Available online at www.sciencedirect.com
SCIENCE DIRECT®
Data & Knowledge Engineering 54 (2005) 279–299
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Clustering Web pages based on their structure

Valter Crescenzi [a], Paolo Merialdo [a,*], Paolo Missier [b]

[a] *Dipartimento di Informatica e Automazione, Università Roma Tre, Via della Vasca Navale, 79, Roma 00146, Italy*
[b] *Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK*

Received 4 November 2004; received in revised form 4 November 2004; accepted 18 November 2004
Available online 8 December 2004

Abstract

Several techniques have been recently proposed to automatically generate Web wrappers, i.e., programs that extract data from HTML pages, and transform them into a more structured format, typically in XML. These techniques automatically induce a wrapper from a set of sample pages that share a common HTML template. An open issue, however, is how to collect suitable classes of sample pages to feed the wrapper inducer. Presently, the pages are chosen manually. In this paper, we tackle the problem of automatically discovering the main classes of pages offered by a site by exploring only a small yet representative portion of it. We propose a model to describe abstract structural features of HTML pages. Based on this model, we have developed an algorithm that accepts the URL of an entry point to a target Web site, visits a limited yet representative number of pages, and produces an accurate clustering of pages based on their structure. We have developed a prototype, which has been used to perform experiments on real-life Web sites.

Keywords: Information extraction; Wrapper induction; Clustering; Web modelling; Web mining

* Corresponding author.
E-mail addresses: crescenz@dia.uniroma3.it (V. Crescenzi), merialdo@dia.uniroma3.it (P. Merialdo), pmissier@acm.org (P. Missier).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2004.11.004

1. Introduction

A large number of Web sites contain highly structured regions. The pages contained in these regions are generated automatically, either statically or dynamically, by programs that extract the data from a back-end database and embed them into an HTML template. As a consequence, pages generated by the same program exhibit common structure and layout, while differing in content.

Based on this observation, several researchers have recently proposed techniques that leverage the structural similarities of pages from large Web sites to automatically derive Web wrappers [8,9,29,28], i.e., programs that extract data from HTML pages, and transform them into a machine processable format, typically in XML. These techniques take a small set of sample pages that exhibit a common template, and generate a wrapper that can be used to extract the data from any page that shares the same structure of the input samples.

Applying automatically generated wrappers on a large scale, i.e. to the structured portion of the Web, could anticipate some of the benefits advocated by the Semantic Web vision, because large amounts of data exposed throughout HTML Web sites could become available to applications. For example, the financial data published by several specialized Web sites only in HTML, could be constantly extracted and processed for mining purposes; data delivered on the Web by thematic communities could be extracted and integrated.

However, automatically building wrappers for a large number of Web sites presents several issues. Firstly, there is the problem of selecting the sample pages to feed the wrapper generation system, i.e., of identifying clusters of structurally homogeneous sample pages—this significantly affects the scalability of the approaches based on wrappers, because presently sample pages are selected manually. Second, once a library of wrappers for a Web site has been generated, given a target page the correct wrapper must be selected.

This paper addresses all these issues; we present a system that automatically discovers the main classes of pages by exploring a small yet representative portion of a site. The system produces a model of the site consisting of clusters of pages. The model is suitable for wrapping, as the pages of each cluster exhibit the required structural uniformity.

1.1. Overview

We now describe the overall approach by means of an example. Consider the official FIFA 2002 world cup Web site, [1] whose roughly 20,000 pages contain information about teams, players, matches, and news. The site content is organized in a regular way; for example we find one page for each player, one page for each team, and so on. These pages are well-structured: for instance, all the player pages share the same structure and, at the intensional level, they present similar information (the name of the player, his current club, a short biography, etc.); moreover, all team pages share a common structure and a common intensional information, which are different from those of the players. Also, pages contain links to one another, in order to provide effective nav-

[1] http://fifaworldcup.yahoo.com

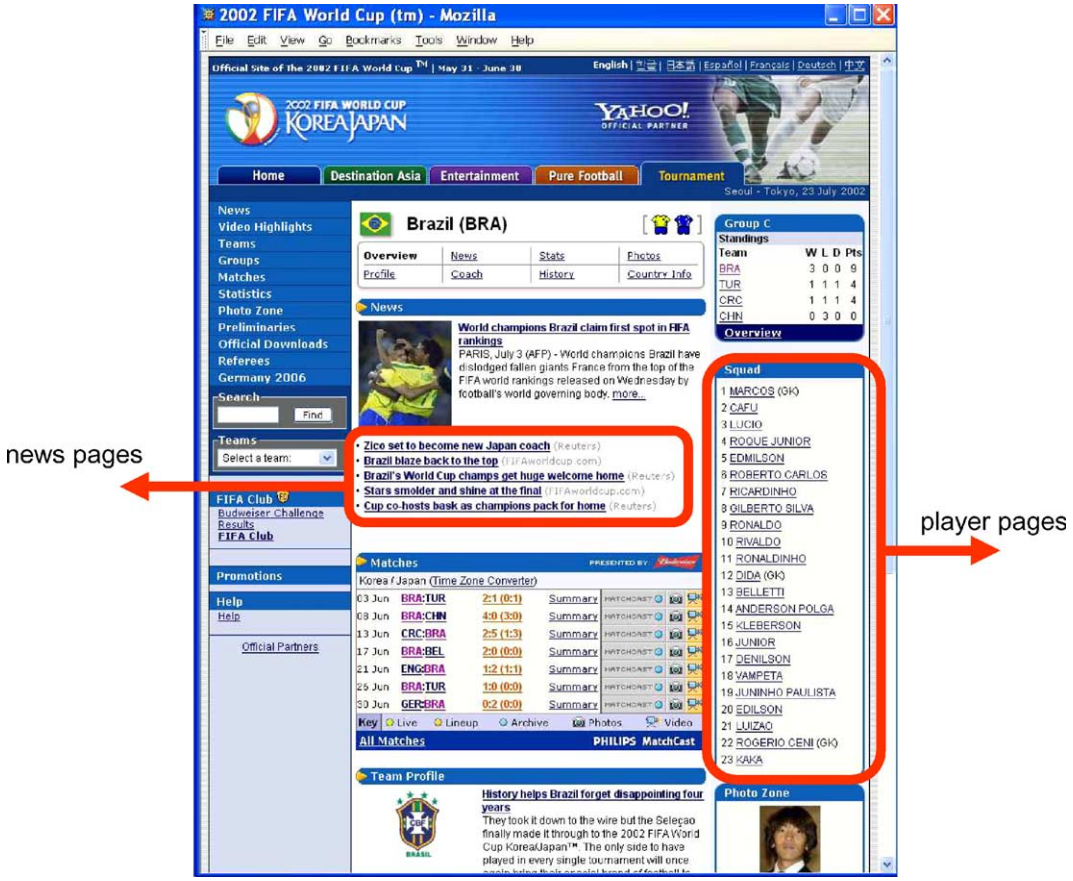

Fig. 1. Example of HTML page: a team pages.

igation paths that reflect semantic relationships; for example, every team page contains links to the pages of its players.

