EFFICIENTLY LOCATING COLLECTIONS OF WEB PAGES TO WRAP

Lorenzo Blanco, Valter Crescenzi, Paolo Merialdo
Universitá Roma Tre
Via della Vasca Navale 79, I00146 Roma – Italy

Keywords: Information Extraction, Wrapper Induction, Web mining.

Abstract: Many large web sites contain highly valuable information. Their pages are dynamically generated by scripts which retrieve data from a back-end database and embed them into HTML templates. Based on this observation several techniques have been developed to automatically extract data from a set of structurally homogeneous pages. These tools represent a step towards the automatic extraction of data from large web sites, but currently their input sample pages have to be manually collected. To scale the data extraction process this task should be automated, as well. We present techniques to automatically gathering structurally similar pages from large web sites. We have developed an algorithm that takes as input one sample page, and crawls the site to find pages similar in structure to the given page. The collected pages can feed an automatic wrapper generator to extract data. Experiments conducted over real life web sites gave us encouraging results.

1 INTRODUCTION

Many web sites represent authoritative information sources for several domains. Unfortunately, the precious information they provide is spread across thousands of HTML pages, connected by a complex network of hypertext links. Such an interface is suitable for a human consumer, but it makes the published data not easily accessible to applications. However, there is a relevant and increasing number of Web sites that are built automatically, possibly dynamically, by scripts that create pages by encoding into HTML templates data usually coming from a back end database.

The automatic nature of the page generation process confers a certain degree of regularity to the internal structure of the pages. Roughly speaking, we may say that pages generated by the same script share a common structure, while they differ one each other in the data they present. Motivated by this observation, recently several researchers have studied techniques that exploit similarities and differences among pages generated by the same script in order to automatically infer a Web wrapper, i.e. a program to extract and organize in a structured format data from HTML pages (Arasu and Garcia-Molina, 2003; Crescenzi et al., 2001; Crescenzi and Mecca, 2004; Lerman et al., 2004; Wang and Lochovsky, 2002). Based on these techniques they have developed tools that, given a set of pages sharing the same structure, infer a wrapper, which can be used in order to extract the data from all pages conforming to that structure.

These tools represent a step towards the automatic extraction of data from large Web sites. However to scale up the approach an open issue is how to crawl a web site in order to automatically locate collections of pages to wrap. To give an example, consider a Web site providing information about the stock market, such as `http://www.axl.co.uk`. The site contains more than 20,000 pages. Among these, there are about 3,750 pages containing descriptions of the UK companies (one page per company): the data provided by these pages can be an important information source for many applications. As these pages are generated automatically, their structure is likely to be regular, and they can be wrapped in order to extract and organize the data about *all* the companies in a structured format, e.g. in XML. This goal can be accomplished by manually choosing all the company pages, and then generating one wrapper for them. However, due to the dimensions and the complexity of the site graph, the manual collection of the company pages results neither feasible nor scalable.

We sketch a big picture of the data extraction process in Figure 1: (1) given one page, represent-

Blanco L., Crescenzi V. and Merialdo P. (2005).
EFFICIENTLY LOCATING COLLECTIONS OF WEB PAGES TO WRAP.
In *Proceedings of the First International Conference on Web Information Systems and Technologies*, pages 247-254
DOI: 10.5220/0001234202470254
Copyright © SciTePress

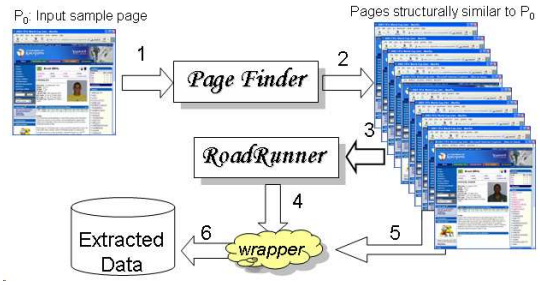

Figure 1: The Web Data extraction process

ing one sample of the class of pages from which data should be extracted, a module, called *Page Finder*, crawls the target Web site, and (2) locates and collects pages that are similar in structure to the input sample; (3) pages collected in the previous step are used to generate a wrapper: we use our ROADRUNNER system, but any other automatic wrapper generator could be used; (4) once a wrapper has been generated, (5) it is run over the collected pages and (6) the data offered by these page are extracted.

