Soft-404 Pages, A Crawling Problem

Víctor M. Prieto, Manuel Álvarez, Fidel Cacheda
Comunications and Information Technologies Department
Facultade de Informática
Universidade da Coruña (University of A Coruna)
Campus de A Coruña, 15071 (A Coruña), Spain
{victor.prieto,manuel.alvarez, fidel.cacheda}@udc.es

Journal of Digital Information Management

ABSTRACT: During its traversal of the Web, crawler systems have to deal with multiple challenges. Some of them are related with detecting garbage content to avoid wasting resources processing it. Soft-404 pages are a type of garbage content generated when some web servers do not use the appropriate HTTP response code for death links making them to be incorrectly identified. Our analysis of the Web has revealed that 7.35% of web servers send a 200 HTTP code when a request for an unknown document is received, instead of a 404 code, which indicates that the document is not found. This paper presents a system called Soft404Detector, based on web content analysis to identify web pages that are Soft-404 pages. Our system uses a set of content-based heuristics and combines them with a C4.5 classifier. For testing purposes, we built a Soft-404 pages dataset. Our experiments indicate that our system is very effective, achieving a precision of 0.992 and a recall of 0.980 at Soft-404 pages.

Subject Categories and Descriptors:
H.2.8 [Database Applications]: Data Mining; H3.1 [Content Analysis and Indexing]: Abstracting Methods; H.3.3 [Information Search and Retrieval]: Information Filtering

General Terms: Design, Algorithms, Performance

Keywords: Soft-404 Error, Web Spam, Web Decay, Link Analysis, Data Mining, Statistical Properties of the Web

Received: 11 November 2012, Revised 13 March 2013, Accepted 19 February 2014

1. Introduction

Currently, the WWW constitutes the biggest repository of information ever built, and it is continuously growing. According to the study presented by Gulli and Signorini [17] in 2005, the Web consists of thousands of millions of pages. Three years later, in 2008, according to Official Blog of Google[1], the Web contained 1trillion of unique URLs. Due to its large size, search engines are essential tools for users who want to access relevant information for a specific query. The set of tasks performed by a search engine is very complex. Therefore, there are many studies that analyse the architecture of each of the main parts of a search engine.

Baeza-Yates and Ribeiro-Neto discuss in [3] the architecture of a search engine. In short these are its main components:

- **Crawling Process [30]:** Responsible for downloading pages from the Web automatically and storing them in a repository.

- **Indexing Process:** Responsible for creating an index, from the repository pages. This index allows performing the user queries. The most used index structure is an inverted index composed of all the distinct words of the collection and, for each word, a list of the documents that contain it.

- **Retrieval & Ranking Processes:** The first one retrieves

[1] http://googleblog.blogspot.com.es/2008/07/we-knew-web-was-big.html

documents that satisfy a user query. Finally, the documents are ranked and those most relevant for the user are returned.

The different tasks of the search engines involve significant challenges in the treatment of vast amounts of information. Among these challenges, specific aspects can be highlighted, such as the technologies used in web pages to access to data, both in the server-side [34] or in the client-side [8]; or problems associated with web content as Web Spam [18] or repeated contents [26], etc. We want to highlight the article by Cambazoglu and Baeza-Yates [10], which presents these challenges through an architectural classification, starting from a simple single-node search system and moving towards a multi-site web search architecture. In this study we will focus on improve the crawling process to do it more efficiently.

There are several studies that try to improve the performance of the crawler, for example, changing its architecture by means of distributed systems [24] [1] (explained in more detail in Section 2). However, there is a complementary way to improve the efficiency of crawling systems: by minimizing the use of resources [20].

That is, for instance, by reducing the garbage content they have to process. This garbage consists of: a) Web Spam [18] [15], b) incorrectly identified death links (Soft-404 pages [5], parking pages); c) duplicate contents [26], etc.

