Architecture for a Garbage-less and Fresh Content Search Engine

Víctor M. Prieto, Manuel Álvarez, Rafael López García and Fidel Cacheda
Department of Information and Communication Technologies, University of A Coruña, A Coruña, Spain

Keywords: Web Crawling, Web Spam, Soft-404, Crawling Refresh, Crawling Performance, Crawling Architecture.

Abstract: This paper presents the architecture of a Web search engine that integrates solutions for several state-of-the-art problems, such as Web Spam and Soft-404 detection, content update and resource use. To this end, the system incorporates a Web Spam detection module that is based on techniques that have been presented in previous works and whose success have been assessed in well-known public datasets. For the Soft-404 pages we propose some new techniques that improve the ones described in the state of the art. Finally, a last module allows the search engine to detect when a page has changed considering the user interaction. The tests we have performed allow us to conclude that, with the architecture we propose, it is possible to achieve important improvements in the efficacy and the efficiency of crawling systems. This has repercussions in the content that is provided to the users.

1 INTRODUCTION

Currently, the WWW is the biggest information repository ever built, and it is continuously growing. Due to its big size, it is indispensable to use search engines to access the information which is relevant to the user.

A search engine faces many challenges because of the quantity, the variability and the quality of the information that it has to gather. Among others, we can highlight: server/client side technologies (Raghavan and Garcia-Molina, 2001) (Bergman, 2000), Web Spam (Gyongyi and Garcia-Molina, 2004) (Fetterly et al., 2004), Soft-404 pages (Bar-Yossef et al., 2004) which are those that return a 200 OK HTTP code instead of a 404 one, with or without content, repeated content (Kumar and Govindarajulu, 2009), content refresh (Brewington and Cybenko, 2000) (Cho and Garcia-Molina, 2003), bad quality content, etc. These challenges can be summarized in one: processing the biggest number of web documents of the best quality, using the lowest quantity of resources and in the shortest time.

There are studies that try to improve the performance of the search engines and their crawling systems by proposing different types of architectures, and trying to reduce the resources that are necessary in certain tasks such as disk access or URL caching. However, to the best of our knowledge, there is not a paper that proposes a global search engine architecture that improves the performance by focusing on avoiding processing those pages whose content is not appropriate to be indexed and by optimizing the refresh time of pages with relevant content. According to Ntoulas *et al.* (Ntoulas and Manasse, 2006), 70% of the pages of the .biz domain are Spam, and the same happens with 20% of .com and 15% of .net. Moreover, considering our own studies that make random requests to existing domains, we have stated that 7.35% of the web servers return a Soft-404 page when a request to an unknown resource is made.

If these "garbage" pages can be detected, we can protect both the users, who are wasting their time and money, and also the companies that develop search engines. The latters are very affected, since they do not only lose prestige when they show Spam among the results, but they are also wasting resources, and therefore money, in analysing, indexing and showing pages that should not be shown. Not protecting any of these parties means an economic loss.

We have also stated that, for domains whose Page Rank is between 0 and 3, Google has not been up to date 60.42% of the times, Yahoo! 70.93% and Bing 66.40%. For domains with PageRank between 3 and 6, Google has been out of date 50.52% of the times, Yahoo! 72% and Bing 70.17%.

This paper presents the architecture of a search engine whose processes are optimised to avoid irrelevant pages and to refresh the contents in the most appropriate moment. This will minimise unnecessary

M. Prieto V., Álvarez M., López García R. and Cacheda F..
Architecture for a Garbage-less and Fresh Content Search Engine.
DOI: 10.5220/0004167203780381
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2012), pages 378-381
ISBN: 978-989-8565-29-7

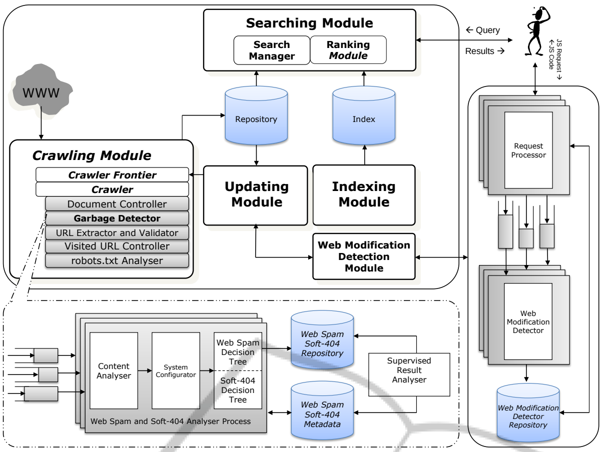

Figure 1: Architecture proposed for a Search Engine and its Crawling System.

refreshing at the same time as their level of obsolescence. For that purpose, the system incorporates a module for detecting "garbage" pages (Spam and Soft-404) and a module for detecting changes which focuses on user accesses, using a collaborative architecture.