This paper concentrates on the development of an efficient algorithm to implement the *Page Finder*'s functionalities. We present techniques to automatically crawling a web site in order to reach pages containing data of interest. The only input we require is one sample Web page: the system picks out from the site all the pages that obey to the same structure of the input sample. Then, the output set of pages can be used to feed an automatic wrapper generator system.

Our goal is to effectively and efficiently obtaining the largest coverage, while minimizing the total number of downloaded pages. The main idea underlying our approach is that of incrementally building an effective yet simple local model of the site structure during the crawling. The site model is used to focus the crawling towards the desired target pages according to the evidence provided by the pages already downloaded.

Contributions The main contribution of the paper is an efficient method for retrieving from a large web site useful pages for data extraction purposes. Our method is based on (i) a simple yet effective model that abstracts the structural features of web pages, and (ii) an efficient algorithm that navigates the site to fetch pages similar in structure to one input sample page. Combined with an automatic wrapper generation system, our method provides infrastructure for efficient and scalable data extraction from the web.

Paper Outline The paper is organized as follows. Section 2 presents a simple model to abstract the structure of web pages, and the topology of a web site. Based on the model, in Section 3 we present an algorithm for crawling the site focusing the exploration towards the pages structurally similar to an input sample page. Section 4 presents an experimental experience we have conducted to evaluate our approach on real-life web sites; Section 5 discusses related works, and Section 6 concludes the paper.

2 WEB PAGE STRUCTURE MODEL

We now introduce the main intuition at the basis of our model by means of an example. Consider the official FIFA 2002 world cup Web site, whose roughly 20,000 pages present information about teams, players, matches, and news. The site contents are organized in a regular way; for example, there is 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. Similarly, all the team pages share a common structure (and, at the intensional level, the same information). However, the information as well as the structure of the team pages are different from those of the player pages. Pages also contain links to one another; these links provide effective navigation paths, and usually they reflect semantic relationships among the pieces of information offered by the connected pages. In our example, every team page has a list of links to the pages of its players (and vice versa), a list of link to the matches it played, and so on.

A key observation for abstracting and modelling in a simple way the structure of pages of automatically generated Web sites, is that they usually contain a large number of links, and these links reflect the regularities of the structure. In particular, we argue that the set of layout and presentation properties associated with the links of a page can characterize the structure of the page itself. Whenever a large majority of links from two (or more) pages share the same layout and presentation properties, then it is likely that the two pages share a common structure. Consider the Web page of Figure 2, which is taken from the FIFA Web site: it presents information about a national team. In this site, every team page contains a sequence of links to the player pages organized on a table on the left, an itemized list of links to the news located in the central part of the pages, and so on. Differently, pages such as that shown in Figure 3, which offer statistics about players, are characterized by a collection of links placed in the left most column of a large table.

We model pages in terms of the presentation and layout properties of the links they offer, and we measure the structural similarity between pages with respect to these features.

