Crawling the Web for Structured Documents

Julián Urbano, Juan Lloréns, Yorgos Andreadakis and Mónica Marrero
University Carlos III of Madrid
Department of Computer Science
jurbano@inf.uc3m.es llorens@inf.uc3m.es gand@ie.inf.uc3m.es mmarrero@inf.uc3m.es

ABSTRACT
Structured Information Retrieval is gaining a lot of interest in recent years, as this kind of information is becoming an invaluable asset for professional communities such as Software Engineering. Most of the research has focused on XML documents, with initiatives like INEX to bring together and evaluate new techniques focused on structured information. Despite the use of XML documents is the immediate choice, the Web is filled with several other types of structured information, which account for millions of other documents. These documents may be collected directly using standard Web search engines like Google and Yahoo, or following specific search patterns in online repositories like SourceForge. This demo describes a distributed and focused web crawler for any kind of structured documents, and we show with it how to exploit general-purpose resources to gather large amounts of real-world structured documents off the Web. This kind of tool could help building large test collections of other types of documents, such as Java source code for software-oriented search engines or RDF for semantic searching.

Categories and Subject Descriptors
H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing.

General Terms
Design, Management, Standardization.

Keywords
Crawling, structured retrieval, XML.

1. INTRODUCTION
The widespread acceptance of XML as a means to represent library catalogues, financial records or any other kind of information with structure, led to the necessity of building XML test collections to evaluate new techniques developed specifically for XML-like documents, and INEX appeared as the forum for researchers to join and carry out these evaluations [1]. However, many XML documents have their own well grounded and well established format, such as XSD, RDF or RSS. Also, there are many other types of structured information that do not follow a XML-like syntax, such as Java source code files or SQL database creation scripts. All these kinds of structured information can be easily found in the Web, and they account for millions of documents that are not being considered by the Structured Information Retrieval community.

Back in the days, a common keyword-based search proved to be sufficient when looking for software artifacts such as source code files or database creation scripts. Indeed, this line of work has been followed by general purpose search engines, like Google Code Search[1], Codase[2] or Koders[3]. However, these communities are evolving towards more semantic-aware information needs. Much of the software development process is nowadays based on models, where UML has become the de-facto standard. Now that everything revolves around models, structured information techniques trying to exploit the structure of the elements contained in a document, and most importantly their relations, is the key to success [2]. One of the most troublesome issues in these cases is finding artifacts to reuse [3], which are not limited to source code or libraries, but also extend to domain models, requirements specifications and various types of schema, such as DTD, XSD or SQL scripts defining a database.

In this demonstration we show a crawler to harvest structured document from the Web, according to the specific needs of particular users. Although our original focus was to download software-related documents, this crawler can be used for any kind of structured or unstructured documents, such as images.

2. WHERE TO CRAWL FROM
The Web is filled with all kinds of documents, from the simplest plain text files to the more advanced multimedia formats. Structured documents are no exception, and thousands of XML or XSD documents, for example, can be easily found. Two general sources can be used for this purpose: general keyword-based search engines like Google or Yahoo, and online repositories such as SourceForge.

2.1 General Keyword-based Search Engines
The usual keyword-based techniques used in general purpose web search engines do not have enough power to represent these relationship-based information needs, let alone to retrieve relevant documents. Nonetheless, they seem to be a very good starting point to crawl the Web when looking for certain types of structured documents.

Using Google Web Search, for instance, one could issue a query making use of the *filetype* operator to indicate what type of documents to look for. Adding some more information particular to each document type of relevance, we can find ourselves before an enormous amount of relevant documents. For example, when looking for SQL database creation scripts about bank accounts, one could issue a query like "+create +table account filetype:sql" and get about 2000 results. Table 1 shows the approximate total number of results supposedly offered by Google and Yahoo for queries regarding different common file types.

CIKM'10, October 25–29, 2010, Toronto, Ontario, Canada.
ACM 978-1-4503-0099-5/10/10.

[1] http://www.google.com/codesearch
[2] http://www.codase.com
[3] http://www.koders.com

An immediate problem using this method is that many of the results returned do not correspond to structured documents, but to web pages pointing to them or pointing to no relevant document at all. Some may even contain the information embedded in the HTML page itself, or with the keywords used appearing in a non-relevant part of the document, such as comment lines. Besides, web servers do not really agree every time when telling the mime-type of a document, which is a problem known to the TREC community for the GOV2 corpus, or the fact that some document types are hierarchical (e.g. XSD is a particular type of XML). Table 1 reflects how many of the first 20 documents retrieved were actually relevant in terms of their file type. This makes it imperative to develop appropriate filters in each case not to end up modeling garbage results.