A key observation in our approach is that links reflect the regularity of the structure. Consider the Web page of Fig. 1, which is taken from the FIFA Web site and presents information about a national team. Observe that links are grouped in collections with uniform layout and presentation properties; we call these groups *link collections*. [2] Usually, links in the same collection lead to similar pages. For example, the large table on the right side of the page contains links to player pages; the list located in the central part of the page has links to news pages. Also, we observe that these link collections appear in every team page. Similar properties hold for the pages in Fig. 2, which offer statistics about teams. Every stats page has a collection of links organized in the left most column of a large table; all these links point to player pages.

[2] Note that a link collection may be singleton.

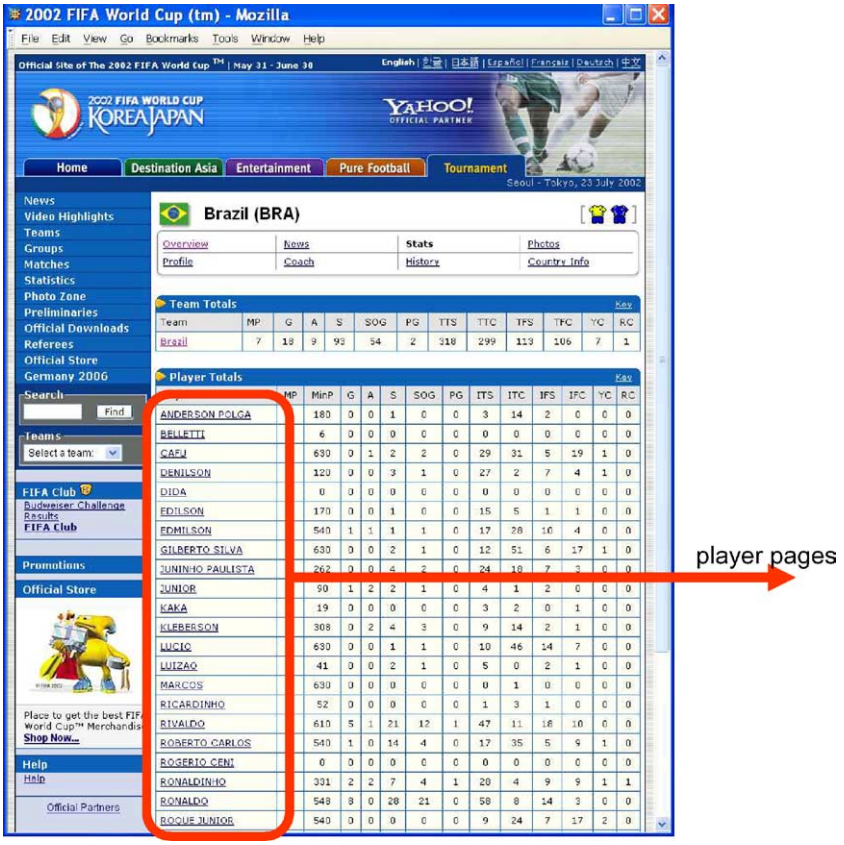

Fig. 2. Example of HTML pages: a stats pages.

Based on these observations, we argue that:

- it is reasonable to assume that links that share layout and presentation properties usually point to pages that are structurally similar. In our example, in a team page, the large table on the right contains links all pointing to player pages, while links in the central list actually lead to news pages;
- the set of layout and presentation properties associated with the links of a page can be used to characterize the structure of the page itself. In other words, whenever two (or more) pages contain links that share the same layout and presentation properties, then it is likely that the two pages share the same structure. In the FIFA example, if two pages contain a link collection inside a large table on the right, and a collection of links inside a central list, then we may assume that the two pages are similar in structure (they look like the team page in Fig. 1).

Our approach relies on the above observations. Pages are modelled in terms of the link collections they offer, and the similarity of a group of pages is measured with respect to these features. We have designed and implemented an algorithm that creates clusters of pages with homogeneous structure, while crawling only a small portion of the site. The algorithm starts from an entry point,

such as the home page, whose (singleton) page class represents the initial class, and then creates new clusters by iteratively exploring the outbound links. To minimize the number of pages to fetch, the algorithm exploits the properties of link collections. Pages reached from the same collection are assumed to form a uniform class of pages; at the same time, suitable techniques are applied for handling the situations where this assumption is violated.

1.2. Paper outline

The remainder of the paper is organized as follows. Section 2 illustrates our model in detail. Section 3 describes the algorithm for exploring the site and to cluster pages according to their structure. Section 4 reports the results of some experiments we have conducted over real-life Web sites. Section 5 discusses related works, and Section 6 concludes the paper.

2. Web site structure model

In this section we present our model to abstract the structure of a Web site, based on the main idea that layout and presentation properties associated with links can characterize the structure of a page.

For our purposes, a Web page is represented as a subset of the root-to-link paths in the corresponding DOM tree representation [1], along with the referenced URLs themselves. [3] Fig. 3 shows the portion of the DOM tree which would be considered for two sample pages. We call *link collection* one such DOM root-to-link path together with all the URLs that share that path. In our model, a page can be described exclusively through its set of link collections. According to our definition, page p_1 of Fig. 3 has two link collections: `HTML-TABLE-TR-TD` $\{url_{B1}, url_{B2}\}$, `HTML-UL-LI` $\{url_{C1}, url_{C2}\}$; and page p_2 has three link collections: `HTML-TABLE-TR-TD` $\{url_{B3}, url_{B4}, url_{B2}\}$, `HTML-B` $\{url_{B1}\}$, `HTML-UL-LI` $\{url_{C2}\}$.

A *page schema* is an abstraction of a page consisting of a set of DOM root-to-link paths. A page schema is computed simply from the page's DOM tree by considering only the set of paths starting from the root and ending in link tags. For example, the schemas of the two sample pages in Fig. 3 are: {`HTML-TABLE-TR-TD`, `HTML-UL-LI`} for p_1, and {`HTML-TABLE-TR-TD`, `HTML-B`, `HTML-UL-LI`} for p_2.

Based on the notion of page schemas we then define a *page class* as a collection of pages, and a page class schema as the union of the individual page schemas, i.e., the union of the root-to-link paths of the pages participating the collection. The page class that includes the two sample pages of Fig. 3, has the following schema: {`HTML-TABLE-TR-TD`, `HTML-UL-LI`, `HTML-B`}.

Intuitively, we expect that the page schemas for a set of structurally similar pages overlap largely, and thus the resulting class schema is not going to be much larger than the individual page schemas.

So far we have discussed the modelling of pages. Now we extend the model to include links among pages. Given a page class C_1 and one of its DOM root-to-link path pt, consider the link

[3] Actually, to exploit the richness of the presentation of modern Web sites, we also consider the name and value of HTML attributes. However, for the sake of simplicity, we will present examples including only HTML elements.

