig. 4(right) demonstrate a slight improvement in recognizing the prior classes which the linear scheme gains with respect to the trivial one. As in the previous case, the last class is the most homogeneous, while the second one exhibits the minimum average intra-class similarity. The good performance of

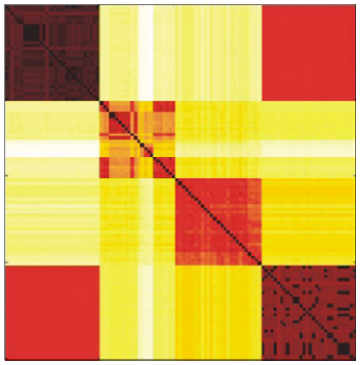
	N_1	N_2	N_3	N_4
N_1	0.0608	0.0039	0.0068	0.0094
N_2	0.0039	0.0053	0.0045	0.0039
N_3	0.0068	0.0045	0.0095	0.0065
N_4	0.0094	0.0039	0.0065	0.1536

Fig. 3. Similarity matrix and average similarities for the trivial encoding scheme.



	N_1	N_2	N_3	N_4
N_1	0.0650	0.0047	0.0064	0.0056
N_2	0.0047	0.0076	0.0044	0.0042
N_3	0.0064	0.0044	0.0108	0.0051
N_4	0.0056	0.0042	0.0051	0.1779

Fig. 4. Similarity matrix and average similarities for the linear encoding scheme.



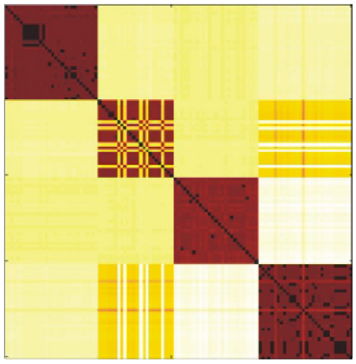
	N_1	N_2	N_3	N_4
N_1	0.0935	0.0023	0.0025	0.0060
N_2	0.0023	0.0055	0.0027	0.0025
N_3	0.0025	0.0027	0.0057	0.0030
N_4	0.0060	0.0025	0.0030	0.1518

Fig. 5. Similarity matrix and average similarities for the nested encoding scheme.

linear scheme is supported by the graphical representation of the similarity matrix in Fig. 4(left), where blocks corresponding to the intra-class similarities can be clearly individuated.

The results in Fig. 5 prove the ability of the nested scheme to adequately evaluate structural similarity over the data set considered. Classes N_1 and N_4 can again be neatly recognized, while the other ones show lower similarity. It is worth noticing that this scheme is able to still reduce the inter-class similarities w.r.t. the intra-class ones, thus allowing for better distinguishing the classes.

The results produced by the multilevel scheme appreciably differ from those presented so far, mainly because the average similarities in Fig. 6(right) are rather close to the maximum value allowed. These results



	N_1	N_2	N_3	N_4
N_1	0.9993	0.9930	0.9932	0.9808
N_2	0.9930	0.9969	0.9932	0.9940
N_3	0.9932	0.9932	0.9990	0.9921
N_4	0.9808	0.9940	0.9921	0.9993

Fig. 6. Similarity matrix and average similarities for the multilevel encoding scheme.

	N_1	N_2	N_3	N_4
N_1	1.0000	0.9997	0.9998	0.9997
N_2	0.9997	0.9998	0.9996	0.9997
N_3	0.9998	0.9996	0.9999	0.9995
N_4	0.9997	0.9995	0.9996	0.9999

Fig. 7. Similarity matrix and average similarities for the pairwise encoding scheme.

can be explained by taking into account the nature of the names and positions of tags inside HTML documents, as well as the strategy of the multilevel document encoding function, which associates each tag with a linear combination of the encodings of the tags enclosing it. In particular, the more external is the tag, the higher is the associated weight, computed by means of a function which exponentially depends on the nesting levels. On the other side, the external levels in HTML documents usually contain a few tag names, such as `html`, `head`, and `body`. Therefore, when the multilevel scheme is applied to HTML documents, the associated time series tend to have roughly similar overall shapes.

In spite of this phenomenon, the quality measures shown in Fig. 6(right) prove that the performances of the multilevel scheme are good. Indeed, we can observe that the intra-class similarities are yet higher than the inter-class ones and allow for well separating the classes, as it is confirmed by Fig. 6(left) (produced by applying a suitable distortion effect to the color scale in order to emphasize the differences).

The similarity matrix in Fig. 7(left) looks rather similar to the one produced by the multilevel encoding scheme. Furthermore, high similarities among most of the documents can still be noticed, even when they belong to different classes. This behavior is due to the combination of the multilevel document encoding with the pairwise tag encoding, that considers all the pairs of consecutive tags, and is therefore prone to produce a high number of tag codes, emphasizing the weight differences of tags at different levels. This combination hence makes the similarity between two documents essentially depend on how they appear in their most external elements, which appear to be nearly invariant in HTML documents. However, even in this case, the results are globally satisfactory since all the classes can be distinguished from one another, in spite of the high inter-classes similarities and the quite low homogeneity of class N_2, which yet exhibits the minimum average intra-class similarity.

