

Cooperative crawling

Marina Buzzi

IIT-CNR

Marina.Buzzi@iit.cnr.it

Abstract

Web crawler design presents many different challenges: architecture, strategies, performance and more. One of the most important research topics concerns improving the selection of “interesting” web pages (for the user), according to importance metrics. Another relevant point is content freshness, i.e. maintaining freshness and consistency of temporary stored copies. For this, the crawler periodically repeats its activity going over stored contents (re-crawling process). In this paper, we propose a scheme to permit a crawler to acquire information about the global state of a website before the crawling process takes place. This scheme requires web server cooperation in order to collect and publish information on its content, useful for enabling a crawler to tune its visit strategy. If this information is unavailable or not updated the crawler still acts in the usual manner. In this sense the proposed scheme is not invasive and is independent from any crawling strategy and architecture.

1. Introduction

Web crawling is an important research issue. Crawlers are SW components which visit portions of web trees, according to certain strategies, and collect retrieved objects in local repositories. Usually a crawler starts from a set of “interesting” URLs, collects new URLs from pages visited and continues the exploration until resources are available [2]. Search engines use crawlers to collect local copies of web pages [7].

A very important problem is keeping the local collection “fresh”, which means a high probability that one stored copy is equal to the original object. At regular intervals the crawler repeats the inspection of web pages in order to refresh modified contents. Many strategies for optimizing re-crawling have been studied but the variety of different contexts and the very dynamic nature of the WWW make it difficult to model the web effectively. Web pages have a life cycle: they are born, change and can also disappear; they change with very different update rates, which can vary over time, thus becoming difficult to model effectively. In addition, the freshness of stored copies is influenced by many factors such as type of retrieval (steady/batch), modality of update (in-place/shadowing), visit frequency (variable/fixed), object-replacing policy [5].

Previous studies showed that, in order to maintain a certain degree of content freshness, it is more effective to

update low-changing pages than very dynamic ones [2]. Usually mathematical functions are used to approximate the real scenarios. For instance, page update frequency has been modeled with statistical functions such as Poisson or Pereto distributions. On the other hand, these models are not exact, and re-crawling unmodified web pages implies a cost in terms of network bandwidth and resource usage; consuming little for a single page, it becomes considerable on a large scale. Wolf et al. studied the problem of crawling frequency in order to minimize the average degree of staleness of stored web pages and the probability that the user will access incorrect search engine results (due to broken links or off-topic pages) [13]. The authors modeled web updates in a general framework by using probability theory (stochastic processes) and resource allocation theory, and propose a scheme to obtain an optimal number of re-crawls for pages and times (scheduled for a set of cooperating crawlers).

Another problem is that of selecting more “interesting” objects, for the users. A search engine is aware of hot topics because it collects user queries. The crawling process prioritizes URLs according to an importance metric such as similarity (to a driving query), back-link count, PageRank or their combinations/variations [6], [2]. Recently Najork et al. showed that breadth-first search collects high-quality pages first and suggested a variant of PageRank [10]. However, at the moment, search strategies are unable to exactly select the “best” paths because their knowledge is only partial. Due to the enormous amount of information available on the Internet a total-crawling is at the moment impossible, thus, prune strategies must be applied. Focused crawling [4], [8] and intelligent crawling [1], for instance, are techniques for discovering web pages relevant to a specific topic or set of topics.

We believe that to be more effective, crawling strategies should be driven by exact information, available on the origin website, such as hot pages, file size and last update, number of links (which can be extracted from the web server log file, file system, database, and the object itself). This information, describing published data, permits a crawler to save bandwidth and resources. The idea of this work is to obtain information about published data from the web server itself in a fashion similar to the use of robot exclusion.

The robot.txt file gives directives for excluding a portion of a web site to be crawled. Analogously, a simple text file can furnish information about the freshness and popularity of published objects. This information permits a crawler to optimize its strategy for refreshing collected data as well as replacing object policy.

When a site is considered relevant (this information is elaborated with external data) a crawler can obtain local information in order to select the most important part of the web site, according to its policy.

A web site can benefit from this approach by improving its data indexing and thus its global visibility on the Internet while reducing crawling overhead, by eliminating retrieval of unchanged data. In addition, search engines can optimize the crawling process and improve the quality of their service by reducing the amount of broken links and obsolete pages.

2. Cooperative crawling

Network transfer time represents a consistent part of the crawling process, because object retrieval is needed to refresh stored contents. Our idea is to use a text file (website.txt) that can be retrieved in a single connection. This file describes website content (metadata) and furnishes information useful for improving the crawling process. It should contain its creation/update time and for each object:

- URLs/URIs (net addresses);
- Last update time;
- File size;
- Local request frequency (percentage value), to evaluate the access load of the object and discover popular paths.
- Local update frequency (expressed in percentage value).

Additional parameters may be included, if useful for aiding the crawling process. However, the important thing is to define an extensible format. This scheme can be implemented with an additional web server Module or an external program scheduled at regular intervals (for instance daily). Implementing a web service is also possible (by using XML and SOAP).