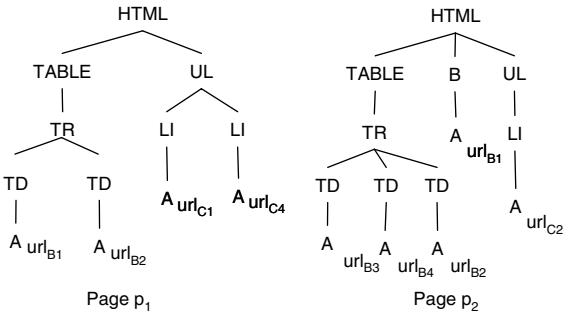

Fig. 3. Sample pages.

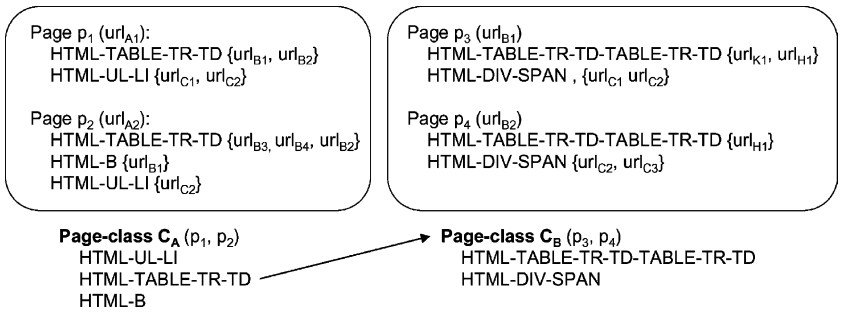

Fig. 4. Page schemas, page classes, and class links.

collections of the Web pages in C_1 associated with *pt*; we say that there exists a *class link* L between C_1 and the page class C_2 if there are links in the link collections associated with *path* that point to pages in C_2. Also, we say that *pt* is the path of L.

A *site model* is a graph whose nodes are page classes, and whose directed arcs are class links.

The notion of class link is illustrated in Fig. 4. Pages p_1 and p_2 are grouped together into a page class C_A, and pages p_3 and p_4 are members of a different page class C_B. Since the link collections in C_A with path `HTML-TABLE-TR-TD` contain links that refer to pages in C_B, there exists a class link from C_A to C_B, and `HTML-TABLE-TR-TD` is its path.

It is worth noting that a site model can actually describe the navigation paths among the classes. To illustrate this, consider the model in Fig. 4: it says that for every page belonging to class C_A, the links having path equal to `HTML-TABLE-TR-TD` lead to pages in C_B.

Intuitively, a desirable model of the site structure is one in which classes represent a useful partition of the pages, such that pages in the same class are structurally homogeneous. In the next section, we address the issue of building a site model by exploring only a fraction of the site.

3. Site-model generation algorithm

We have designed an algorithm that builds the site model incrementally, while crawling the site. The quality of the site model is evaluated with an information-theoretical approach based on the Minimum Description Length Principle (MDL) [24,16].

```
Algorithm ComputeModel
Parameter: n: max size of selected link collection subset
Parameter: dt distance threshold for candidate selection
Input: l0, a seed link to start off the site navigation;
Output: the site model M as a set of page classes C;
 1  begin
 2    Let M = ∅; // set of page classes
 3    Let Q = {l0}; // priority queue of link collections
 4    while (Q is not empty) {
 5      extract the highest priority link collection lc from Q;
 6      fetch at most n pages pointed by lc into a set w;
 7      //Candidate Class Selection
 8      Let H = (G1, ..., Gk) be the pages in w grouped by schema
                                          and ordered by cardinality;
 9      for i : 1 ... k do
10        for j : k ... i + 1 do   // from right to left
11            if (dist(Gi, Gj) < dt); collapse Gj into Gi
12        done;
13      done;
14      Let S = (C1, ..., Cg) be the resulting candidate classes in H;
15      //Model Update
16      for each C ∈ S do
17        consider all M' = (M − {C'}) ∪ {C ∪ C'} for any C' ∈ M;
18        Let Mmerge be M' such that MDL(M') is minimized;
19        Mnew = M ∪ {C};
20        if (MDL(M') < MDL(M'')) M = Mmerge;//merge class
21        else M = Mnew; //create new class
22      done;
23      add to Q the new links collections from w;
24    done;
25    return M;
25  end

Function MDL(Model M)
  {return the mdl cost of M as model of all the visited pages;}
```

Fig. 5. The Compute-Model algorithm.

Fig. 5 presents the pseudo code for the algorithm: the entry point to the site is a single given seed page, which becomes the first member of the first class in the model. Its link collections are extracted, and pushed into a priority queue. Then, the algorithm iterates until the queue is empty: at each iteration (lines 5–23), the algorithm selects one of the link collections from the queue, and fetches a subset of the pages pointed to by its links. [4]

[4] We only follow a subset of the potentially large set of links, assuming that is sufficient to determine the homogeneity properties of the entire collection.

The fetched pages are then processed in two sequential phases, as follows:

- In the first phase, called *candidate selection* (lines 8–14), pages are grouped according to their schemas. Groups are then analyzed and clustered into candidate classes; every candidate contains groups of pages having "similar" schemas.
- In the second phase, which we call *model update* (lines 16–22), the candidate classes are used to update the model. The goal of this phase is to determine whether each candidate class is a new class to be added to the model, or it should be merged with an existing class.

At the end of the second phase, the algorithm has created a refined version of the model. Finally, the new collections of outgoing links from the set of just clustered pages are added to the priority queue (line 23) and the next collection to visit is popped from the queue. The algorithm terminates when the queue is empty.

In the remainder of this section, we describe the two main phases of the algorithm in detail, and the heuristics used to set the priorities of the collections in the queue.

3.1. Candidate selection phase

The input of the candidate selection phase is a set of pages obtained by following the links of a collection extracted from the queue. This phase partitions the input pages into disjoint sets of candidate classes.

As discussed above, we assume that the links that belong to the same link collection lead to pages of the same class, i.e., to pages that are similar in structure. However, we observe that this assumption may be violated by a link collection containing links that provide access to different sections of a site, e.g. a menu or a navigational bar. According to our assumption, these entries should point to similar pages, while in reality they lead to heterogeneous pages. For example, considering again Fig. 1, the links in the leftmost column of the page form a menu, and they point to structurally unrelated pages. On the other hand, our assumption holds for link collections that lead to pages with minor differences in their schemas. For example, the algorithm should be able to group all player pages together, although pages for champion players contain one extra link with respect to other players.

The goal of this phase is to preprocess the pages coming from a link collection in order to keep menu destination pages separated, and to group pages with only minor schema differences.

First, the algorithm groups together pages with identical schema (line 8). The cardinality and number of the groups provide indications on the heterogeneity of the entire collection: few, large groups are indicative of homogeneous collections, while a large number of small or even singleton groups shows heterogeneity. In the first case, the groups are passed on to the model update phase of the algorithm, while in the second, the groups are not used for model building (however, they represent precious entry points into different regions of the site, and hence they are added to the crawler's queue).

Additionally, we deal with the important intermediate situation in which a few large groups as well as a few much smaller groups are created. If the schemas for the smaller groups are similar to one of the larger groups, this may indicate that the latter are more "authoritative" than the small groups, e.g. when a small number of old players are in the same link collection as many newer

players. In this situation, we merge pages from the smaller into the larger groups. Specifically, we define a function that measures the distance between the schemas of two groups, and repeatedly merge pairs of groups whose distance is lower than a given threshold. Since the goal is to attract small groups into large groups, an efficient way to accomplish the merge is to order groups by their cardinality, and to perform pairwise comparisons between the two largest and smallest groups, iteratively, until we run out of pairs.