4.2. Quality measures

In the following experiments, we aim at assessing the quality of the similarity measures obtained. A natural quality measure can be the error rate of a k-nearest neighbor classifier. Indeed, for each document, we can measure whether the dominant class of the k most similar elements allows to correctly predict the actual class of the document, and consider the total number of documents correctly predicted as a measure of effectiveness. This 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. Therefore, we define q_k as the average percentage of documents in the k-neighborhood of a generic document which belong to the same class of the document:

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

where N is the total number of documents, $Cl(i)$ represents the class associated with the ith document, and $F_{k(i)}$, is the set of k documents having the lowest distances from d_i according to the similarity measure. In principle, a nearest neighbor classifier has good performances 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 kNN classifier less sensitive to increases in the number of neighbors considered.

The sensitivity of the similarity measure can also be 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 one, but z is more similar to x than y is. We denote this probability by $\varepsilon(S)$, which is estimated as follows:

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

where δ_S is 1 if $S(i,j) < S(i,k)$, and 0 otherwise.

Tables 1 and 2 summarize the quality values obtained. To compute q_k, in each test we chose a neighborhood size equal to the minimum class cardinality.

These results are very interesting as a whole, as they prove the effectiveness of our Fourier-based similarity analysis on HTML documents, whatever encoding scheme is chosen. In particular, as proved by a closer look at the results, all the considered classes are recognized as sufficiently homogeneous from a structural point of view, i.e., the intra-class similarities are generally higher than inter-class ones. These good performances are rather surprising, considered that HTML tag names belong to a rather small set of predefined terms, and do not express semantics.

A comparative analysis of the encoding strategies is not straightforward, due to the very low differences in the quality values shown in Tables 1 and 2. Certainly, the encoding schemes based on the multilevel document encoding function do not exhibit as brilliant results as they do over pure XML documents [15]: in a few cases, they perform worse than other encoding techniques. This behavior is essentially due to the fact that the multilevel document encoding function mainly focuses on structural differences at more external levels, which tend to be rather similar in HTML documents. On the basis of this observation, a straightforward improvement could be that of decreasing their dependence on the external levels of the documents structure.

On the contrary, very good results are obtained by the nested and, surprisingly enough, by the rather simple trivial and linear schemes. In particular, the nested scheme tends to perform best when applied to classes from

Table 1
Error ε for several data sets and methods

Test	Document classes	Trivial	Linear	Nested	Multilevel	Pairwise
1	E_1, E_2, E_3, E_4	0.0114	0.0379	0.0067	0.0212	0.0329
2	M_1, M_2, M_3, M_4	0.0442	0.0314	0	0.1218	0.0829
3	N_1, N_2, N_3, N_4	0.0351	0.0021	0	0.0142	0.0430
4	U_1, U_2, U_3, U_4	0.0796	0.0375	0.0515	0.0498	0.0413
5	E_2, M_3, N_2, U_2	0.0148	0.0053	0.0002	0.0167	0.0687
6	E_3, M_2, N_3, U_3	0.0251	0.0165	0.0552	0.0436	0.0349
7	E_4, M_4, N_4, U_1	0.0002	0.0038	0.0318	0.0075	0.0160

Table 2
Quality measure q_k for several data sets and methods

Test	Document classes	Trivial	Linear	Nested	Multilevel	Pairwise
1	E_1, E_2, E_3, E_4	0.9768	0.9162	0.9808	0.9666	0.9643
2	M_1, M_2, M_3, M_4	0.9501	0.9518	1	0.8300	0.9211
3	N_1, N_2, N_3, N_4	0.9546	0.9914	1	0.9643	0.9287
4	U_1, U_2, U_3, U_4	0.9064	0.9665	0.9144	0.9106	0.9154
5	E_2, M_3, N_2, U_2	0.9640	0.9798	0.9988	0.9786	0.8871
6	E_3, M_2, N_3, U_3	0.9803	0.9857	0.9529	0.9265	0.9578
7	E_4, M_4, N_4, U_1	1	0.9924	0.9271	0.9710	0.9595

the same category, whereas in other cases the trivial and linear schemes obtain appreciably good results. The dissimilar behaviors of these encoding schemes mainly depends on the different ways they deal with the context of tags.