More precisely, in our model, a web page is de-

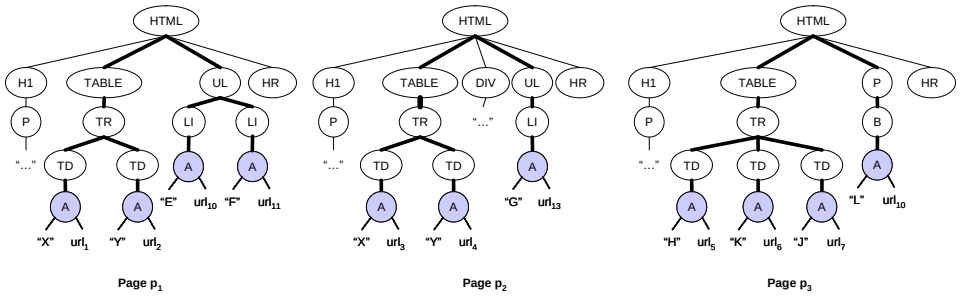

Figure 4: DOM trees

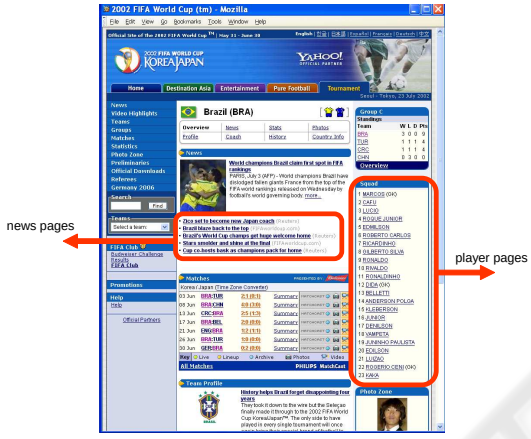

Figure 2: Example of HTML page: a team pages

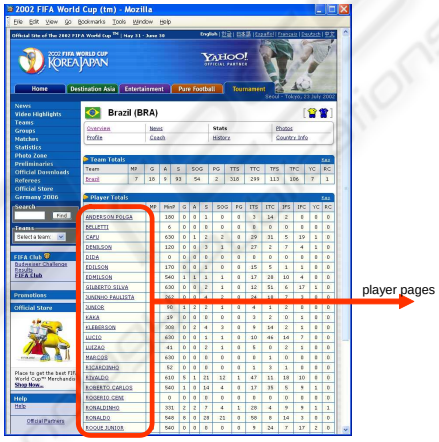

Figure 3: Examples of HTML pages: a stats pages

scribed by considering only the paths of the corresponding DOM tree that start from the root and terminate into an anchor node. We call *link collection* one of such DOM paths together with all the `<url, anchor>` pairs that share that path. Also, we call *tag-list* the root-to-anchor path of each link collection, and the *page-schema* of a page p, denoted $\Delta(p)$, is the set of tag-lists in p.[1]

To give an example consider Figure 4: it shows the (simplified) DOM trees of three fictional pages p_1, p_2 and p_3. According to our definition, only paths leading from the root node to anchor nodes (these paths are marked in bold) are taken into account, and the three pages are modelled by link collections, as follows:

p_1 : `HTML-TABLE-TR-TD` $\{\langle url_1, "X"\rangle, \langle url_2, "Y"\rangle\}$
`HTML-UL-LI` $\{\langle url_{10}, "E"\rangle, \langle url_{11}, "F"\rangle\}$
p_2 : `HTML-TABLE-TR-TD` $\{\langle url_3, "X"\rangle, \langle url_4, "Y"\rangle\}$
`HTML-UL-LI` $\{\langle url_{13}, "G"\rangle\}$
p_3 : `HTML-TABLE-TR-TD` $\{\langle url_5, "H"\rangle, \langle url_6, "K"\rangle, \langle url_7, "J"\rangle\}$
`HTML-P-B` $\{\langle url_{20}, "L"\rangle\}$

Observe that according to our model, p_1 and p_2 have identical page-schemas:
$\Delta(p_1) = \Delta(p_2) =$ {`HTML-TABLE-TR-TD, HTML-UL-LI`};
the page-schema of p_3 is:
$\Delta(p_3)$={`HTML-TABLE-TR-TD, HTML-P-B`}.

Pages can be grouped by their schema into clusters. A *cluster of pages* is a (possibly singleton) set of pages with identical schema. Consider again the three pages in Figure 4: pages p_1 and p_2 form a cluster, as they have identical schemas. Also, page p_3 gives raise to a singleton cluster.

Finally, we introduce the concept of *class-link* to model at the intensional level a set of link collections sharing the same root-to-anchor path. Thus, given a cluster of pages, i.e. a set of pages with identical schema, each tag-list identifies a class-link.