In this article we study a method to avoid part of the garbage contents. More precisely, we focus on Soft-404 pages, the set of web pages that some web server returns with a 200 HTTP code when a nonexistent resource is requested. In Section 6 we will show that many of these pages come from parking domains. These pages have no content or are generated automatically, with many anchors and advertisements, to obtain revenue. This causes a crawling system to believe that the page exists and that it must be processed and indexed, with the consequent loss of resources.

So, protection mechanisms should be created for a) the final users who wasting their time and perhaps their money and b) the companies that provide search engines. The latter are very much affected since they not only lose prestige when Soft-404 pages are shown among their results, but they are also wasting money and resources in analysing and indexing these types of pages.

We propose a system, called Soft404 Detector, based on web content analysis to identify web pages that are Soft-404 pages. The idea is to include the proposed system as a module of a crawling system. Thus, the crawler will avoid storing, processing and indexing these types of pages. In summary, our system contributes to detect part of the "*garbage*" pages (Soft-404) presented on the Internet.

The structure of this article is as follows. Section 2 contains comments on the importance of Soft-404 pages and the existing detection methods. This section also analyses studies about resource usage in critical environments, such as crawling systems. Section 3 shows the presence of Soft-404 pages on the Web and the need to propose solutions for their detection. Section 4 includes a discussion on the proposed heuristics, which characterize Soft-404 pages, used later by the classification system. Section 5 explains the method for Web Spam detection that we created from the combination of different heuristics. Section 6 analyses the results we obtained by applying the distinct heuristic sets. We also compare the effectiveness and performance of our system and the systems proposed by Bar- Yossef et al. [5] and Lee et al. [27]. Finally, Sections 7 and 8 present our conclusions and possible future works, respectively.

2. Related Works

The Web cannot be defined and characterized in a simple way. Numerous studies show its complexity [37]. Its continuous growth, combined with mass access to the data it contains, have caused many problems for search engines, and at the same time, for crawling processes. These problems are wasting resources and worsening the system performance.

Some articles have tried to improve crawling efficiency by enhancing their architecture. A study that tries to improve the efficiency of the crawlers is [1], where Akamine et al. propose an efficient architecture for a crawler and a search engine. Another article which studies the efficiency of crawling systems is that by Hafri and Djeraba [20]. In this article the authors discuss the main problems of a crawler, such as URL caching, detection of similar documents, extraction and standardization of URLs, and so on. Finally, an article by Kimball et al. [24] proposes to use MapReduce (Hadoop) and GFS (Google File System), an architecture for the execution of algorithms and data storage, respectively.

A different way to improve the efficiency of crawling systems is by preventing the waste of resources in the treatment of Web Spam, Link Farms or Soft-404 pages.

Some important articles study Web Spam in general, like the one by Henzinger et al. [22], who discuss the importance of the phenomenon and the quality of the results that search engines offer. Gyöngyi and Garcia-Molina [18] propose a taxonomy of Web Spam, too. The survey by Castillo and Davison [11] shows further reference on this topic. However, most of them focus on some of the main Web Spam types: Content Spam, Cloaking, Redirection Spam and Link Spam. Content Spam is a technique based on the modification of the content with the purpose of simulating more relevance to search engines. Fetterly et al. [15], highlight the importance of analyzing the content and their properties in order to detect

Web Spam. Fetterly et al. [16] also discuss the special interest of the "*cut- and-paste*" content between Web Pages in order to detect Web Spam. Hidalgo [23] and Sahami et al. [36] propose similar techniques for Email Spam detection, combining content analysis and classifiers (e.g.: C4.5). Another relevant technique is Cloaking, which is based on showing different copies of each web page to the crawlers and the web browsers. Gyöngyi and Garcia-Molina [18] explain some existing techniques. Wu and Davison [38] propose a detection method which is based on calculating the common words between three copies of a web page. Link Spam has been studied by Wu and Davison [39] and Gy¨ongyi and Garcia-Molina [19], who presented some methods for detecting link farms. Amitay et al. [2] also analyse the importance of the properties of connectivity among pages in order to identify them. On the other hand, studies like the one by Zhang et al. [40], propose methods to prevent illegitimate changes in the PageRank by means of link farms. Benczur et al. [7] introduce the idea of TrustRank, which consists in starting from a set of "*clean*" pages that have been analysed previously, extracting links from them and adding the targeted pages with the appropriate level of trust. The technique called Redirection Spam consists in hiding redirections to pages with a different content by means of the use of scripting languages. This way, the normal processing of a crawler will not detect the redirection and it will obtain only the static content and the links, without noticing that the user will never see this content. Studies like the ones by Wu and Davison [38] and Chellapilla and Maykov [12] contribute to the formal analysis about the use of the aforementioned technique.