We can further observe that when the examined dataset contains classes coming from very different categories (applicative or semantic contexts), it is likely to have some tag names characterizing each category, i.e., being very frequent in a category and rare in the others. This is the case, e.g., of the tag names `table`, `tr`, `td`, `form`, `input`, and `option` for the e-commerce category. In such a situation, a simple recognition of characteristic tag names or their repetitive sequences, as the one carried out by both the trivial and the linear encodings, may be more profitable than the finer encoding strategies adopted by the nested, multilevel and pairwise schemes. In fact, the tag context information these strategies encode is not so useful in such cases and, rather, it introduces an higher detail level which, acting as a sort of noise, could make the massive presence of frequent and distinctive tag names less evident. In particular, while in the nested, multilevel and pairwise schemes different occurrences of such tag names can be associated with different codes in the resulting time series, they would be encoded in almost the same way by the trivial and linear schemes. The Fourier-based similarity measure applied to the time series scales down the differences between the two latter approaches.

5. Conclusions

In this paper, we proposed a technique for categorization of HTML documents based on Fourier transform. Also, an architecture for integrating crawling and wrapping of Web pages based on such technique is presented. The proposed technique has been tested by performing an huge number of experiments. Indeed, in specific application domains, the technique has been proven effective in collecting homogeneous structures for wrapper induction. As a future work the proposed structural similarity measure could be improved by exploiting information retrieval techniques, such as the traditional text processing techniques [16].

References

[1] R. Baumgartner, S. Flesca, G. Gottlob, Visual Web information extraction with Lixto, in: Proc. 27th VLDB Conf., 2001.
[2] I. Muslea, S. Minton, C. Knoblock, Hierarchical wrapper induction for semistructured information sources, Autonomous Agents Multi-Agent Systems 4 (2001) 93–114.
[3] V. Crescenzi, G. Mecca, P. Merialdo, RoadRunner: towards automatic data extraction from large web sites, in: Proc. of the 27th VLDB Conf., 2001.
[4] N. Kusmerick, Wrapper induction: efficiency and expressiveness, Artif. Intel. J. 118 (1–2) (2000) 15–68.
[5] T. Kistler, H. Marais, WebL—A programming language for the web, Comput. Networks ISDN Systems 30 (1–7) (1998) 259–270.
[6] C. Junghooa, H. Garcia-Molinaa, L. Pagea, Efficient crawling through URL ordering, Comput. Networks ISDN Systems 30 (1–7) (1998) 161–172.
[7] S. Raghavan, H. Garcia-Molina, Crawling the hidden Web, in: Proc. of the 27th VLDB Conf., 2001.
[8] A. Nierman, H. Jagadish, Evaluating structural similarity in XML documents, in: Proc. of WebDB 2002, 2002.
[9] E. Bertino, G. Guerrini, M. Mesiti, Matching an XML document against a set of DTDs, in: Proc. ISMIS 2002, 2002, pp. 412–422.
[10] R. Agrawal, C. Faloutsos, A. Swami, Efficient similarity search in sequence databases, in: Proc. 4th Int. Conf. on foundations of data organization, 1993.
[11] G. Cobena, S. Abiteboul, A. Marian, Detecting changes in XML document, in: 18th Int. Conf. on data engineering (ICDE 2002), 2002.
[12] S. Flesca, G. Manco, E. Masciari, L. Pontieri, A. Pugliese, Fast detection of XML structural similarity, IEEE Trans. Knowledge Data Eng. 17 (2) (2005) 160–175.
[13] A. Oppenheim, R. Shafer, Discrete-Time Signal Processing, Prentice Hall, 1999.
[14] B. Porat, A Course in Digital Signal Processing, Wiley, 1997.
[15] S. Flesca, G. Manco, E. Masciari, L. Pontieri, A. Pugliese, Detecting structural similarities between XML documents, Tech. Rep., ICAR-CNR, 2003.
[16] R. Baeza-Yates, B.R. Neto, Modern Information Retrieval, Addison Wesley-ACM Press, 1999.

Sergio Flesca is currently Associate Professor at the University of Calabria, Italy. He graduated in Computer Science Engineering summa cum laude, in 1996, and holds a Ph.D. in Computer Science from the University of Calabria. He has been visiting fellow at 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 is currently senior researcher at the Institute of High Performance Computing and Networks (ICAR-CNR) of the National Research Council of Italy, and contract professor at University of Calabria, Italy. He graduated in Computer Science summa cum laude, in 1994, and holds a Ph.D. in Computer Science from the University of Pisa. He has been contract researcher at the CNUCE Institute in Pisa, Italy, and 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 his Laurea degree summa cum laude in Computer Engineering and his Ph.D. in Systems and Computer 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 is senior researcher at the High Performance Computing and Networks Institute (ICAR-CNR) of the National Research Council of Italy, and contract professor at University of Calabria, Italy. He received the Laurea Degree in Computer Engineering, in July 1996, and the Ph.D. in System Engineering and Computer Science, in April 2001, from the University of Calabria. His current research interests include Information Integration, Data Warehousing, and Data Mining.

Andrea Pugliese received his Laurea degree summa cum laude in Computer Engineering and his Ph.D. in Systems and Computer Engineering from the University of Calabria, Italy. He is currently an Assistant Professor at the DEIS department of University of Calabria. His current research interests include Grid computing, databases, and semistructured data. Pugliese is a member of the ACM.