The distance between schemas is defined as the normalized cardinality of the symmetric set difference between the two schemas. Namely, let G_i and G_j be the schemas of groups i and j; then:

$$\mathrm{dist}(G_i, G_j) = \frac{|(G_i - G_j) \cup (G_j - G_i)|}{|G_i \cup G_j|}$$

Note that if $G_i = G_j$ (identical schemas), then $\mathrm{dist}(G_i, G_j) = 0$; conversely, if $G_i \cap G_j = \phi$ (the schemas are disjoint), then $\mathrm{dist}(G_i, G_j) = 1$. If $\mathrm{dist}(G_i, G_j)$ is lower than a given threshold dt, [5] then G_j is *collapsed* into G_i.

This phase of the algorithm produces a few large classes, rather than a large number of smaller classes. Whenever large dominant groups are not present, like in the case of menus (which are characterized by small groups with different schemas), groups are kept well separated.

3.2. Model update phase

The candidate classes produced in the previous step must now be combined with the existing site model, to yield a refined model. The main issue is whether the candidate classes already appear in the model, or they do represent genuinely new classes. This decision process is modelled by generating a set of candidate models and then choosing the best among them, following an information-theoretical approach based on the Minimum Description Length (MDL) principle [24,16].

Intuitively, the MDL principle states that the best model to describe a set of data is the one which minimizes the sum of: (A) the length of a string of bits which encodes the model, and (B) the length of a string of bits which encodes the data with the help of the model: the shorter the description, the better the model.

Computing the model MDL involves a tradeoff between its conciseness and its precision: conciseness is expressed as the number of bits required to describe the model (Part A), while Part (B) captures a model's precision: the more precise the model, the more accurately it captures the common structure of the data it describes, with the consequence that fewer bits are required to encode the data according to the model. In our context, the data is the set of the visited pages, and the MDL principle can be used as an effective mechanism for quantifying how well a set of Web pages fits the model. In order to apply the MDL principle, we have to define an encoding function specifically tailored to our model.

Let us consider first part (A) of the encoding, which represents the model cost. Since the model is a set of page class schemas, it can be written as the concatenation of the encoding of the individual schemas. In turn, since a schema is a set of DOM root-to-link paths, to encode a class schema we may concatenate the encoding of its paths.

[5] We have experimentally determined $dt = 0.2$.

Consider for example a model M consisting of two page classes, $M = \{C_A, C_B\}$, and assume that: (*i*) C_A is associated with a schema having three paths $\{path_1, path_2, path_3\}$, and (*ii*) $C_A = \{p_1\}$ and $C_B = \{p_2, p_3\}$ (i.e. C_A contains page p_1 and C_B contains pages p_2 and p_3). Then, $enc(M) = enc(C_A) \cdot enc(C_B)$. [6] In turn, the encoding of C_A can be expressed as $enc(C_A) = enc(path_1) \cdot enc(path_2) \cdot enc(path_3)$. Similarly for C_B.

Part (B) of the MDL cost represents the encoding of a set of instances D, i.e., a set of pages, with the help of the model M, and is denoted $enc(D|M)$. As discussed in Section 2, we represent a Web page as a set of DOM root-to-link paths, together with the URLs associated with each path. Thus, we may rewrite $enc(D|M)$ in terms of the encoding of every page p in D with the help of the class C it belongs to, as follows. For our set D of pages, $D = \{p_1, p_2, p_3\}$, we write $enc(D|M) = enc(p_1|C_A) \cdot enc(p_2|C_B) \cdot enc(p_3|C_B)$.

These terms can in turn be encoded as follows. For every path in the schema of the instance p of C, we encode either its index if that path is in C's schema, or directly the path if it is not. For any path, the URLs are explicitly encoded. Also, we consider a special encoding function, denoted $enc(\{\})$, to take into account paths that are present in the class schema but are missing in the page schema.

In the previous example, suppose that page p is described as follows:

$p = \{path_1(url_{11}, url_{12}),\ path_3(url_{31}, url_{32}, url_{33}),\ path_4(url_{41}, url_{43})\}$.

In this case, the encoding of p with the help of C_A, is:

$$enc(p|C_A) = enc(1) \cdot enc(\{url_{11}, url_{12}\}) \cdot enc(2) \cdot enc(\{\}) \cdot enc(3) \cdot enc(\{url_{31}, url_{32}, url_{33}\}) \cdot enc(path_4) \cdot enc(\{url_{41}, url_{43}\})$$

In order to compute the cost for this encoding, we assign weights to the different type of schema components, namely URLs (c_u), paths (c_p), and indices (c_i), along with the cost of encoding a link whose path is not in the schema (c_{miss}). [7] The cost of $enc(p|C)$ is the weighted sum of each component in the page.

To conclude our example, the cost of the schema of C_A is $1 \cdot c_p$. The cost of encoding p according to the class C_A is $3 \cdot c_i + 7 \cdot c_u + 1 \cdot c_p + 1 \cdot c_{\text{miss}}$.

With this MDL cost formulation, we may now complete the description of the incremental model generation algorithm. For each generation step, the model update phase compares each candidate class C' with classes $\{C_1, \ldots, C_n\}$ in the model, by computing the MDL cost of the model obtained by merging C' with each C_i. The merge that produces the minimum cost (lines 17–18) is then compared with the MDL cost of the model consisting of classes $C' \cup \{C_1, \ldots, C_n\}$. Of these two final options, the one with lower cost is the new model at the end of the iteration.

3.3. Navigation heuristics

Link collections are assigned a priority based on heuristics designed to generate a high-quality model by visiting the fewest possible pages. Heuristics may be used to pursue two contrasting goals. The first goal is to drive the crawler towards unexplored regions in the site, and seek the

[6] $a \cdot b$ denotes the concatenation of strings a and b.

[7] The values of these parameters have been determined experimentally, as follows: $c_u = 1$, $c_p = 1$, $c_i = 0.8$, $c_{\text{miss}} = 1$.

relatively few pages that are potentially representative of entire classes of pages. The resulting model is rich in nodes and arcs, but enjoys low *support* because it contains few instances. The second goal is to discover a portion of the model and then accumulate support for only that portion, by visiting pages that belong to the same classes with high probability.

For the same number of visited pages, the two goals differ in terms of completeness of the model vs. the probability of finding new pages that do not fit with the model. In the first case, the model is more complete but the probability of non-fitting pages is higher than in the second case, where the crawling effort is spent on accumulating confidence in the model fragment.

In our algorithm, we have experimented with, and compared results for heuristics that cover both goals. The first, called *densest-first*, favors support-building by assigning higher priority to link collections that have many instances relative to the total number of outgoing links for a cluster. The reason is that long lists in a page are likely to point to pages with similar content (perhaps because they are generated by a program), and therefore the next set of pages will provide high support to their common class. Symmetrically, the *sparsest-first* strategy assigns high priority to the thinner link collections, in the hope that they may lead to new regions in the site. An extreme case is that of menu items, which usually belong to short collections. Regardless of the strategy chosen, collections in singleton classes are always assigned top priority.

