# Exploiting Structural Similarity For Effective Web Information Extraction

Sergio Flesca[1], Giuseppe Manco[2], Elio Masciari[2], Luigi Pontieri[2], and Andrea Pugliese[1]

[1] DEIS, University of Calabria

[2] ICAR-CNR

**Abstract.** In this paper we propose an architecture that exploit web pages stuctural information for the extraction of relevant information from them. In this architecture, a primary role played by a distance-based classification methodology is devised. Such a methodology is based on an efficient and effective technique for detecting structural similarities among semistructured documents, which significantly differs from standard methods based on graph-matching algorithms. The technique 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 a given impulse. By analyzing the frequencies of the corresponding Fourier transform, we can hence state the degree of similarity between documents. Experiments on real data show the effectiveness of the proposed technique.

## 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 available from Web sites, in order to effectively exploit such data for business purposes. Typical scenarios include, e.g., competitors monitoring, automatic news filtering, product finding and price comparing, etc. In order to make Web information effectively available, it is suitable to manage it through an enterprise information system. 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 [3, 11, 6, 10], to extract information from sets of HTML pages having a similar structure. The extracted information, encoded in XML, can be exploited by the enterprise information system to help decision makers or simply to offer new services to the customers. Thus, the use of HTML wrappers allows for making high-quality semistructured data available for various purposes, with the major advantage of a 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

Dagstuhl Seminar Proceedings 05061
Foundations of Semistructured Data
http://drops.dagstuhl.de/opus/volltexte/2005/230

wrapper for each set. Once these wrappers have been generated, they can continuously extract information from pages, and it is only necessary to monitor the extraction process in order to handle 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 [9, 8, 14]. In this case, pages collected by crawlers are not necessarily similarly structured, and, as a consequence, they cannot be automatically handled by wrapper programs. Moreover, most of currently available tools only permit the extraction of textual information. Thus, a company interested in exploiting this information needs a relevant human effort 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 data contained in the document. Finding heterogeneous representations of semantically similar information is a very common situation in the Web. Different tags and different combinations of them 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 document are written in HTML, that is a language specifically designed to address presentation issues and, thus, providing little expressiveness from a semantic point of view.

However, to the best of our knowledge, most wrapper languages [6, 11, 10] use only the syntactic structure of Web pages to define how to extract information, while only a few [3] have semantic facilities (actually very limited). Thus, in the following we mainly concentrate on syntactic similarity, as defined by the formatting structure of HTML tags. Despite the limited number of HTML tag names and their lack of explicit semantics, we believe 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 for them. Indeed, a wrapper is able to process only pages that exhibit almost the same syntactic structure, at least in their portion containing the relevant data to be extracted. Obviously, other portions of the pages, for instance those containing advertisements, can exhibit very different syntactic structures. We can nevertheless look at the overall syntactic structure since usually the irrelevant parts of the pages are smaller than the ones containing relevant data. 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, instead, are often not well-formed, i.e. the end tag is not always required to appear. However, most HTML parsers are able to parse not well-formed documents and represent them as a tree of elements.

In such a context, data mining techniques can be profitably exploited to classify Web pages 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.

*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 main contribution of this work is twofold:

1. we propose an architecture for the extraction of information from the Web and its storage into an enterprise information system, where crawling and wrapping modules, with specifically designed document categorization modules, are integrated to speed up the wrapping task;
2. we develop a technique aimed at HTML document categorization, which allows for both classifying found pages w.r.t. the set of available wrappers and identifying new sets of similarly structured pages, for which new wrappers can be defined.

We point out that our architechture is flexible (any wrapper or crawler can be exploited as it will be clear in the next section) and adopts an efficient and effective technique for measuring the structural similarity between semistructured 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 can hence state the degree of (structural) similarity between documents. The efficiency of this approach (it works in $O(nlog(n))$) is compelling when compared to other approaches defined in the literature [12, 4]. 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., [1]), 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 permits to reduce the costs of extracting relevant information from them. In the following subsections we first propose an architecture where crawling and wrapping systems are integrated in order to reduce the human efforts needed to extract semistructured information from the Web. As this architecture exploits document categorization algorithms to detect structurally similar pages, and to select (or build) a suitable wrapper program to process them, we next introduce the problem of structural document categorization.