Page similarity To have a measure of the structural similarity between two pages we consider the distance between their schemas. Since schemas have been defined as sets (of tag-lists), we consider well known set

[1] Actually, we also consider the name and value of HTML attributes associated to the nodes of the path. However, for the sake of simplicity, we present examples including only HTML elements.

similarity methods (Van Rijsbergen, 1979). In particular we adopt the Jaccard coefficient; given two sets X and Y, the Jaccard coefficient is defined as:

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

In our context, let $\Delta(p_i)$ and $\Delta(p_j)$ be the schemas of pages p_i and p_j, respectively; then:

$$sim(p_i, p_j) = J(\Delta(p_i), \Delta(p_j))$$

Note that if $\Delta(p_i) = \Delta(p_j)$ (i.e. the two pages have identical schemas), then $sim(p_i, p_j)$ equals 1; conversely, if the two schemas are disjoint, then $sim(p_i, p_j)$ equals 0. Given two pages, p_1 and p_2, we say that they are *similar* if $sim(\Delta(p_1), \Delta(p_2)) \geq T$, where T is a threshold value determined empirically.[2]

We now conclude the presentation of our model with an observation that plays a relevant role in our approach for efficiently collecting structurally homogeneous pages: usually links in the same link collection lead to similar pages. For example, consider the Web page in Figure 2: all the links contained in the large table on the right side of the page lead to player pages; similarly, all the links located in the central part of the page bring to news pages. Then, given a cluster of pages, we can make the conjecture that most (possibly all) of the links contained in the link collections of a class-link are homologous, that is, they lead to structurally similar pages. In our running example, given a cluster of team pages, it is reasonable to suppose that all the links belonging to the link collections corresponding to the large table on the right side of the page lead to similar pages (in fact, they lead to player pages).

With these ideas in mind, we can now introduce the algorithms to crawl a web site focusing the search on pages structurally similar to one input sample page.

3 CRAWLING FOR STRUCTURE

Given one sample page containing data of interest, our goal is to pick out from the site the largest number of pages similar in structure to the input page. A first intuitive approach is to fetch pages that are reachable from the sample page, and to select those that fall in the cluster which would include the sample page.As pages of large web site are widely connected one each other, it is likely that one page contains links leading to pages sharing the same structure (i.e., offering, at the intensional level, the same information content). For example, consider the team pages the FIFA Web site: they contain links pointing to the other teams they played (or will play) against. As another example, consider the pages describing products in any e-commerce Web site: they usually contain links to pages of related items, as well. This technique is very simple, but it usually gathers a small number of pages, so it is not effective for our purposes.However, we observe that some of the pages that we have reached, though not directly target pages, may result useful because they contain links that point backward to the target page; it is likely that these pages also list links to other pages belonging to the target set. Continuing our FIFA example, from every the team page several pages containing back links can be reached; for example, from every team page there are links to pages listing matches, and links to referee pages; all these pages contain links pointing back to the teams. So, it could be convenient to focus the crawling towards the outbound links of these pages. However, observe that one page can offer a large number of links, and just a subset of them could be useful for our purposes. To define such a subset we still rely on our model: namely, we only consider links belonging to the link collections that contain some pointer to pages in the target cluster, as it is likely that other links of the same link collection point to similar pages.

Other target pages might be reached by following the link departing from new index pages. Actually, the searching of new index pages is another instance of the original problem: the goal is now to collect all the pages with the same structure as those serving as indices over the original target.

The above approach is simple but it is not efficient (in terms of number of total pages to download). Since it visits *any* page that can serve as an index to the target, it may crawl a large portion of the target Web site, just to discover that most of downloaded pages are either completely useless or redundant. Some of the fetched index pages could be very effective, as they can provide a quick access to a large number of new pages; others simply go to pages that have been already reached, adding the cost of searching such index pages without any benefits. Also, another problem that limits the efficiency of this approach is related to the depth at which index pages should be searched. As we assume the structure of the site is not known, we have to statically set up the depth of the recursive step in order to visit a large portion of the site; in many cases, the searching involves a large number of pages that do not carry any contribution to the overall result.

In order to overcome these limitations, we have developed a more involved algorithm, called INDESIT.

3.1 Learning Promising Paths

INDESIT has been designed in order to follow only paths to effective index pages, i.e. pages containing links that lead to a large number of new target pages. The underlying idea of INDESIT is that while crawl-

[2] In our experiments, we have set $T = 0.85$.

ing, it is possible to acquire knowledge about the navigational paths the site provides and to give higher priority to the most promising and efficient paths.