4. Experiments

A working prototype of the algorithm has been used to conduct several experiments. To tune the algorithm parameters, we have run the system against two test sites with known structural properties. The first test site has been hand-crafted with an ad hoc structure; we used it for early experiments, specifically designed to test the algorithm implementation. The second site is the Web site of the teaching activities of our Department: [8] starting from the results of the first experiments, we used this site to get an assessment of expected performance of the algorithm.

We then moved to production Web sites. As there are no previous experiences in the literature (see Section 5), we could not rely on any test set bed. We have therefore selected some Web sites, according to the following criteria: (1) there is evidence that the pages of the site are built automatically, possibly starting from a back-end database; (2) as we rely on an external classification to evaluate our results (see below), the main classes of pages offered by the site have to be easily identified, in order to facilitate the task of building the classification of pages by hand; [9] (3) all the pages of the site must be reachable by traversing links.

Fig. 6 reports the Web sites that we have selected for our experiments: (1) the official Web site of FIFA 2002 World Cup, (2) the official Web site of the Davis Cup, (3) the official Web site of the olympic games, and (4) the ''Artcyclopedia'' Web site; also, we report the results of the experiments run over the Web site of the teaching activities of our Department. All these sites have a complex hypertext structure and contain a number of pages ranging from 2500 to 30,000.

[8] http://didattica.dia.uniroma3.it, only in Italian.

[9] In order to satisfy this requirement, sites whose pages refer to some well defined real-world entity are nice candidates.

Name	UR L	#pages
FIFA	http://fifaworldcup.yahoo.com	20,000
Davis	http://www.daviscup.com	30,000
Olympics	http://www.olympic.org	3,000
Arts	http://www.artcyclopedia.com	8,850
Teaching	http://didattica.dia.uniroma3.it	2,590

Fig. 6. The Web sites of our experimental setting.

For each of the test sites, we have built a local mirror, then we have manually classified all the pages. To ease the manual classification task, we have chosen sites for which the URLs of the page could work as a first, rough classification: we have first created classes based on the URL, and then we have refined the classification with a manual inspection of the classes. This hand-built classification was then used as *reference model* for all the experiments described in this section.

We have conducted experiments in two main directions. First, we were interested in evaluating the quality of the final computed model. Second, we aimed at tracking the quality of a model during its generation.

4.1. Evaluating model quality

To evaluate the effectiveness of the clustering produced by our algorithm, we have used the output model to classify pages from the target Web site. For each Web site, we have randomly chosen a test set S of pages, and we have classified them with the help of the corresponding computed model, by assigning page p to the class C that minimizes $enc(p|C)$. The results of the classification have been evaluated by adopting well-known measures; namely, we have used: F-measure, entropy, and purity. We now recall the definition of such measures, then present the results of the experiments.

Given the test set S, let $C_1, C_2, \ldots, C_n$ be the correct classification according to the reference model, and let $\widehat{C}_1, \widehat{C}_2, \ldots, , \widehat{C}_m$ be the set of classes from the computed model. Precision and recall for a class $\widehat{C}_j$ with respect to a reference class C_i are defined as:

$$P(C_i, \widehat{C}_j) = \frac{|C_i \cap \widehat{C}_j|}{|\widehat{C}_j|} \quad R(C_i, \widehat{C}_j) = \frac{|C_i \cap \widehat{C}_j|}{|C_i|}$$

To compute these terms, we organize our results in a *confusion matrix* as usual: element e_{ij} contains the count of pages from class C_i that have been assigned to $\widehat{C}_j$. From this matrix, precision and recall of a computed class $\widehat{C}_j$ with respect to a reference class C_i can be computed as follows:

$$P(C_i, \widehat{C}_j) = \frac{e_{ij}}{\sum_{k=1}^{m} e_{ik}}, \quad R(C_i, \widehat{C}_j) = \frac{e_{ij}}{\sum_{k=1}^{n} e_{kj}}$$

F-measure: This is just the harmonic mean of precision and recall. Given a correct class C_i and a cluster $\widehat{C}_j$:

$$F(C_i, \widehat{C}_j) = \max\left(\frac{2P(C_i, \widehat{C}_j)R(C_i, \widehat{C}_j)}{P(C_i, \widehat{C}_j) + R(C_i, \widehat{C}_j)}\right)$$

For each correct class C_i, the F-measure of that class is defined as:

$$F_i = \max_{j=1,\ldots,m} \frac{2P(C_i, \widehat{C}_j)R(C_i, \widehat{C}_j)}{P(C_i, \widehat{C}_j) + R(C_i, \widehat{C}_j)}$$

In order to provide a single synthetic value for F, we compute the F-measure of the entire model, denoted F^*, as

$$F^* = \sum_{i=1}^{n} F_i \cdot \frac{|C_i|}{|\cup_{k=1}^{n} C_k|}$$

The F-measure scores is in the range $[0, 1]$: the higher the score, the better the performance. [10]

Entropy: The entropy of a cluster $\widehat{C}_j$ is a measure of the cohesion of the elements within the cluster, and it is defined as a function of the precision $P(C_i, \widehat{C}_j)$ over all C_i:

$$H(\widehat{C}_j) = -\sum_{i=1}^{n} P(C_i, \widehat{C}_j) \log(P(C_i, \widehat{C}_j))$$

The Entropy (H) of the overall clustering is the sum of the entropy values of the clusters, weighted by their relative size:

$$H = -\sum_{j=1}^{k} E(\widehat{C}_j) \frac{|\widehat{C}_j|}{|\cup_{k=1}^{n} C_k|}$$

Here, lower scores mean better clusters.

Purity: Each computed cluster may contain pages from different classes. Purity is defined as the ratio of the largest class represented in the cluster, to the size of the cluster itself. Intuitively, a "pure" cluster is one that contains elements from only one of the reference classes. Purity measures the size of the subset of the dominant class. Specifically, the purity of cluster $\widehat{C}_j$ with respect to classes $C_1, \ldots C_n$, is a function of the number of elements assigned to $\widehat{C}_j$ during testing. Using the confusion matrix, purity is defined as:

$$P(\widehat{C}_j) = \frac{1}{|\widehat{C}_j|} \max_{k:1..n} e_{j,k}$$

where $e_{j,k}$ denotes the entry for cluster j and class i.

Similar to Entropy, overall Purity is computed as the weighted sum of each cluster's purity.

4.2. Model quality: experimental results

Fig. 7 reports the main results of our experiments. For each site we present the values of F-measure (F^*), entropy (H), and purity (P); also, we specify the number of pages of the test sets used in the experiments.