The website.txt file permits saving network bandwidth in case of unmodified pages because only one file is retrieved compared to the "conditional GET" (GET method with the If-Modified-Since request-header field) of each page, thus permitting a crawler to save resource usage and improve performance.

One problem with this approach is the trustiness of the information furnished by the websites. However, the relevance of the web site is not a locally available parameter. This means that a deviation of this data can only influence the visit of a crawler inside the web site

itself. This has no influence on the global relevance of retrieved objects.

This method can be applied regardless of the crawling strategies. If the site information file is present, the crawler can transfer them and extract information useful for speeding up the retrieving process. The quality of page content still depends on external information.

3. Discussion

Complete web crawling coverage cannot be achieved, due to the vast size of the whole WWW and to resource availability. Usually a kind of threshold is set up (number of visited URLs, level in the website tree, compliance with a topic, etc.) to limit the crawling process over a selected website. The website.txt file furnishes a picture of the website's state at a certain time (for instance it can be updated once a day). This information is available to guide search engines to store/refresh most relevant and updated web pages, thus improving quality of retrieved contents while reducing stale content and missing pages.

However some important benefits are received by the origin website. Using the website.txt file, the quality of the retrieving process can be improved in many ways. First of all the quality of search results; second the web server load (due to crawling) is restricted to the necessary one. Third, very important, visibility of hidden content can easily be accomplished. Today in fact an ever-increasing number of web sites use dynamic pages generated with script or programming languages querying database or data repository. The size of the hidden web has been estimated to be as much as 500 times greater than the public indexable Web [11]. Bringing up this information is a very relevant effort. By using the proposed scheme this information can be added to the file by using an ad-hoc SW.

Various cooperation schemes between web server and search engine have been formulated in previous works (aimed at improving data retrieval by means of meta-data, modified pages or differences) [3]. However, this proposal extends the simple use of meta-data by adding information that serves to drive the crawling process towards local hot pages (local relevance) and tune search engine refreshing policy (local update frequency). In this way, a search engine would be able to discover new relevant contents more rapidly, overcoming the limits of link-based ranking algorithms in evaluating new contents (i.e. new pages have a very low rank) as discussed in [12]. In this work authors showed that PageRank was biased against new pages and propose a variant, which takes page age into account.

From the architectural point of view, parallel and distributed crawlers can improve performance [6]. For instance, by combining distributed crawling with grid architecture the Grub Project attempts to take advantage

of idle resources on the net for continually crawling the Web [9]. However, this solution requires the installation of SW (out of the control of the local administrator) and communicating any change to the central search engine. In contrast, our approach is not invasive and can be easily adopted without any security risk. Furthermore, it can be applied regardless the crawling strategy or architecture. Companies hosting web services and applications, which are more sensitive to customer data protection, can omit access statistics and still increase the quality of stored and indexed contents.

In conclusion, adoption of the proposed scheme could bring advantages for both crawlers and websites. The important point is the definition of the data format by means of a standardization organization, such as the IETF or W3C

4. References

- [1] C. Aggarwal, F. Al-Garawi, P. Yu, "Intelligent crawling on the World Wide Web with arbitrary predicates", WWW 2001, pp. 96-105.
- [2] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, S. Raghavan, "Searching the Web", ACM Transactions on Internet Technology, Vol. 1, Num. 1, August 2001, pp. 2-43.
- [3] C. Castillo, "Cooperation schemes between a Web server and a Web search engine", LA-WWW 2003.
- [4] S. Chakrabarti, K. Punera, M. Subramanyam, "Accelerated focused crawling through online relevance feedback", WWW 2002, pp. 148-159.
- [5] J. Cho, H. Garcia-Molina, "The Evolution of the Web and Implications for an Incremental Crawler", VLDB 2000, pp 200-209.
- [6] J. Cho, H. Garcia-Molina, L. Page, "Efficient Crawling Through URL Ordering", WWW7/Computer Networks 30 (1-7): 161-172 (1998).
- [7] C. Chung, C. Clarke, "Topic-oriented collaborative crawling", CIKM 2002, pp. 34-42.
- [8] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, M. Gori, "Focused Crawling Using Context Graphs", VLDB 2000, pp. 527-534.
- [9] GRUB'S DISTRIBUTED WEB CRAWLING Project. <http://www.grub.org/>.
- [10] M. Najork, J. Wiener, "Breadth-first crawling yields high-quality pages", WWW 2001, pp. 114-118.
- [11] S. Raghavan, H. Garcia-Molina, "Crawling the hidden Web", VLDB Conference, 2001, Rome, Italy.
- [12] R. A. Baeza-Yates, F. Saint-Jean, C. Castillo, "Web Structure, Dynamics and Page Quality", SPIRE 2002, pp. 117-130.
- [13] J.L. Wolf, M.S. Squillante, P. Yu, J. Sethuraman, L. Ozsen, "Optimal crawling strategies for web search engines", WWW 2002, pp. 136-147.