Among the works that mention the problem of Soft-404 pages is the popular book "*Modern Information Retrieval*" of Baeza-Yates and Ribeiro-Neto [3], where the authors discuss the problems of misusing the HTTP protocol. This misuse leads, among other things, to the appearance of Soft-404 pages. However, the authors do not study this problem or present any solutions.

Other study that discusses briefly the problem of the Soft-404 pages is the presented by Bar-Yossef et al. [6], where the authors develop an algorithm to detect different URLs with similar text. According its study the 7.5% of pages with similar text are Soft-404 pages. However, the method proposed by them is not focused in detecting Soft-404, although a collateral effect is that can detect some Soft-404 pages which contain similar text.

To the best of our knowledge, there are only three studies that propose some kind of solution for detecting Soft-404 pages.

Meneses et al. present in [28] a method to detect Soft-404 pages based on Naïve- Bayes classifier. However, this work contains, in our opinion, several deficiencies. The method is only based on two simple features, and the authors do not explain the reasons because these features work. Moreover, the dataset of Soft-404 pages (5,017 pages), the test dataset (1,000 web pages) and the training dataset (100 pages, 50 Soft-404 and 50 legitimate pages) are very small to trust in the results shown. Finally, the study does not contain a comparison between their method and the studies presented in the state of the art. Due this reasons, we have decided not include this method in our experiments.

Bar-Yossef et al. in [5], carried out a detailed analysis of this type of web pages, and propose an algorithm to detect them. This algorithm always sends two requests, one to the desired page and other to a random page of the same domain, to know the response of the web server when an unknown resource is requested. This random page will not exist in the domain, and so, the web server will send a 404 code or a Soft-404 page. With the response of both requests they compare the type and number of redirects, and the content of both responses, and decide if it is a normal page or a Soft-404 page.

The other study was conducted by Lee et al. [27], which discusses some of the limitations of the system presented in the article by Bar-Yossef et al. in [5], and proposes three heuristics to try to detect Soft-404 pages. This study tries to detect Soft-404 pages from a different point of view. The authors analyse the log of a crawling system to define three heuristics to carry out the detection. In summary, the heuristics are: a) the number of redirections of each host, b) the relation between the number of pages on a host and the number of redirections from this host, and c) the number of distinct target hosts reached by redirections from this host.

The study presented by Bar-Yossef et al. in [5] and the one by Lee et al. [27] contain several limitations and disadvantages, which will be discussed in sections 6.3 and 6.5. In this work, we propose a new detection method for Soft-404 pages, which is based on a set of content-based heuristics, that improves the systems proposed by Bar-Yossef et al. in [5] and by Lee et al. in [27].

Our system, on the one hand, improves efficiency because it is able to detect parking pages and various types of Soft-404 pages, such as root web pages, that would not be detected with the comparisons (content and redirects) proposed by Bar-Yossef et al. On the other hand, the proposed system reduces to half the HTTP requests performed using the method proposed by Bar-Yossef et al. At last, the computacional complexity of our algorithm is less than the algorithm of Bar-Yossef et al. To demonstrate this, we will discuss the efficiency and performance of our system and the comparison of its results with the obtained by the systems proposed by Bar-Yossef et al. in [5] and Lee et al. [27] (sections 6.3 and 6.5).