Overall, the results indicate that the models are built correctly. The results obtained for the Davis Cup Web site show a more accurate model. We comment on this, observing that the

[10] For a more in-depth introduction and motivation of F-measure, see e.g. [27].

site	F^*	H	P	#samples
FIFA	0.87	0.015	0.985	1959
DAVIS	0.98	0.003	0.997	1220
Olympics	0.87	0.04	0.93	1200
Arts	0.71	0.02	0.96	1390
Teaching	0.78	0.03	0.96	1054

Fig. 7. Page Class Quality results.

structure of the Davis Cup Web site is simpler than that of the other Web sites, since it contains a larger number of pages, organized in a very small number of classes. Specifically, the results are remarkably encouraging, showing high cluster cohesion and high purity across all the test sites. However, in relative terms, we note that the worst (Olympics) site has an entropy that is over 12 times that of the best (DAVIS) and about 2.7 times that of FIFA, indicating a higher model cohesion coming from the two larger sites. Purity, on the other hand, shows good uniformity across the test sites, indicating that most of the clusters are good representatives of individual reference classes. Note however, that both entropy and purity reflect only precision, not recall, while F^* is the only metrics that compounds both.

Upon closer inspection of the resulting clusters, two main issues contribute to lowering some of the scores. First, many misclassifications occur for very small (often singleton) classes, for which the clustering technique is not very powerful. Second, we have observed that for large classes with low performance, the problems are due to a lack of sufficient links information on a page. For example, pages in the Teaching Web site make a simple use of HTML, using a small number of link formatting styles for most pages. Similarly, in the FIFA and in the Olympics Web sites, several pages that belong to different classes contain large, unstructured textual descriptions, and most of their links are common to other page types, e.g. those in the header, in the footnote, and in the navigational bars. As our modelling algorithm benefits from a high number of class-specific links, class partitioning in these sites is less accurate.

It is worth mentioning that all the experiments were conducted using the same configuration. In fact, we have tuned the parameters of our algorithm in the first experiments, namely those involving the hand-built Web site, the Teaching and the FIFA Web sites. Experiments against the other sites were conducted adopting the same values.

4.3. Tracking model quality in incremental generation

In this experiment we aim at tracking the quality of the model during its generation, and how quickly, in terms of number of fetched pages, the algorithm converges towards a final result.

Since during its generation the model is incomplete, and the class supports are not reliable, standard quality measures are not suitable. Therefore we base the quality evaluation on the similarity between the schema of a reference model and that of the ongoing computed model.

In order to evaluate how well the schemas of the classes in the current computed model match with those in the reference model we adapt the F-measure, introduced earlier, applying it directly on the schemas.

Let σ_p, σ_C be the schemas for page p and optimal class C, respectively. We measure the extent of overlap between σ_p and σ_C in terms of precision and recall, as $R(\sigma_p,\sigma_C) = |\sigma_p \cap \sigma_C|/|\sigma_C|$ and $P(\sigma_p,\sigma_C) = |\sigma_p \cap \sigma_C|/|\sigma_p|$. Then:

$$F(\sigma_p, \sigma_C) = 2\frac{P(\sigma_p, \sigma_C)R(\sigma_p, \sigma_C)}{P(\sigma_p, \sigma_C) + R(\sigma_p, \sigma_C)}$$

We use $F(\sigma_p,\sigma_C)$ as an elementary measure of similarity between a single page and a class. By extension, we measure the similarity between a computed class $\widehat{C}$ containing pages $p_1,\ldots,p_k$, and the schema of a reference class C as the average similarity over each $p \in \widehat{C}$:

$$\mathrm{sim}(C, \widehat{C}) = \frac{\sum_{p\in\widehat{C}} F(\sigma_p, \sigma_C)}{k}$$

The overall similarity between the classes $\widehat{C}_1, \widehat{C}_2, \ldots, \widehat{C}_m$ of the current computed model, and the classes $C_1, C_2, \ldots, C_n$ of the reference model, is computed as:

$$F = \frac{\sum_{i=1}^{n} max_{\widehat{C}_j}\{\mathrm{sim}(C_i, \widehat{C}_j)\}}{n}$$

However, observe that by proceeding from the computed classes to reference classes in the similarity matching, the "asymmetric" use of F alone would tolerate the introduction into the model of spurious classes that are not similar to any reference class. Also, a single comprehensive class with a schema that trivially includes all the paths in the reference classes would receive a high score.

A complementary function, called RevF, is introduced to remedy these shortcomings. For each computed class, RevF finds its most similar reference class:

$$\mathrm{RevF} = \frac{\sum_{j=1}^{m} max_{C_i}\{\mathrm{sim}(C_i, \widehat{C}_j)\}}{m}$$

In order to achieve a balance that rewards computed classes that are both relevant and at the same level of granularity as the reference classes, we combine F with *RevF* into their harmonic average Q:

$$Q = 2\frac{F\mathrm{RevF}}{F + \mathrm{RevF}}$$

We use Q for tracking the quality of the model during its generation.

4.4. Incremental generation: Experimental results

The main results of our experiments over the FIFA Web site are summarized in Fig. 8, where the quality Q of the computed model is plotted against the number of visited pages during a run. The main interesting result is that both densest-first and sparsest-first strategies produce a satisfactory model after visiting only a small fraction of the site, with the former strategy performing slightly better. At each iteration, we identify a discovery phase during which the queue fills up with a new set of links to follow, up to a maximum that corresponds to the model reaching some

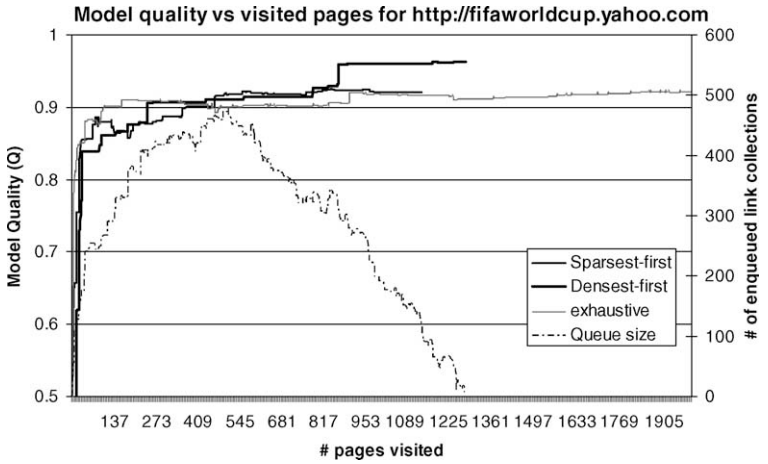

Fig. 8. Quality evaluation for FIFA.

stability. Beyond this point, the algorithm is not discovering new structure, and the time is spent to increase the support for the existing model.

These results [11] suggest that the termination condition used in our algorithm (empty queue) may actually be too pessimistic. A more aggressive condition would look for the max in queue size, although we have not shown that this is a sufficient condition in the general case.

5. Related work

The issue of modelling the logical structure of Web sites for extraction purposes has been studied in several research projects. A pioneering approach is proposed in the ARANEUS project [3,2], where a Web page is considered as an object with an identifier (the URL) and a set of attributes. The notion of *page scheme* is then introduced to model sets of homogeneous pages. Attributes of a page may have simple or complex type. Simple attributes correspond to text, images or links to other pages; complex attributes model possibly nested collections (lists) of objects. The structure of a Web site is then modelled in terms of page-schemes, and to extract data from Web pages, a wrapper is associated to each page scheme.