The INDESIT algorithm performs four main steps, as summarized in Figure 5. In order to clarify its recursive nature, it is more convenient to restate the problem as if we aimed at obtaining all the pages of clusters similar to a given cluster (namely, that of the sample page). The algorithm starts with the (singleton) cluster C_0 containing the input sample page p_0.

In the first step, *Neighborhood cluster discovery*, the link collections of the input cluster are used to discovered as many neighborhood clusters as possible. In order to minimize the number of downloads, whenever more that n links[3] of the same collection lead to pages of the same cluster, the remaining links of the collection are not followed. The rationale is that for large link collections it is likely that the skipped links will lead to clusters already met at the beginning of the collection. In essence, we are checking the homologous link conjecture.

In the second step, *Neighborhood clusters evaluation*, the neighborhood clusters are analyzed to select the most promising directions. As discussed above, we can look for navigational paths pointing back to pages of the target cluster. At the intensional level of clusters, paths are represented by class-links, i.e. sets of link collections with the same tag-list. So we now search for class-links leading back to the target cluster. We could consider all the class-links made by link collections with at least one link pointing back to pages of the target cluster. However, a simple optimization consists in anticipating the navigation of class-links that provide access to a large number of *new* target pages. To this end, for every class-link we compute a score indicating a measure of earn, in terms of new target pages, the class-link carries on.

The score of a class-link is computed by navigating its link collections, and then counting how many *new* pages with schema similar to that of p_0 are reached. A page is considered as new if it was not already present in C_0. Observe that during the evaluation of one link collection, it may happen that a page has been already downloaded to evaluate another link collection; but, if such a page was not in C_0 at the beginning of the evaluation, it will be considered as new. Note that in this way the evaluation order does not affect the scores. The scores of class-links are finally computed as the average scores of its link collections.

Most collections can be evaluated by following only a small fraction of its links: as soon as a link leads to a page whose schema is not similar to the schema of C_0, the collection and its class-link are immediately scored 0 without downloading any other page. At the end of this step we have a score associated with every class-link. Only class-link leading to fresh target pages will have positive scores; we call them *back class-links*. The overall score of each neighborhood cluster is finally computed as the sum of the scores of its back class-links. A cluster with a high score corresponds to an effective index for our target pages.

[3]We set n=3.

Algorithm INDESIT
Input: a cluster containing page p_0
Output: a set of pages structurally similar p_0
1. Neighborhood clusters discovery links of the initial cluster are followed to gain knowledge of as many neighborhood clusters as possible
2. Neighborhood clusters evaluation neighborhood clusters are evaluated and ordered: the best cluster concentrates in a small number of pages the largest number of links leading to new target pages
3. Recursion to catch index pages a new instance of our initial problem is triggered. Pages similar to the best neighborhood cluster are the target of the new problem. They will serve as indices over target pages of the original problem
4. Iteration discovered index pages are used to fetch as many target pages as possible. Other neighborhood clusters are eventually considered

Figure 5: Summary of the INDESIT algorithm

In the third step of the algorithm, *Recursion to catch index pages*, a new instance of our problem is recursively triggered. The target of the new search is the neighborhood cluster with the highest score, which we call *index cluster*. The recursive invocation will return a set of *index pages* (possibly of different clusters) all similar to the pages of the index cluster. The link collections of these index pages with the same tag-list as the back class-links in the index cluster are followed to gather up as many target pages as possible.

Once these pages have been collected, the last step, called *Iteration*, considers the remaining neighborhood clusters. As after the last evaluation several pages have been downloaded, the scores can significantly decrease. Usually in few iterations all the scores equals zero. Otherwise a new index cluster is chosen and the algorithm goes on along the same way.

4 EXPERIMENTS

We have implemented the INDESIT algorithm and have used it in order to conduct experiments on real-life web sites. The goal of the experiments was to evaluate the performances, in term of effectiveness and efficiency, of the algorithm. We now present the sites and the input sample pages that we have used in our experiments, and the metrics we use to evaluate the performance of our algorithms against this settings. Then we report the results of our experiments.