### 2.1 An Architecture for Integrating Crawling and Wrapping

The proposed architecture, shown in Fig. 1, is devoted to extract interesting information from the Web and to store it into an enterprise information system. As discussed above, the aim of this architecture is to provide usable semistructured information.

The module which is responsible for finding interesting information on the Web is the *Web crawler*. This module continuously crawls the Web yielding new 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 implicitly stored in (a class of) Web documents into semi-structured data. Each class of similarly structured pages is then forwarded to the chosen wrapper program that translates the information they contain and stores them.

Obviously, not all the pages found by the 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.

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 into this architecture any kind of crawling technique [9, 8, 14] and wrapper generation system [6, 3, 11, 10] defined in the literature.

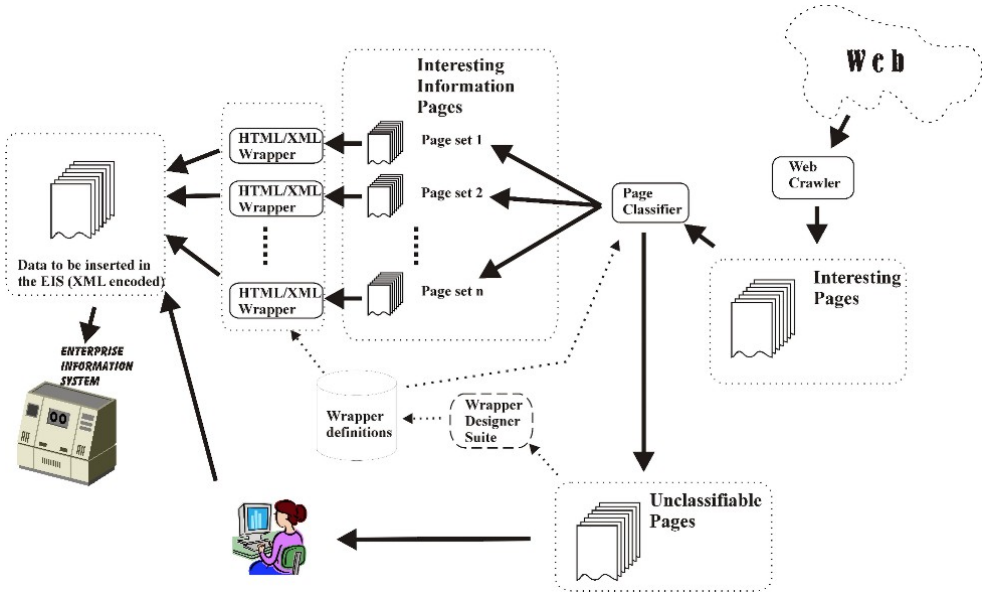



**Fig. 1.** Architecture of the information extraction system

### 2.2 Structural Document Categorization

The complexity of wrapper generation systems is strongly related to the structuring level 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 typically 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 which can be purchased on-line. If such pages are referred to the same product category, or even if they are extracted from the same service provider, it is likely that the information they provide is structured in a similar way (e.g., each product is represented as a row in a table, in which the first cell contains either the product name or the product picture). Thus, in order to design a wrapper for extracting pricing data from these pages, one has to assume that all the pages under consideration have a similar structure.

The capability of recognizing structures within Web pages is fundamental in the context of the architecture shown in Fig. 1. In particular, the page classifier and the wrapper design suite are mainly based on the structural categorization of documents according to their structure, which can be summarized as follows.