In *WebOQL* [4], Arocena and Mendelzon describe Web data according to an object data model, and thus propose a language (*WebOQL*) for extracting and querying the Web. The language allows users to pose queries in a SQL-like fashion, and includes features to specify a mapping between data from the actual pages and the logical constructs of the underlying data model. Also in this case, the clustering of pages according to their structure is an activity requiring a human intervention.

A more recent contribution is that of the *Wiccap* project [20]. In *Wiccap*, Web data are mapped onto a hierarchical logical structure. The focus is on the usability of the model: the goal is to map

[11] As reported in [11], a similar situation is obtained running the system over a different Web site.

information from a Web site into a logical organization of concepts, as they would be perceived by ordinary users. Nodes of the target structure can then contain data extracted from several pages, integrated to compose a uniform concept. Nodes are associated with mapping rules, i.e. primitives that extract data from the physical structure and map them onto the target model. The authors of *Wiccap* recognize that creating the hierarchical view of a target Web site is a costly activity, and therefore they have concentrated their efforts on the development of a suite of visual tools, called *Mapping Wizard* [22].

The issue of clustering HTML Web pages and XML documents according to their structure has been recently addressed in several works [10,13]. Both the approaches developed in [10] and [13] take as input a set of HTML (or XML) pages, and create clusters of pages based on properties related to the frequency and the distribution of tags. Compared to our work, these approaches do not address the issue of automatically collecting the set of pages to be clustered. As they take as input a set of pages and cluster them, they assume that all the pages of the target site should be given as input to the system. [12] On the contrary, our method includes a crawling process, that explores the site in order to collect a small yet representative number of pages. Also, the method proposed in [13] has been experimented only with small collections of synthetic XML documents. Finally, it is important to say that in [10] a technique to take into account also the similarity of the URLs is presented. However, the experiments reported in that paper show that this approach is not effective for clustering Web pages.

The use of the MDL principle [24,16] for model selection in the Web context is not new. In [14], Garofalakis et al. address the problem of inferring a reasonable DTD from a collection of XML documents; the MDL principle is used to choose a high quality DTD from a set of candidates inferred DTD. A similar technique has been proposed by Hong and Clark in order to infer wrappers for Web sources [17].

In [21], Liu et al. have developed an algorithm, called *SEW*, for discovering the *skeleton* of a target Web site. The skeleton here refers to the hypertext structure throughout the delivered contents are organized. It is assumed that a skeleton is a hierarchical structure whose nodes are either *content pages* or *navigation pages*. The former are those pages providing information contents; the latter are pages containing links to content pages. The *SEW* algorithm aims at automatically discovering the hierarchical organization of navigational and content pages. It relies on a combination of several domain independent heuristics to identify the most important set of links within each page. A similar problem has been studied also by Kao *et al.* [18], who have developed a technique for analyzing news Web sites. The goal of their proposal is to identify, within a news Web site, pages of indexes to news, and pages containing news. An entropy based analysis is performed to eliminate the redundancy of the hyperlink structure, thus distilling its complexity. A companion technique, also based on entropy analysis, eliminates redundant information, such as navigational panels and advertisements.

Compared to our approach, it is important to observe that both *SEW* and the proposal by Kao *et al.* only distinguish two predefined classes of pages, navigational (indices) vs. content pages. Also the approach by Kao et al. focuses on pages of a specific domain (news). Our goal is broader,

[12] In fact, in [10], experiments were run on small numbers, about 300, of (manually chosen) pages belonging to a few predefined classes.

as we aim at classifying pages according to their structure, without any a priori assumption about the number of classes, the exposed features, and the information domain.

Another field that is related to our research is that of focused-crawling [7]. The goal of a focused crawler is to selectively seek out pages that are relevant to a pre-defined set of topics possibly from many different Web sites. It is interesting to observe that methods to support focus-crawling rely on the notion of "pagelet", which was first introduced by Chackrabati et al. to denote a contiguous set of links [6]. In particular, Chackrabati et al. make the hypothesis that links from the same pagelet more tightly focus on a single topic than links from the entire page can do. We observe that the notion of pagelet is close in spirit, to our link collections. Compared to a focus-crawler, we may say that in some sense, we have developed a crawler that is specialized in recognizing structures, rather than topic-relevant pages. The notion of pagelet has been formalized by Bar-Yossef and Rajagopalan [5], who introduce a semantic and a syntactic definition, and an algorithm to extract pagelets from a Web page, as well. According to their syntactic definition, a pagelet is a DOM subtree such that none of its children contains more than k links, and none of its ancestor elements is a pagelet. We argue that such a definition is not suitable for our purposes, as it depends on a parameter (k) that limits the number of link that can be contained in a pagelet.

Our assumption that links from the same link collection point to related pages is reminiscent of the co-citation principle, first described in [25], which claims that the relationship between two document can be rated by the frequency at which they appear together in citation lists. In the context of the Web this principle has been exploited by several authors (e.g. [26,12]): the overall idea is that if a page A points to two pages B and C, then it is likely that there exists some relationship between B and C. In the overall context of the Web, it is assumed that such a relationship is topical; we claim that, in the restricted context of a large automatically built Web site, a structural affinity holds as well.

6. Conclusions and further work

In this paper we have presented an algorithm to cluster pages from a data intensive Web site, based on the page structure. The structural similarity among pages is defined with respect to their DOM trees. The algorithm identifies the main classes of pages offered by the site by visiting a small yet representative number of pages. The resulting clustering can be used to build a model that describes the structure of the site in terms of classes of pages and links among them.

The model can be used for several purposes. First, for each class of pages in the model we can generate a wrapper: the visited pages which have been grouped into one cluster can be used as input samples for automatic wrapper generator systems, overcoming the issue of the manual selection phase. Once a wrapper for each class has been built, the model can be used also for classifying pages, with the objective of determining which wrapper has to be applied against a given page.

It bears noting that our approach cannot crawl the so called hidden Web, i.e., pages behind forms. These pages represent a large majority of the Web, and they are suitable for wrapping, as they are usually generated automatically. In order to extract data also from these important sources, our system could be complemented by additional specific techniques, such as those proposed in [23,19].

We are currently working to improve the performance of the system. In particular, we are refining the model in order to infer other structural properties, that can help the tasks of the crawler, and the ability of discriminating pages into different classes, as well. Also, we are studying how to deal with pages containing less structured data, such as long text descriptions.

Another interesting idea we are exploring is that of associating semantics with the discovered classes. So far, we produce anonymous classes. Pages in each class are related because of their common structure. We have observed that in many cases, within a Web site, each class is associated with specific concepts. Techniques that aim at classifying pages by analyzing their contents (such as those proposed, for example, in [15]) could be integrated with our approach.

