GoGetIt!: A Tool for Generating Structure-Driven Web Crawlers

Márcio L.A. Vidal, Altigran S. da Silva, Edleno S. de Moura, João M. B. Cavalcanti
Dept. of Computer Science, Universidade Federal do Amazonas
Manaus, Amazonas, Brazil
mvidal@dcc.ufam.edu.br, alti@dcc.ufam.edu.br, edleno@dcc.ufam.edu.br, john@dcc.ufam.edu.br

ABSTRACT

We present *GoGetIt!*, a tool for generating structure-driven crawlers that requires a minimum effort from the users. The tool takes as input a sample page and an entry point to a Web site and generates a structure-driven crawler based on *navigation patterns*, sequences of patterns for the links a crawler has to follow to reach the pages structurally similar to the sample page. In the experiments we have performed, structure-driven crawlers generated by *GoGetIt!* were able to collect all pages that match the samples given, including those pages added after their generation.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Clustering, Search process*

General Terms

Algorithms, Experimentation

Keywords

Web Crawlers, Tree Edit Distance, Web Data Extraction

1. INTRODUCTION

An ever increasing number of applications on the Web are targeted at processing collections of similar pages obtained from Web sites. The ultimate goal is to take advantage of the valuable information these pages implicitly contain to perform such tasks as querying, searching, data extraction, data mining and feature analysis. For some of these applications, notably for searching, the criteria to determine when a page is to be present in a collection are related to the page contents, e.g., words, phrases, etc. However, there are other important situations in which the features of the inner structure of the pages provide better criteria to define a collection than their contents.

Consider an application that requires collecting pages containing information about Jazz artists whose pages are available in the *E-Jazz* Web Site[1]. To define a set of content-related features that encompasses all the artists would be a hard task, since artists are usually related to some jazz style or to a musical instrument and there will be several distinct styles and instruments to be considered. On the other hand, there will also be non-artist related pages that share a same subject with several artist pages.

[1]http://www.ejazz.com.br

In cases such as these, representing the pages to be collected using features related with contents is not appropriate. Motivated by this problem, we propose a new Web crawling technique that is based on the structure of the Web pages instead of on their content. While many works in the literature have addressed the issue of content-driven Web crawling, the use of the structure of the pages as a criterion to guide the traversal of crawlers has been almost neglected. Nevertheless, this is an interesting alternative to solve practical problems in several important Web data management applications. The main application we consider is to automatically provide data-rich Web pages for Wrappers, which generally rely on structural patterns for performing extraction of implicit data [1]. However, other applications require collections of pages with similar structure. This is the case of structure-aware Web page searching and querying; building of digital libraries; and Web usage mining.

We present *GoGetIt!*, a tool for generating structure-driven crawlers that requires a minimum effort from users, since it relies on a handful of examples (usually one) of the pages to be fetched. To accomplish this, given a sample page and an entry point to a Web site, the tool greedily traverses the Web site looking for *target* pages, i.e., pages that are structurally similar to the sample page. Next, it records all paths that lead to target pages and generates a *navigation pattern* [2] which is composed by sequences of patterns of links a crawler has to follow to reach the target pages. Finally, the tool generates a crawler based on these patterns. From this point on, the crawler can be used to fetch pages that are structurally similar to the sample page, even if new similar pages are added later.

2. GENERATION OF STRUCTURE-DRIVEN CRAWLERS

In this section we provide an overview of the process of semi-automatically generating structure-driven crawlers as performed by the *GoGetIt!* tool. We begin by illustrating the functioning of the tool by means of a simple example and then we proceed to discuss the two phases involved in the process: *site mapping* and *navigation pattern generation.*

Through this section, we will illustrate some of the concepts we use by taking the *E-Jazz* Web site as an example. The site features information on jazz styles, instruments, recordings and artists. We concentrate on the artist-related portion. The overall structure of the site is presented in a simplified form in Figure 1.

In Figure 1, pages are labeled for reference in the discussion that follows. Assume that all pages labelled $0/2/i$

($1 \leq i \leq k$) are similar to page 0/2/0, i.e., they all contain links to artists pages. We call these pages, *artist hub pages*. Also, assume that all pages labelled $0/2/i/j$ ($1 \leq i \leq K, 1 \leq j \leq K_i$) are similar to page 0/2/0/0, i.e., they are all *artist pages*.