Site	Target Class	|Class|
www.fifaworldcup.com	player	736
	teams	32
www.supercarsite.net	car	294
www.sendit.com	dvd	1,011
www.olympic.org/uk	sport	36
	hero	292
www.guitaretab.com	guitar tab	29,032
	group	3608
www.axl.co.uk	company	3,744
www.nba.com	player	648
access.euro2004.com	team	16
www.euro2004.com	team	16

Figure 6: Site and classes of the test bed

4.1 Sites and sample selection

As there are no analogous experiences in the literature, we could not rely on any reference test bed. We have therefore selected a bunch of sites, and for each of them we have identified one or two pages to be used as input samples. As our studies concentrate on large automatically built web sites, we have chosen sites for which there is evidence that their pages are generated by scripts. We selected sites from different domains and of different sizes (ranging from 5,000 to more than 30,000 pages). For all the sites we have built a local mirror. Since we aim at measuring the coverage of our output, we restricted our experiments to sites for which we were able to determine the size of the classes of searched pages. For some of the sites it is possible to derive this information as classes of pages describe real-world entities whose number is known; for example, we have used the official web sites of some relevant sport events, for which the number of athletes and teams is given. Also, we have chosen sites with pages generated by scripts such that the urls were able to classify the pages (e.g. the url of a page contains the name of the generating script); based on the urls, we have computed the cardinalities just counting the number of files with a given pattern on the file system storing the local mirror.

We have chosen pages belonging to classes with different cardinalities. For every class, we have run the algorithm three times, each starting with a different sample page; the goal is to verify that the behavior of the algorithm does not depend on specific features of the starting page. As we describe in Section 6, we have also run the algorithm in order to find pages of singleton classes. The description of the sites is given in Figure 6. Note that, for the Uefa2004 web site we have considered also the WAI version. The peculiarity of this version is that it offers the same contents and the same site topology as the standard version, but the HTML code of its pages is extremely simple.

4.2 Metrics

The effectiveness of our approach can be measured in terms of precision and recall. Let C_i be the set of pages to be retrieved, and C_j the set of retrieved pages. Precision P, and recall R are defined as follows:

$$P = \frac{|C_i \cap C_j|}{|C_j|} \qquad R = \frac{|C_i \cap C_j|}{|C_i|}$$

As our algorithm aims at minimizing the pages to fetch, a straightforward measure of the efficiency is given by the total number of downloads.

4.3 Experimental Results

The experimental results of our evaluation are summarized in Figure 7, that illustrates the performances of the INDESIT algorithm over several samples. For each sample page, we report recall (R), precision (P), and the number of downloaded pages (#dwnl).

As we can see, recall values are rather high for every experiment. This means that INDESIT is able to retrieve most of the pages with the same structure as the input sample page. The values of precision are more interesting. The precision is related to the number of false positives produced by the search. Overall, the results are good and the pages collected by our approach are structurally homogeneous, as required. The worst results are those obtained in the WAI version of the UEFA2004 web site. The extremely low value of precision (0.02%) indicates that almost all the downloaded pages have been considered as member of the target class. This is not surprising: as the HTML code of these page is extremely simple and uniform, the algorithm is confused, as it is not able to distinguish pages. On the contrary, observe that in the standard version of the same Web site, INDESIT produces sharp results, with good performances.

Comparing the numbers of downloads with the cardinalities of the target classes, INDESIT is able to learn quickly the most promising navigation paths, confining the crawling around a small yet useful portion of the site. This is particulary evident for small classes, such as, for example, the 32 FIFA team pages. Another important comment on the experimental results is that the approach is rather stable with respect to the input page. As shown in Figure 7, we obtain convergent results though starting from different samples of the same class.

5 RELATED WORK

The issue of extracting information from the web has been the subject of several studies. Most of the