References

[1] Document Object Model (DOM) Level 1 specification, W3C Recommendation, http://www.w3.org/TR/REC-DOM-level-1 (October 1998)..
[2] P. Atzeni, G. Mecca, P. Merialdo, Managing web-based data: Database models and transformations., IEEE Internet Computing 6 (4) (2002) 33–37.
[3] P. Atzeni, G. Mecca, P. Merialdo, To Weave the Web, in: Proceedings of 23rd International Conference on Very Large Data Bases, Athens, Greece, August 25–29, 1997, pp. 206–215.
[4] G.O. Arocena, A.O. Mendelzon, Weboql: Restructuring documents, databases, and webs, TAPOS—Theory and Practice of Object Systems 5 (3) (1999) 127–141.
[5] Z. Bar-Yossef, S. Rajagopalan, Template detection via data mining and its applications, in: Proceedings of the 11th International Conference on World Wide Web, ACM Press, 2002, pp. 580–591.
[6] S. Chakrabarti, B. Dom, S. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson, J. Kleinberg, Mining the web's link structure, Computer 32 (8) (1999) 60–67.
[7] S. Chakrabarti, M. van den Berg, B. Dom, Focused crawling: a new approach to topic-specific Web resource discovery, Computer Networks (Amsterdam, Netherlands) 31 (11–16) (1999) 1623–1640.
[8] C.-H. Chang, S.-C. Lui, Iepad: information extraction based on pattern discovery, in: Proceedings of the Tenth International World Wide Web Conference, Hong Kong, China, May 1–5, 2001, pp. 681–688.
[9] V. Crescenzi, G. Mecca, P. Merialdo, ROADRUNNER: Towards automatic data extraction from large Web sites, in: Proceedings of the 27th International Conference on Very Large Data Bases, Roma, Italy, September 11–14, 2001, pp. 109–118.
[10] V. Crescenzi, G. Mecca, P. Merialdo, Wrapping-oriented classification of web pages, in: Proceedings of the ACM Symposium on Applied computing, Madrid, Spain, March 11–14, 2002, ACM Press, pp. 1108–1112.
[11] V. Crescenzi, P. Merialdo, P. Missier, Fine-grain web site structure discovery, in: Proceedings of the 5th ACM International Workshop on Web information and data management, New Orleans, Lousiana, USA, November 7–8, 2003, ACM Press, pp. 15–22.
[12] J. Dean, M.R. Henzinger, Finding related pages in the world wide web, Computer Networks 31 (1999) 1467–1479.
[13] S. Flesca, G. Manco, E. Masciari, L. Pontieri, A. Pugliese, Detecting structural similarities between xml documents, in: Proceedings of the 5th International Workshop on the Web and Databases, Madison, Wisconsin, USA, June 6–7, 2002 pp. 55–60.
[14] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, K. Shim, XTRACT: A system for extracting document type descriptors from XML documents, in: ACM SIGMOD International Conference on Management of Data, Dallas, Texas, USA, May 14–19, 2000 pp. 165–176.
[15] E. Glover, K. Tsioutsiouliklis, S. Lawrence, D. Pennock, G. Flake, Using Web structure for classifying and describing Web pages, in: Proceedings of the 11th International World Wide Web Conference, Honolulu, Hawaii, USA, May 7–11, 2002, pp. 562–569.
[16] P. Grunwald, The minimum description length principle and reasoning under uncertainty, Ph.D. thesis, University of Amsterdam, 1998 (1998).

[17] T. Hong, K. Clark, Using grammatical inference to automate information extraction from the Web, in: Principles of data mining and knowledge discovery, 5th European Conference, Freiburg, Germany, September 3–5Lecture Notes in Computer Science, 2168, Springer, 2001, pp. 216–227.
[18] H.-Y. Kao, S.-H. Lin, J.-M. Ho, M.-S. Chen, Mining web informative structures and contents based on entropy analysis, IEEE Transactions on Knowledge and Data Engineering 16 (1) (2004) 41–44.
[19] J.P. Lage, A.S. da Silva, P.B. Golgher, A.H.F. Laender, Automatic generation of agents for collecting hidden Web pages for data extraction, Data and Knowledge Engineering 49 (2) (2004) 177–196.
[20] Z. Liu, F. Li, W.K. Ng, Wiccap data model: Mapping physical websites to logical views, in: Proceedings of the 21st International Conference on Conceptual Modeling, Tampere, Finland, October 7–11Lecture Notes in Computer Science, 2503, Springer, 2002, pp. 120–134.
[21] Z. Liu, W.K. Ng, E.-P. Lim, An automated algorithm for extracting website skeleton, in: Proceedings of the 9th International Conference on Database Systems for Advanced Applications, Jeju Island, Korea, March 17–19, 2004, pp. 799–811.
[22] Z. Liu, W.K. Ng, E.-P. Lim, F. Li, Towards building logical views of websites, Data and Knowledge Engineering 49 (2) (2004) 197–222.
[23] S. Raghavan, H. Garcia-Molina, Crawling the hidden web, in: Proceedings of the 27th International Conference on Very Large Data Bases, Roma, Italy, September 11–14, 2001, pp. 129–138.
[24] J. Rissanen, Modeling by shortest data description, Automatica 14 (1978) 465–471.
[25] H. Small, Co-citation in the scientific literature: a new measure on the relationship between two documents, Journal of the American Society for Information Science 24 (4) (1973) 28–31.
[26] E. Spertus, Mining structural information on the web, in: Proceeding of the 6th International World Wide Web Conference, Santa Clara, California USA, April 7–11, 1997, pp. 587–595.
[27] C. Van Rijsbergen, Information Retrieval, second ed., Dept. of Computer Science, University of Glasgow, 1979.
[28] J. Wang, F. Lochovsky, Data extraction and label assignment for web databases, in: Proceedings of the 12th International World Wide Web Conference Budapest, Hungary, 20–24 May, 2003, pp. 187–196.
[29] J. Wang, F. Lochovsky, Data-rich section extraction from html pages, in: Proceedings of the 3rd International Conference on Web Information Systems Engineering, Singapore, 12–14 December, 2002, IEEE Computer Society, pp. 313–322.

Valter Crescenzi received his Laurea degree in Computer Engineering from Università Roma Tre, Italy in 1998, and his Ph.D. degree from Università di Roma *La Sapienza*, Italy in 2002. He is currently research assistant at Università Roma Tre. His research interests focus on information extraction from Web data sources.

Paolo Merialdo received his Laurea degree in Computer Engineering from Università di Genova, Italy in 1990, and his Ph.D. degree from the Università di Roma "La Sapienza", Italy in 1998. He is currently research assistant at Università Roma Tre, Italy. His research interests focus on methods, models and tools for the management of data for Web-based information systems and, more recently, on information extraction from Web data sources.

Paolo Missier has extensive research and industrial experience in the area of data management and software architectures. He has worked as a Research Scientist at Telcordia Technologies (formerly Bellcore), NJ, USA for eight years and later as an independent collaborator on multiple research projects with Università Roma Tre in Rome, and Università Milano Bicocca, in Italy, where he has also been teaching. He currently works as a researcher on data management issues for bioinformatics at the University of Manchester, UK, Department of Computer Science. He received a M.Sc. in Computer Science from University of Houston, Tx., USA in 1993 and a B.Sc. and M.Sc. in Computer Science from Università di Udine, Italy in 1990. ACM and SIGKDD member since 1998, IEEE member since 2001.