This architecture includes the Spam and Soft-404 detection techniques published by Prieto *et al.* (Prieto et al., 2012). We also propose a system to detect "real-time" changes in Web pages too. This, along with the refreshing policies (in which current search engines are based), will improve significantly the update of web resources in the search engine.

2 ARCHITECTURE OF THE SEARCH ENGINE

Figure 1 shows the architecture we propose for a search engine, along with the modules for detecting web "garbage" and for detecting modifications in the pages. In an introductory overview of the system, we can appreciate that the search engine consists of the following elements:

- Crawling Module: it consists of a group of crawlers that are in charge of traversing the Web and creating a repository with the contents found.
- Repository: set of downloaded pages.
- Updating Module: its objective is to keep the repository up to date. This module communicates with the modification detection system that is explained in section 2.2.
- Indexing Module: it is responsible for extracting the keywords of each page and for associating them the URLs the system has found.
- Index: it represents the set of indices that have been generated by the indexing module. These indices will be used to solve the queries sent by the user.
- Query Manager: its mission is to process the queries and to return those documents that are relevant for them.
- Ranking Module: it is in charge of sorting the results the Query Manager has returned according to the relevance of the documents.

Focusing on the architecture of the crawling module, we can highlight the following modules:

- Document Controller: it is in charge of controlling that the content of a web document is not repeated or it is not very similar to others. In this case it should not be processed and indexed again.
- Garbage detector: it is in charge of detecting Spam and Soft-404 pages. It will be explained in section 2.1.
- URL extractor and validator: it is responsible for processing the content of each page and the extraction of new valid links.
- Visited URL Controller: its mission consists in checking if a URL has already been processed.
- Robot.txt Analyser: following the robots exclusion protocol, this module determines which pages must be indexed and/or visited.

2.1 Garbage Detection Module

The garbage detection module has the goal of detecting both Web Spam and Soft-404 pages in order to avoid their processing and indexing. It is based on decision tree algorithms that use as input the result of applying different heuristics that try to partially characterize both Web Spam and Soft-404 pages. Regarding Web Spam, in contradistinction to other approaches in the literature, this module is designed to detect all kinds of Web Spam: Cloaking (Wu and Davison, 2005a), Link Farm (Wu and Davison, 2005b), Redirection Spam (Chellapilla and Maykov, 2007) and Content Spam (Fetterly et al., 2005). It is based in the techniques introduced in (Prieto et al., 2012). Regarding Soft-404 pages, this module detects parking pages as well as pages that notify 404 errors, but whose web server does not send the corresponding HTTP code. This allows the system to use fewer resources than the used by other algorithms in the literature, which send several random requests to the domain in order to check the differences between the normal pages and the Soft-404 pages.

This module uses C4.5 (Quinlan, 1996) as the decision tree algorithm, along with "bagging" and "boosting" techniques to improve the results. The system contains a set of decision trees to detect Web Spam and another one for Soft-404 pages, with each tree corresponding to a configuration that employs more or less resources (maximizing the detection or the performance respectively). The process of this module for each new page is the following:

- The Content Analyser extracts the results of each heuristic.
- The System Configurator chooses the decision trees that will be used in each case. This election depends on the detection level that has been previously configured and the resources that are available for the detection module of that level.
- Depending on the results obtained by the heuristics we have applied, the Web Spam and Soft-404 decision trees will establish the type of page.
- If the page is considered Spam or Soft-404, its processing is stopped and the domain will be stored in the Spam and Soft-404 Repository, which will be queried by the URL Validator. If any link points to a Spam domain the page will be discarded. If the link is part of a domain that returns Soft-404 pages, this page must be marked as "possible" Soft-404 and analysed later.
- Supervised Result Analyser allows recalculating the decision tree from time to time in order to improve the quality of the detection. Feedback is received by means of manual labelling of Spam pages.

Finally, this module communicates with the Document Controller, since the first step in the Garbage Detection Module consists in knowing whether a document has already been processed or whether it has already been detected as Spam or Soft-404.

2.2 Web Modification Detection Module

This section describes the architecture of the module we propose to improve the information about modifications employed by the Updating Module of the search engine. This module is based on a collaborative architecture similar to the one used in "Google Analytics"[1] for obtaining data from the client that accesses a page. In our case, the navigation of a user through a web resource is used as a changes notification agent. This module is shown on the right side of Figure 1. The creator of a web site should embed a little JavaScript code in the pages that wish a "real time" updating by the search engines.

Due to the JavaScript code that has been embedded in the pages, for each request of a user, the "Request Processor" will check in the repository the last date which contains information about that page. If the information is new enough, we will not send the JavaScript code, so the client's browser will not send anything to the server. Otherwise, the code we send will make a MD5 summary of several parts of the page and it will send it to the web modification detection module, which will queue the request. Finally, the "Web Modification Detector" will traverse the queues of pending operations, choosing the most updated information and updating the repository.