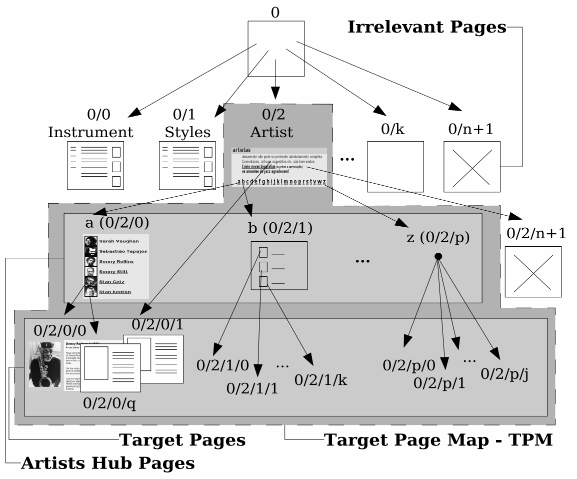

Figure 1: Overall structure of the E-Jazz Web site

Suppose we want to fetch all artist pages available on the *E-Jazz* Web site. To accomplish this task, a crawler would begin its traversal at page 0/2. Next, the crawler would have to access all artist hub pages and finally, it would fetch all artist pages.

Notice that new artist pages can be added to the Web site. This implies that some artist hub pages are often updated to include new links to these new pages. Thus, these new links and pages will have to be taken into account in future crawling processes.

To generate a crawler capable of accomplishing such a task, the *GoGetIt!* tool requires two parameters as input: an URL indicating a sample page to serve as an example for the pages to be fetched and another URL indicating an entry point for the site where these pages have to be found. In our example, page 0/2/0/0 would serve as the sample page, while page 0/2 will serve as the entry point.

The creation of a crawler with the *GoGetIt!* tool is accomplished in two phases: *site mapping* and *navigation pattern generation*.

2.1 Site Mapping

In the site mapping phase, the tool traverses all paths starting from the entry point and looking for any target pages it can find on its way. A page is considered as a target when it is structurally similar to the given sample page. Notice that, the traversing is limited to the same Web site. Every path from the entry point that lead to a target page is recorded. Thus, if we consider a Web site as a graph, the site mapping phase generates as output a *minimum spanning tree* where all leaves are target pages. This tree is called *Target Pages Map (TPM)*. In Figure 1 the output of the mapping corresponds to the tree delimited with the label *TPM*.

2.2 Navigation Pattern Generation

In the second phase, navigation pattern generation, the goal is to create a generic representation of the TPM. This generic representation, is a list of regular expressions, where each regular expression represents the links in a page the crawler has to follow to reach the target pages. This generic representation is called a *Navigation Pattern (NP)*. A NP is meant to guide a crawler in the traversal of a site to fetch the target pages. Thus, it corresponds to a single path into the TPM. However, as many paths that lead to target pages can exist, we choose the one that potentially leads to the greatest number of target pages. Notice that, if correctly generated, regular expressions in the NP will meet our requirement of accounting for new links added to the pages in the site, even after the mapping phase has been completed, since it uses regular expressions to identify links that lead to target pages.

As an example, Table 1 shows a very simple but real navigation pattern for the E-Jazz Web site. In this table, expression E_1 is applied to the entry page. It matches only links that lead to artist hub pages. Next, the expression E_2 is applied to each artist hub page and matches all the links in these pages that lead to all the artist pages and only to these pages. To generate this navigation pattern, the tool did not require any user intervention other than providing the sample page and the entry point.

Table 1: Example of a simple navigation pattern. A Perl-like syntax is used.

Exp.	Regular Expression
E_1	www.ejazz.com.br/artistas/default_[a-zA-Z]+
E_2	www.ejazz.com.br/detalhes-artistas.asp?cd=d+

3. SUMMARY OF EXPERIMENTS

Experiments we have performed involving several crawling sessions over 11 real Web sites, structure-driven crawlers generated by *GoGetIt!* were able to collect almost all pages that matched the samples given, including those pages added long after their generation, achieving 100% precision and at least 95% recall in all cases.

4. CONCLUSION AND REMARKS

The structure-driven approach we use is complementary to the traditional content-driven approach, in the sense that it is more suited for sites that are data intensive and, at the same time, present regular structure. This means that our new method is the best option for a restricted set of crawling tasks. However, it is important to notice that this kind of data intensive Web sites is becoming more popular as the Web grows.

5. ACKNOWLEDGEMENTS

This work is partially supported by projects GERINDO (CNPq/CT-INFO 552.087/02-5) and SIRIAA (CNPq/CT-Amazônia 55.3126/2005-9) and individual grants by CNPq (303032/2004-9 Altigran S. da Silva) (303576/2004-9 Edleno S. de Moura) and FAPEAM (Marcio Vidal). This research is also sponsored by UOL (www.uol.com.br), through its "UOL Bolsa Pesquisa" program, proc. num. 200503301456.

6. REFERENCES

[1] D. de Castro Reis, P. B. Golgher, A. S. da Silva, and A. H. F. Laender. Automatic web news extraction using tree edit distance. In *Proc. of the 13th Intl. Conf. on World Wide Web*, pages 502–511, 2004.

[2] J. P. Lage, A. S. da Silva, P. B. Golgher, and A. H. F. Laender. Automatic generation of agents for collecting hidden web pages for data extraction. *Data an Knowledge Engineering*, 49(2):177–196, 2004.