Table 1. Number of results for different types of document.

Type	Google (P@20)	Yahoo (P@20)
XML	25M (0.85)	238K (0.8)
DTD	48K (0.95)	48K (1)
XSD	134K (1)	181K (1)
SQL	104K (1)	152K (0.95)
JAVA	3M (1)	1.6M (1)

One more issue when crawling documents off general purpose search engines is that they usually return just about 1000 documents at most, even if there were millions. One way around this is to use resources like word listings, to issue queries with a couple of different terms each time to try covering different domains. This way, the whole set of documents of a particular type should eventually be returned, as the union of all the result sets generated with these terms.

2.2 Online Repositories

Another valuable source for such documents is websites dedicated to open-source projects, like SourceForge[4] or Google Code Hosting[5]. These sites present a clear structure, which makes them easy to navigate and crawl when looking just for particular documents. They comprise thousands of projects, each of which has a dedicated page usually linking to source code packages and other artifacts in a very particular way. These linking patterns can be exploited by a focused crawler to go directly for the important files and ignore the non-relevant ones [4].

3. IMPLEMENTATION

We developed a multi-threaded prototype crawler to gather structured documents off the Web. It has been implemented with the Microsoft .net framework 3.5 SP1 and the free version of SQL Server Express 2005. It contains a core module, in charge of distributing tasks to several threads, according to a very extensive and adaptable configuration file that allows the crawler to focus one way or the other. This configuration file also has information as to what types of documents the crawler should treat, and how to do so. Each of these document types has several information associated, such as possible MIME-types, file extensions, file sizes and what we called processors.

These processors are focused on a particular type of document, and they are in charge of the screening process that discards non-appropriate files. For example, files for which the server returned an XSD MIME-type but turned out to be plain text, or just XML files not well-formed. These processors could also turn into indexers for a search engine focused on these particular document types. For instance, a SQL processor could parse a database creation script, extract the table names, fields and relationships, and index them for a SQL retrieval engine.

A very special processor is the one in charge of processing HTML pages (i.e. the traditional crawler). It offers several configuration parameters dependent on the domain, which make the crawler as focused as possible. For example, we can have it collect only certain URL patterns and navigate through others, depending on the host being visited. Therefore, when using Google Web Search we could indicate to collect the URLs corresponding to the results and navigate through the pagination links. In the case of SourceForge, we could indicate the crawler to navigate the pages corresponding to categories, and collect the URLs corresponding to the projects' homepages. Then, the crawler navigates towards the download pages and collects only the files of our interest, such as Java source code, while linking it to the project it belongs to.

The core module and most of the functionality is detached from the GUI, so it can be ran and managed programmatically without the need for user interaction. In addition, the crawler can scale up with more machines, which can be hot-plugged in and collaborate with no additional configuration. New document types can be added with virtually no effort, and the crawler can be easily extended to specialize certain processes or add new ones. To allow this, the crawler is designed to work on-the-fly with different forms of meta-information associated to the documents downloaded, with no need to change the database schema or re-compile the source code.

The crawler is freely available for research and educational purposes from the UML Models website at http://www.umlmodels.org.

4. DEMONSTRATION PLAN

In this demonstration, we show how to use our crawler and the techniques explained above to gather structured documents using general purpose search engines like Google. We also show how to use online repositories, such as SourceForge, to download software-related files of interest. We also describe and demonstrate the great number of configuration parameters that can be set up to satisfy particular user needs.

ACKNOWLEDGEMENTS

We acknowledge the Spanish National Plan of Scientific Research, Development and Technological Innovation, which has funded this work through the research project TIN2007-67153.

REFERENCES

[1] N. Gövert and G. Kazai. Overview of the INitiative for the Evaluation of XML retrieval (INEX) 2002. *INEX Workshop*, pages 1-17, 2002.

[2] S. Betsi, M. Lalmas, et al. User Expectations from XML Element Retrieval. *International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 611-612, 2006.

[3] W.B. Frakes and T. Pole. An Empirical Study of Representation Methods for Reusable Software Components. *IEEE Transactions on Software Engineering*. 20(8): 617-630, 1994.

[4] Y. Xiong, P. Luo, et al. OfCourse: Web Content Discovery, Classification and Information Extraction for Online Course Materials. *ACM International Conference on Information and Knowledge Management*, pages 2077-2078, 2009.

[4] http://sourceforge.net

[5] http://code.google.com/hosting