Sample	Results		
	R	P	#dwnl
FIFA: player			
"Ronaldo"	100%	100%	1,046
"Lovenkrands Peter"	100%	100%	1,079
"Luna Braulio"	100%	100%	1,104
FIFA: team			
"Turkey"	100%	100%	249
"Mexico"	100%	100%	248
"Germany"	100%	100%	265
Supercars: car			
"Ferrari 360 Modena"	100%	99.32%	351
"MG TF 160"	100%	98.98%	352
"Jaguar XKR 100"	100%	98.98%	350
SendIt: dvd			
"Mystic River"	85.95%	100%	3,377
"M.A.S.H."	84.37%	100%	3,333
"Big Jake"	74.38%	100%	3,209
Olympics: sport			
"Tennis"	100%	100%	125
"Skating"	100%	100%	140
"Athletics"	100%	100%	128
Olympics: hero			
"Stefania Belmondo"	100%	100%	489
"Abele Bikila"	100%	100%	465
"Gregory Louganis"	100%	100%	458
Guitar tabs: song			
"Smells like ..."	87.29%	100%	29,099
"Wish you were here"	87.30%	100%	29,110
"Oribidoo"	87.04%	100%	29,037
Guitar tabs: group			
"Led Zeppelin tabs"	100%	95.98%	3,809
"Quaye Finley"	100%	95.98%	3,811
"John Fogerty"	97,22%	93.56%	3,583
AXL: company			
"Imperial Chem."	99.89%	98.09%	3,835
"Pacific Media"	99.89%	98.09%	3,835
"Xaar"	99.89%	98.09%	3,835
NBA: player			
"Marc Jackson"	76.23%	100%	936
"David Brown"	75.93%	100%	946
"Kevin Willis"	75.93%	100%	938
NBA: coach			
"Jerry Sloan "	100%	100%	764
"Scott Byron"	100%	100%	764
"Rick Adelman "	100%	100%	764
UEFA2004: team			
"Portugal"	100%	100%	163
"Sweden"	100%	100%	157
"England"	100%	100%	152
UEFA2004-WAI: team			
"Portugal - WAI"	100%	0.02%	755

Figure 7: Experimental results

work concentrates on the generation of wrappers (for a survey see, e.g. (Laender et al., 2002)). Most recent studies have focused on the automatic generation of wrappers. The ROADRUNNER system is based on a grammar inference approach (Crescenzi et al., 2001; Crescenzi and Mecca, 2004). Based on different techniques, systems for the same goal have been developed also by Arasu and Garcia-Molina (Arasu and Garcia-Molina, 2003) and by Wang and Lochovsky (Wang and Lochovsky, 2002). All these systems are based on the observation that data published in the pages of large sites usually come from a backend database and are embedded within a common template. Therefore the wrapper generation process consists of inferring a description of the common template. To the best of our knowledge, both automatic and semi-automatic approaches currently known assume that the pages containing the data have been collected by some external module (possibly they have been manually chosen).

Our work is related to recent research for discovering the structure of web sites. Liu et al. model a web site as a hierarchical structure, whose nodes are either *content pages*, i.e. pages providing information contents, or *navigation pages*, i.e. pages containing links to content pages (Liu et al., 2004). A combination of several domain independent heuristics is used to identify the most important set of links within each page. Based on these heuristics they have developed an algorithm to infer the hierarchical organization of navigational and content pages. Kao *et al.* (Kao et al., 2004) have studied methods to address a similar problem; they focus on news web sites with the objective of distinguishing pages containing indexes to news, and pages containing news. Compared to our approach, these proposals aims at finding paths to all generic content pages, while we aim at finding all the pages of a specific class. In addition, the approach by Kao et al. is tailored for web sites of a specific domain (news). In (Crescenzi et al., 2003), a preliminary version of our page model has been adopted in order to infer an intentional description of a web site. In that work, a crawler navigates the web site starting from the home page and an agglomerative clustering algorithm groups pages into classes. The goal is to identify all the classes of pages offered by a site and to collect a minimal characteristic sample for the whole web site.