Let $\mathbf{w}$ be a wrapper, and $\mathcal{T}_{\mathbf{w}} = \{p_1, \ldots, p_m\}$ the set of Web pages used to generate $\mathbf{w}$ (the training set for $\mathbf{w}$). 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 $\mathcal{T}_{\mathbf{w}_i}$ if *(i)* the structural similarity between $p$ and each $q \in \mathcal{T}_{\mathbf{w}_i}$ is acceptable, i.e., it is higher than a given threshold, and *(ii)* no other set $\mathcal{T}_{\mathbf{w}_j}$ 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*. Then, a set $\mathcal{U}$ of unclassified pages is worth further consideration if it is possible to define a partition of $\mathcal{U}$ in $k$ clusters, where each cluster $\mathcal{C}_i$ can be exploited as a training set for learning a new wrapper $\mathbf{w}_{\mathcal{C}_i}$. Notice that the former task can be efficiently accomplished by means of $k$-Nearest Neighbor techniques, while the latter is mainly a clustering problem, for which many similarity-based approaches can be defined. Nevertheless, a major issue is the definition of the notion of similarity among Web documents according to the structure they exhibit.

## 3 Detecting Structural Similarity Among Documents

The concept of structural similarity is difficult to understand by itself. 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 in the two documents. Observe that even if it is easy to detect whether the structure of two documents is exactly the same, this test is not useful for our aims. Indeed we would like to quantify the similarity between the structures of two documents emphasizing the differences that are more relevant in defining a completely different structure. For instance we would like to consider similar two documents that have the same features with different regularities. In this respect, two HTML documents are similar if it is possible to identify equivalent subparts, even if they appear in the two documents with different frequencies.

The current literature has devoted much attention to the problem of detecting structural similarity between complex objects. In particular, several methods for detecting the similarity of XML documents [5, 12] have been recently proposed. All these methods are based on the concept of edit distance and use graph-matching algorithms to calculate a (minimum cost) edit script that contains the updates necessary to transform a docu-

ment 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.

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 (*document encoding*). By exploiting such an encoding, we check the structural similarities of documents 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[7]. However, when dealing with HTML documents, some issues arise which need to be tackled. In the following, we briefly introduce our technique for encoding and measuring the similarity of XML documents according to their structure then we show how the technique can be adapted to deal with HTML documents. Further details on the encoding techniques for XML and on the similarity measures for time series associated with the documents can be found in [7].

### 3.1 Document Encoding

An XML document is structured as a tree of elements, where each element is associated with a relevant piece of information. To our purposes, the structure of the tree shall represent the structure of the document, and in this section we define several ways of associating a time series with such a structure. In principle, we would like to flatten the tree structure into a time series which summarizes the relevant features of the original document. Notice that exploiting injective flattenings is not sufficient: since we are interested in directly comparing two time series, we would like to make this comparison as effective as possible, giving greater weights to the more relevant structural characteristics of the documents.

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 $[t_0, t_1, \cdots, t_n]$ such that $t_i \in sk(d) \Leftrightarrow t_i \in tags(d)$ 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*. The effectiveness of the document encoding is strongly influenced by the choices in the functions to adopt. Intuitively, a tag encoding function provides a numerical encoding of a tag appearing in the skeleton of a document, by looking at the "internal" properties of the tag. On the other side, a document encoding function aims at encoding a sequence of tags, by looking mainly at the features of the sequence seen as a whole. In a sense, a tag encoding corresponds to the analysis of the *locality* of a tag, while the nesting of different tags within the whole document provides a *overall* perspective; we look at the document as a globally uniform entity.

***Tag Encoding Functions.*** Given a set $D$ of XML documents, a function $\gamma$ from $tags(D)$ to $\mathbb{R}$ is a *tag encoding function* for $D$. $\gamma$ 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, it is *null* if $\gamma(el_e) = 0$. 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 mainly differ for their capability to *contextualize* a given tag, i.e., to capture information about its neighbors.

The simplest tag encoding function we consider is named *Direct tag encoding* ($\gamma_d$) and is defined 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 associates each start tag $el_s$ with the position $n$ of the tag name $tn$ of $el$ in the sequence ($\gamma_d(el_s) = n$). We complete the above definition by distinguishing between two possible encoding strategies for end tags: symmetric and null.