The system we have designed is highly scalable since a big part of the processing is done in the users' web browsers. Moreover, in situations of high load or attacks against our system (DDOS - Distributed Denial of Service), the module will not send the JavaScript, and, in case that the load is big enough to affect the system for a long time, the system will wait, not receiving information for that period of time, but not affecting the final user either.

The Updating Module queries the Web Modification Detection Module to know what pages have changed. Then, the latter decides whether to re-crawl the page by considering other data (relevance, frequency of change, type of page, etc.).

Current re-crawling policies consist in polling web sites more or less frequently to detect changes. This approach has two problems: (1) visiting pages that have changed, which implies wasting resources, and (2) not having the repositories up to date at "real time". However, in our approach the crawler is indicated when a page has changed, so our repository will be more updated, with the consequential improvement of the quality of the indices that the search en-

[1] http://www.google.com/analytics/

gines are using. The system will also improve performance use since resources will only be used when it is necessary.

3 DISCUSSION

According to some studies existing in the literature and others we have performed, we have observed that Spam and Soft-404 pages represent 27.35% of the content of the .com domain. In addition, we have checked that the resources a search engine has indexed are obsolete more than 50% of the time, which affects to the quality of the results, too.

In the literature, there are not any search engine architecture that improve their performance and the quality of their results by detecting and ignoring "garbage" content, and therefore, reducing the number of used resources. The architecture of the web search engine we have proposed in this paper contains a module that is in charge of detecting the web "garbage", to improve the quality of the pages we index and process. Furthermore, a module for detecting modifications provides the system with information about the changes in the pages when the users navigate through a site, which helps to improve the refresh policies. The results we have obtained point out that a crawler that uses the Web Spam detection module and the Soft-404 detection module would avoid processing 22.37% of the resources in the worst case and 26.62% in the best one, which is the practical totality of the aforementioned 27.35% of "garbage pages". What is more, the modification detection module would allow the search engine to know the exact change moment, not wasting resources in returning to a page that has not changed and allowing the system to decide the best moment to re-crawl it.

Our future work consists in developing this architecture completely and assessing it in real environments. In parallel, we will develop the modules for a distributed architecture.

ACKNOWLEDGEMENTS

This work was supported by the Spanish government (TIN 2009-14203).

REFERENCES

Bar-Yossef, Z., Broder, A. Z., Kumar, R., and Tomkins, A. (2004). Sic transit gloria telae: towards an understanding of the web's decay. In *Proceedings of the 13th international conference on World Wide Web*, WWW '04, pages 328–337, New York, NY, USA. ACM.

Bergman, M. K. (2000). The deep web: Surfacing hidden value.

Brewington, B. and Cybenko, G. (2000). How dynamic is the web? pages 257–276.

Chellapilla, K. and Maykov, A. (2007). A taxonomy of javascript redirection spam. In *Proceedings of the 3rd international workshop on Adversarial information retrieval on the web*, AIRWeb '07, pages 81–88, New York, NY, USA. ACM.

Cho, J. and Garcia-Molina, H. (2003). Estimating frequency of change. *ACM Trans. Internet Technol.*, 3:256–290.

Fetterly, D., Manasse, M., and Najork, M. (2004). Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004*, WebDB '04, pages 1–6, New York, NY, USA. ACM.

Fetterly, D., Manasse, M., and Najork, M. (2005). Detecting phrase-level duplication on the world wide web. In *In Proceedings of the 28th Annual International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 170–177. ACM Press.

Gyongyi, Z. and Garcia-Molina, H. (2004). Web spam taxonomy. Technical Report 2004-25, Stanford InfoLab.

Kumar, J. P. and Govindarajulu, P. (2009). Duplicate and near duplicate documents detection: A review. *European Journal of Scientific Research*, 32:514–527.

Ntoulas, A. and Manasse, M. (2006). Detecting spam web pages through content analysis. In *In Proceedings of the World Wide Web conference*, pages 83–92. ACM Press.

Prieto, V. M., Álvarez, M., and Cacheda, F. (2012). Analysis and detection of web spam by means of web content. In *Proceedings of the 5th Information Retrieval Facility Conference*, IRFC '12.

Quinlan, J. R. (1996). Bagging, boosting, and c4.5. In *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 725–730. AAAI Press.

Raghavan, S. and Garcia-Molina, H. (2001). Crawling the hidden web. In *Proceedings of the 27th International Conference on Very Large Data Bases*, VLDB '01, pages 129–138, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Wu, B. and Davison, B. D. (2005a). Cloaking and redirection: A preliminary study.

Wu, B. and Davison, B. D. (2005b). Identifying link farm spam pages. In *Special interest tracks and posters of the 14th international conference on World Wide Web*, WWW '05, pages 820–829, New York, NY, USA. ACM.