The notion of link collection is close to that of "pagelet", which was first introduced by Chackrabati et al. to denote a contiguous set of links (Chakrabarti et al., 1999a). 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. This hypothesis has been exploited in subsequent works (Chakrabarti et al., 1999b), with the goal of driving a crawler towards pages of a given topic. Compared to their work we may say that, in some sense, our crawler is specialized in recognizing structures, rather than topic-relevant pages. The notion of pagelet has been formalized by Bar-Yossef and Rajagopalan, who introduce a semantic and a syntactic definition, and an algorithm to extract pagelets from a web page, as well (Ziv Bar-Yossef and Rajagopalan, 2002). 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 weak for our purposes, as it depends on a parameter (k) and 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 (Small, 1973), 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. (Spertus, 1997; Dean and Henzinger, 1999)): the overall idea is that two pages are likely to be related if they are linked by the same page. In the context of large web sites, we claim that, as sites are built automatically, this principle can be restated by saying that pages pointed by the same link collection are likely to be structurally homogeneous.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we have developed an automatic technique for retrieving web pages similar in structure to one only sample page. The output set of pages, as structurally homogeneous, can be done as input to a wrapper generator system. Our approach relies on the observation that in Web sites made by automatically generated pages, a degree of regularity occurs also on the overall topological structure of the site graph. Though large and complex, the site graph is organized according to tricky yet regular networks of links. Our techniques aim at exploiting such regularities. One could think that a different approach for collecting pages is to download all the pages of the target web site, and then to cluster them according to their structure (the cluster containing the sample page would be the expected output). We observe that such an approach would not be efficient, due to the number of pages of the Web site, which can be several magnitudes higher than the number of the searched pages. We have experimented our method over several real-life web sites obtaining interesting results. We are currently integrating the presented techniques with ROADRUNNER, our web wrapper generation system. We believe that this step can lead to significant improvements in the scalability of data extraction from the web.

Our approach is suitable for crawling dynamic pages. On the other hand it is not designed to crawl pages behind forms. In order to extract data also from these important sources, our system could cooperate with specific techniques, such as, for example, those proposed in (Raghavan and Garcia-Molina, 2001; Palmieri et al., 2002).

REFERENCES

Arasu, A. and Garcia-Molina, H. (2003). Extracting structured data from web pages. In Proc. of *ACM SIGMOD 2003*.

Chakrabarti, S., DOM, B., Kumar, S., Raghavan, P., Rajagopalan, S., Tomkins, A., Gibson, D., and Kleinberg, J. (1999a). Mining the web's link structure. *Computer*, 32(8):60–67.

Chakrabarti, S., van den Berg, M., and Dom, B. (1999b). Focused crawling: a new approach to topic-specific Web resource discovery. *Computer Networks (Amsterdam, Netherlands)*, 31(11–16):1623–1640.

Crescenzi, V. and Mecca, G. (2004). Automatic information extraction from large web sites. *Journal of the ACM*, 51(5).

Crescenzi, V., Mecca, G., and Merialdo, P. (2001). ROADRUNNER: Towards automatic data extraction from large Web sites. In *VLDB 2001*.

Crescenzi, V., Merialdo, P., and Missier, P. (2003). Fine-grain web site structure discovery. In Proc. of *ACM CIKM WIDM 2003*.

Dean, J. and Henzinger, M. R. (1999). Finding related pages in the world wide web. *Computer Networks*, 31:1467–1479.

Kao, H., Lin, S., Ho, J., and M.-S., C. (2004). Mining web informative structures and contents based on entropy analysis. *IEEE Trans. on Knowledge and Data Engineering*, 16(1):41–44.

Laender, A., Ribeiro-Neto, B., Da Silva, A., and J., T. (2002). A brief survey of web data extraction tools. *ACM SIGMOD Record*, 31(2).

Lerman, K., Getoor, L., Minton, S., and Knoblock, C. (2004). Using the structure of web sites for automatic segmentation of tables. *SIGMOD 2004*.

Liu, Z., Ng, W. K., and Lim, E.-P. (2004). An automated algorithm for extracting website skeleton. In Proc. of *DASFAA 2004*.

Palmieri, J., da Silva, A., Golgher, P., and Laender, A. (2002). Collecting hidden web pages for data extraction. In Proc. of *ACM CIKM WIDM 2002*.

Raghavan, S. and Garcia-Molina, H. (2001). Crawling the hidden web. In Proc. of *VLDB 2001*.

Small, H. (1973). 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):28–31.

Spertus, E. (1997). Mining structural information on the web. In Proc. of *WWW Conf. 1997*.

Van Rijsbergen, C. (1979). *Information Retrieval, 2nd edition*. University of Glasgow.

Wang, J. and Lochovsky, F. (2002). Data-rich section extraction from html pages. In *WISE 2002*.

Ziv Bar-Yossef, Z. and Rajagopalan, S. (2002). Template detection via data mining and its applications. In Proc. of *WWW Conf. 2002*.