A simple extension of the above strategy 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 $< tn_i, tn_{i+1} >$ such that there exists a pair of tags $< t_i, t_{i+1} >$, resp. of type $tn_i, tn_{i+1}$, that appear consecutively in a document $d \in D$. We associate an integer number $P_{<tn_i, tn_{i+1}>}$ with each pair of types of tags $< tn_i, tn_{i+1} >$ 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 $P_{<tn_i,tn_{i+1}>}$.

The last strategy we propose encodes 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 $[pn_1, pn_2, \cdots, pn_k]$ 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 $pos(pn)$. Again, we distinguish between symmetric and null version of this encoding.

***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. $enc(d) = h_0, h_1, \cdots, h_n$. In the following we concentrate on three different document encoding functions. Notice that all these functions are defined w.r.t. a tag encoding function that associates tags to numbers. In particular, we assume a set of XML documents $D$, a document $d \in D$ with $sk(d) = [t_0, \cdots, t_n]$ and a tag encoding function $\gamma$.

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

A *linear encoding* of $d$ ($lenc(d)$) is a sequence $[S_0, S_1, \cdots, S_n]$, where $S_0 = \gamma(t_0)$ and $S_i = \sum_{k \leq i} \gamma(t_k)$. The main idea underlying this type of encoding is that each element $e$ of the time series associated with a document should encode more than the information corresponding to a single tag. Indeed, it computes a linear combination of the codes of the tags that appear before $t$ in the document.

A *multilevel encoding* of $d$ ($mlenc(d)$) is a sequence $[S_0, S_1, \cdots, S_n]$, where $S_i = \gamma(t_i) \times B^{maxdepth(D)-l_{t_i}} + \sum_{t_j \in nest_d(t_i)} \gamma(t_j) \times B^{maxdepth(D)-l_{t_j}}$. 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. In Example 1 we illustrate the application of the various encoding functions proposed here to a simple XML document.

*Example 1.* In this example we show the application of the concepts introduced in section 3 to the toy XML document shown below, and referred to as `book`.

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

The *tag set* of the book document is: {<xml>, <book>, <title>, </title>, <author>, </author>, <author>, </author>, <publisher>, </publisher>, </book>, </xml>}.
For the same document *tnames* = {xml, book, title, author, publisher}. Observe that tags with the same name are not considered to be the same object, so that <author> appears twice in the tag set, whereas the set of tag names does not contain duplicates.
Finally, the skeleton of the document is: <xml>, <book>, <title>, </title>, <author>, </author>, <author>, </author>, <publisher>, </publisher>, </book>, </xml>.

Figure 2 shows the output of the tag and document encoding functions described in Section 3 when applied to the book document shown above.

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

*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 | Path name | Encoding |
|---|---|---|
| <xml> | xml | 1 |
| <book> | xml.book | 2 |
| <ATTRIB@year> | xml.book.ATTRIB@year | 3 |
| <title> | xml.book.title | 4 |

*nested tag encoding*

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

*trivial document encoding*

| Sequence | Encoding | Value |
|---|---|---|
| $S_0$ | $\gamma_d$(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 |
|---|---|---|
| $S_0$ | $\gamma_d$(<xml>) $* 6^3$ | 36 |
| $S_1$ | $\gamma_d$(<book>) $* 6^1 + \gamma_d$(<xml>) $* 6^2$ | 48 |
| $S_2$ | $\gamma_d$(<ATTRIB@year>) $* 6^0 + \gamma_d$(<book>) $* 6^1 + \gamma_d$(<xml>) $* 6^2$ | 51 |
| $S_3$ | $\gamma_d$(</ATTRIB@year>) $* 6^0 + \gamma_d$(<book>) $* 6^1 + \gamma_d$(<xml>) $* 6^2$ | 45 |

*multilevel document encoding*

**Fig. 2.** Tag and document encodings for the book document.

### 3.2 Similarity Measures

Faced with the above definitions, we can now detail the similarity measure for XML documents. Observe that a document encoding function provides us with a particular view of the structure of a document $d$, corresponding to the preorder visit of the tree-structure of $d$ starting from an initial time $t_0$. Considering an encoding function, we also assume that each node (tag) is found after a fixed time interval $\Delta$. The total time spent to visit the document is $t_0 + N\Delta$, where $N$ is the size of $tags(d)$. During the visit, as we find a start-tag, we produce an impulse, that depends on a given tag encoding $e$ and the overall structure of the document, as it is represented by the selected document encoding *enc*. As a result of the above 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 too costly resizing and alignment operations, and *(ii)* stretching (or narrowing) signals is not a solution, since such operations heavily affect the corresponding document structure.

These drawbacks can be avoided if the structural properties of the signals associated with two XML documents are compared by examining their DFT transforms, which reveal much about the distribution and relevance of signal frequencies. Given a document $d$, we denote as $\texttt{DFT}(enc(d))$ the Discrete Fourier Transform of the time series resulting from the encoding. In order to compare two documents we propose to consider the difference in the magnitude of frequency components, that allows us $(i)$ 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. The overall computation of the dissimilarity between documents can be done as follows. Let $d_1$, $d_2$ be two 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:

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

where $\tilde{\texttt{DFT}}$ 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). 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. The approximation error due to interpolation is inversely related to the lengths of the sequences.

It is worth noticing that, when comparing two documents with length $N$, our method requires $O(N \log N)$, since computing their transforms is $O(N \log N)$ that is compelling w.r.t. other approaches (e.g., graph-based techniques).

## 4 Experimental Results

In this section, we present some experiments we conducted to evaluate the effectiveness of the proposed approaches in measuring structural similarity among HTML documents. The experiments were performed on real HTML documents, gathered from different sources in the Internet. From now on, we will denote a group of documents coming from the same source as a *class*. The documents we used in our tests are about 400 and belong to 16 classes, which can be grouped into the following 4 high-level *categories* (each of them corresponding to a distinct application domain): 1) `E-commerce`, containing 102 HTML documents and consisting of 4 classes, named $E_1$, $E_2$, $E_3$ and $E_4$, corresponding to 4 e-commerce Web sites; 2) `Museums`, 96 HTML documents coming from 4 classes, named $M_1$, $M_2$, $M_3$ and $M_4$, corresponding to the Web sites of 4 museums; 3) `Newspapers`, 111 HTML documents, grouped in 4 classes, named $N_1$, $N_2$, $N_3$, $N_4$, corresponding to the Web sites of 4 newspapers; 4) `Universities`, 94 HTML documents, grouped in 4 classes, named $U_1$, $U_2$, $U_3$, $U_4$, corresponding to the Web sites of 4 Universities.
The evaluation of the results in each test relies on some a priori knowledge about the used data set. In fact, we remember that the data considered in our tests belong to a predefined number of classes, i.e., documents' groups, each of them related to a given data source. The immediate result of each test is a similarity matrix $S$ representing the degree of structural similarity for every pair of documents.

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 for evaluating the effectiveness of the similarity. 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. Practically, 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 that document.

Formally:

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

where $N$ is the total number of documents, $Cl(i)$ represents the class associated with the $i$-th document in the collection, and $\mathcal{F}_k(i)$, 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 tends to have 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 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 class, but $z$ is more similar to $x$ than $y$ is. We denote this probability by $\varepsilon(S)$, which is estimated as

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

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

### 4.1 Detailed Results

In this section we describe in detail the results obtained over the data set `Newspapers`, which is composed of 4 disjoint document classes, namely $N_1$, $N_2$, $N_3$, and $N_4$. Each class corresponds to a news Web site and contains the documents extracted from it.

We recall that the direct result of each test is a similarity matrix $S$. In order to allow for an immediate feeling of the similarity relationships, we will visualize the similarity matrix as an image, using a scale of colors which range from white to black through several tones of yellow, first, and red, after. The color tone of each pixel in such an image is proportional to the value stored in the corresponding cell of the matrix (i.e., darker pixels correspond to higher similarity values). In the case of highly dense subrange of similarity values, suitable distortions will be applied to the color scale, in order to emphasize the differences among such values.

The average values of all the intra-class similarities and inter-class similarities in $S$ are summarized into a matrix $CS$ to support 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 the generic element $CS(i, j)$ is computed as follows:

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

For each of the encoding strategies presented above, we will show a graphical representation of the similarity matrix, the average of all intra-class and inter-class similarities, and the values

obtained for the error $\varepsilon$ and the quality measure $q_k$. We notice that for the latter a neighborhood size equal to 22, the minimum class cardinality, is considered.

(a)

| | $N_1$ | $N_2$ | $N_3$ | $N_4$ |
|---|---|---|---|---|
| $N_1$ | 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 |

(b)

| *measure* | *value* |
|---|---|
| $\varepsilon$ | 0.0351 |
| $q_k$ | 0.9546 |

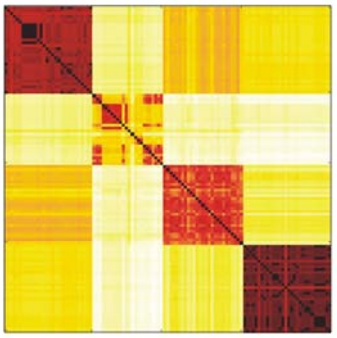
(c)

**Fig. 3.** Similarity matrix (a), average similarities (b) and quality measures (c) for **Trivial** encoding scheme

*Trivial.* At a first glance at Fig. 3.(a), the Trivial scheme seems not to be able to suitably distinguish the prior classes in the data set. In fact, while the 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 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 each other. In particular, adopting an iterative approach, we can first extract classes $N_1$ and $N_4$, 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.

*Linear.* The results shown in the right-hand side of Fig. 4 demonstrate a slight improvement in recognizing the prior classes which Linear scheme gains with respect to Trivial. As in the previous case, the last class looks as the most homogeneous one, while the second exhibits the minimum average intra-class similarity. The good performance of Linear scheme is supported by the graphical representation of the similarity matrix in Fig. 4.(a), where blocks corresponding to the intra-class similarities can be clearly individuated.



(a)

| | $N_1$ | $N_2$ | $N_3$ | $N_4$ |
|---|---|---|---|---|
| $N_1$ | 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 |

(b)

| *measure* | *value* |
|---|---|
| $\varepsilon$ | 0.0021 |
| $q_k$ | 0.9914 |

(c)

**Fig. 4.** Similarity matrix (a), average similarities (b) and quality measures (c) for **Linear** encoding scheme

*Nested.* Both the optimal values for the overall similarity measures, shown in Fig. 5.(c), and the similarity matrix, depicted in Fig. 5.(a), prove the ability of Nested scheme to adequately evaluate structural similarity over the considered data set. Also in this case, classes $N_1$ and $N_4$ can be neatly recognized, while the other ones show lower similarity, as Fig. 5.(c) confirms. It is worth noticing that this scheme emphasizes such a general trend, but it is also able to reduce the inter-class similarities relatively to the intra-class one, thus allowing for better distinguishing the classes.

*Multilevel.* The results produced by the Multilevel scheme appreciably differ from those presented so far, mainly because all the average similarities in Fig. 6.(b) are rather close to the maximum value of similarity allowed.

These results 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 codes related to all the tags enclosing it. In particular, more external is the tag the higher is the weight associated with it and computed by means of a function which exponentially depends on the nesting level of the tag. On the other side, the external levels, in any HTML document, are usually occupied by a few tag names, such as `html`, `head`, `body`, but not only. Therefore, when Multilevel scheme is applied to HTML documents, the obtained time series tend to have roughly similar shapes, thus motivating the high values of similarity detected among every pair of documents.

In spite of this phenomenon, the quality measures shown in Fig. 6.(c) prove that the performances of Multilevel scheme are good enough. Indeed, in the same figure we can observe that the intra-class similarities are yet higher than the inter-class ones and allow for separating the

(a)

| | $N_1$ | $N_2$ | $N_3$ | $N_4$ |
|---|---|---|---|---|
| $N_1$ | 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 |

(b)

| *measure* | *value* |
|---|---|
| $\varepsilon$ | 0 |
| $q_k$ | 1 |

(c)

**Fig. 5.** Similarity matrix (a), average similarities (b) and quality measures (c) for **Nested** encoding scheme

classes, as it is confirmed by Fig. 6.(a) (produced by applying a suitable distortion effect to the color scale in order to emphasize the differences).

*Pairwise.* The similarity matrix in Fig. 7.(a) 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 essentially due to the Multilevel document encoding, where each tag occurrence is associated with a linear combination of the codes assigned by a given tag encoding function to the tags enclosing that occurrence. In particular, the weight associated with each tag in such a combination is a power function, where the base is the number of distinct tag codes globally generated by the tag encoding function and the exponent depends on the nesting level of the tag in an inverse manner. Since the Pairwise tag encoding strategy considers all the distinct pairs of consecutive tags, it is likely to produce a high number of tag codes, so emphasizing the differences in the weights that tags at different levels are assigned to. Combining the Multilevel document encoding and the Pairwise tag encoding functions, hence, makes the similarity between two generic documents essentially depend on how they appear in most external elements, which we have noticed to be nearly invariant over HTML documents.

Such an effect determines some decrease on the quality of the performed similarity analysis with respect to most of the other methods, as confirmed by the global measures in Fig. 7.(c).

However, even in this case the results are globally satisfactory since all the classes can be distinguished from each other, in spite of the high inter-classes similarities and the quite low homogeneity of class $N_2$, which yet once exhibits the minimum average intra-class similarity.

(a)

| | $N_1$ | $N_2$ | $N_3$ | $N_4$ |
|---|---|---|---|---|
| $N_1$ | 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 |

(b)

| *measure* | *value* |
|---|---|
| $\varepsilon$ | 0.0142 |
| $q_k$ | 0.9643 |

(c)

**Fig. 6.** Similarity matrix (a), average similarities (b) and quality measures (c) for **Multilevel** encoding scheme

*Remarks.* In these experiments we examined a number of ways for encoding Web documents, obtained by combining a document encodings with a tag encodings. Tables 1 and 2 summarize the quality values obtained when using the above defined encoding schemes. To compute $q_k$, in each test we chose a neighborhood size equal to the minimum class cardinality w.r.t. the classes considered in the test.

| *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 1.** Error $\varepsilon$ for several data sets and methods

The shown 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 among the ones previously described. In particular, as proved by a closer look inside any test, all the considered classes are recognized as sufficiently homogeneous from a structural point of view, i.e. the similarities inside any class are generally higher than similarities between documents of

(a)

| | $N_1$ | $N_2$ | $N_3$ | $N_4$ |
|---|---|---|---|---|
| $N_1$ | 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 |

(b)

| *measure* | *value* |
|---|---|
| $\varepsilon$ | 0.0430 |
| $q_k$ | 0.9287 |

(c)

**Fig. 7.** Similarity matrix (a), average similarities (b) and quality measures (c) for **Pairwise** encoding scheme

| *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 |

**Table 2.** Quality measure $q_k$ for several data sets and methods

that class and the ones of the other classes. These generally good performances are surprising enough, especially if we consider 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, we can point out that the encoding schemes based on Multilevel document encoding function, i.e. Multilevel and Pairwise, do not exhibit as brilliant results as they do over pure XML documents [7]. In a few cases, see test 2 for Multilevel and test 5 for Pairwise, they perform yet worse than other encoding techniques. This behavior essentially come from the fact that the multilevel document encoding function, used in both these two schemes, mainly focuses on structural differences localized at most external levels, which tend to be rather similar in HTML documents. Taking advantage of this observation, we could straightforwardly improve these ap-

proaches by decreasing their dependence on the first levels of the document structure. However, due to space limitations and considering the overall satisfactoriness of the results achieved even by these encoding schemes, we will omit further investigations on this issue.

On the contrary, very good results are obtained by Nested and, surprisingly enough, by the rather simple schemes Trivial and Linear. In particular, Nested tends to perform best when applied to classes from the same category, namely in tests 1 to 4, whereas in other cases it is Trivial and Linear that obtain appreciably good results.

The dissimilar behaviors of the encoding schemes mainly depends on the different ways they deal with the context of a tag when encoding the skeleton of a document into a time series.

We can further observe that when the examined data set contains classes coming from very different categories (applicative or semantic contexts) it is likely to have a number of distinctive tag names which characterize each category, i.e. they are very frequent in any document of that category but they appear rarely, or do not appear at all inside documents of the other categories. This is the case, as an example, of the tag names `table`, `tr`, `td`, `form`, `input` and `option` for the category E-commerce, as evidenced by a simple statistic analysis we performed on the data set. In such a situation a simple recognition of characteristic tag names or repetitive sequences of them, as the one carried out by both Trivial and Linear encodings, may be more profitable than the finer encoding strategies adopted by the encoding schemes Nested, Multilevel and Pairwise. In fact, the tag context information these strategies encode is not so useful in such cases and, rather, it introduces an excessive level of detail which, acting as a sort of noise, could make less evident the massive presence of some frequent and distinctive tag names. In particular, while in Nested, Multilevel and Pairwise schemes different occurrences of such tag names can be associated with different codes in the resulting time series, all those occurrences would be encoded in almost the same way by the Trivial and Linear approaches. In fact, 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 an architecture for integrating crawling and wrapping of Web pages, by exploiting structural document categorization. Document categorization is possible due to a notion of structural similarity developed and analyzed throughout the paper. The technique, originally developed for XML documents, was successfully adapted to HTML documents, allowing for a "syntactic" structural similarity analysis. Indeed, in specific application domains, the technique has been proved effective in collecting homogeneous structures for wrapper induction. However, the proposed structural similarity measure can be improved by exploiting information retrieval techniques [2]. In particular, the combination of the distance measure we propose with traditional text processing techniques can be extremely profitable.

## References


1. R. Agrawal, C. Faloutsos, and A. Swami. Efficient Similarity Search in Sequence Databases. In *Procs. 4th Int'l Conf. on Foundations of Data Organization*, 1993.
2. R. Baeza-Yates and B. Ribeiro Neto. *Modern Information Retrieval*. Addison Wesley-ACM Press, 1999.
3. R. Baumgartner, S. Flesca, and G. Gottlob. Visual Web Information Extraction with Lixto. In *Procs. 27th VLDB Conf.*, 2001.
4. E. Bertino, G. Guerrini, and M. Mesiti. Matching an XML Document against a Set of DTDs. In *Procs. ISMIS 2002*, pages 412–422, 2002.
5. G. Cobena, S. Abiteboul, and A. Marian. Detecting Changes in XML Document. In *18th Int.l Conf. on Data Engineering (ICDE 2002)*, 2002.
6. V. Crescenzi, G. Mecca, and P. Merialdo. RoadRunner: Towards Automatic Data Extraction from Large Web Sites. In *Procs. of the 27th VLDB Conf.*, 2001.
7. S. Flesca, G. Manco, E. Masciari, L. Pontieri, and A. Pugliese. Fast detection of xml structural similarity. *IEEE Transaction on Knowledge Data Engeneering 17(2)*, pages 160–175, 2005.
8. C. Junghooa, H. Garcia-Molinaa, and L. Pagea. Efficient crawling through URL ordering . *Computer Networks and ISDN Systems*, 30(1-7):161–172, 1998.
9. T. Kistler and H. Marais. WebL - A Programming Language for the Web. *Computer Networks and ISDN Systems*, 30(1-7):259–270, 1998.
10. N. Kusmerick. Wrapper Induction: Efficiency and expressiveness. *Artificial Intellegence Journal*, 118(1-2):15–68, 2000.
11. I. Muslea, S. Minton, and C. Knoblock. Hierarchical Wrapper Induction for Semistructured Information Sources. *Autonomous Agents and Multi-Agent Systems*, 4:93–114, 2001.
12. A. Nierman and H.V. Jagadish. Evaluating Structural Similarity in XML Documents. In *Procs. of WebDB 2002*, 2002.
13. A.V. Oppenheim and R.W. Shafer. *Discrete-Time Signal Processing*. Prentice Hall, 1999.
14. S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. In *Procs. of the 27th VLDB Conf.*, 2001.