3. How many Soft-404 Pages?

Before analyzing the proposed heuristics, we will demonstrate the importance of Soft-404 pages on the Web

and the validity of the proposed system. To the best of our knowledge, there is no public dataset of Soft-404 pages. Therefore, we designed a system that generates requests on random pages, which will not exist in their domains, expecting the server to respond with a 404 HTTP code. For the cases where the web server returned a 200 HTTP code, we have a Soft-404 page. For this process we started with a set of 680,000 .es domains, obtained from "*Red.es*"[2]. After it, we check if the DNS responds to the domain and if the corresponding IP has a web server. Then, we generate a request of a random web page with 30 alphanumeric characters, so we ensure that the probability of the same web page existing, in this web site, is extremely small. Since the web servers may have different behavior in generating Soft-404 pages for different directories, we have selected a directory of each domain and we have appended the corresponding random string to it. The results are shown in Figure 1.

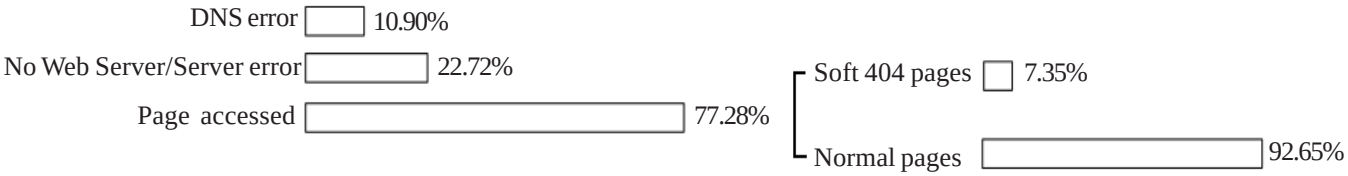


Figure 1. Number of Soft-404 pages and errors in DNS or in the web server

Analyzing the results, we observe that 89.10% have responded to the DNS requests and from that set, 22.72% had not a web server or returned some kind of error. From the set of pages that could access (77.28%), 7.35% responded with a Soft-404 when we made a request for an unknown page. This percentage of Soft-404 pages, 7.35%, is very similar to the 7.5% obtained by Bar-Yossef et al. in [6], which verifies the approach used and the results obtained. This demonstrates that Soft-404 pages constitute a significant problem on the Web and so, techniques for detecting them must be proposed.

To test our system, we developed two datasets. The first one, named Dataset #1, uses 52,000 Soft-404 pages extracted from the .es domain. In order to complete the dataset with no Soft-404 pages, we have selected random pages, of different subdomains of the .es domain. The second dataset was generated based on the 52,000 Soft-404 pages and a global set of pages obtained from the "*Stanford WebBase Project*"[3], which is part of "*The Stanford Digital Libraries Technologies Project*"[4]. We want to remark that we have not only used root pages of web sites, because this will obviously lead to biased results.

From these two datasets, we built four subsets, one that follows the distribution of Soft-404 links discussed in [5], i.e., 25% of the total, and another subset, which assumes 50% of Soft-404 pages. In short, these are the datasets used:

- **Dataset #1-A:** .es pages with 25% of Soft-404 pages (50,000 normal pages and 15,625 Soft-404).

- **Dataset #1-B:** .es pages with 50.00% of Soft-404 pages (50,000 normal pages and 50,000 Soft-404).

- **Dataset #2-A:** Global pages with 25% of Soft-404 pages (50,000 normal pages and 15,625 Soft-404).

- **Dataset #2-B:** Global pages with 50.00% of Soft-404 pages (50,000 normal pages and 50,000 Soft-404).

With the B datasets, we want to demonstrate that the system does not depend on the number of the Soft-404 pages, i.e., on the probability of a Soft-404 page can appear. The datasets #1 and #2 demonstrate that the heuristics do not depend on the language or the typical features of each country and, therefore, can be used on the Global Web.

Finally, we want to do two final notes about the datasets:

- We are aware that in studies applying machine learning techniques it is important that the dataset used does not contain near-duplicates. To avoid this, we have checked manually a 10% of each of the created datasets, and we have detected that only the 2% of the web pages are near-duplicates. Extrapolating this result, only 2% of the datasets are near-duplicates. In our opinion, this is a good result and it indicates that the results are reliable.

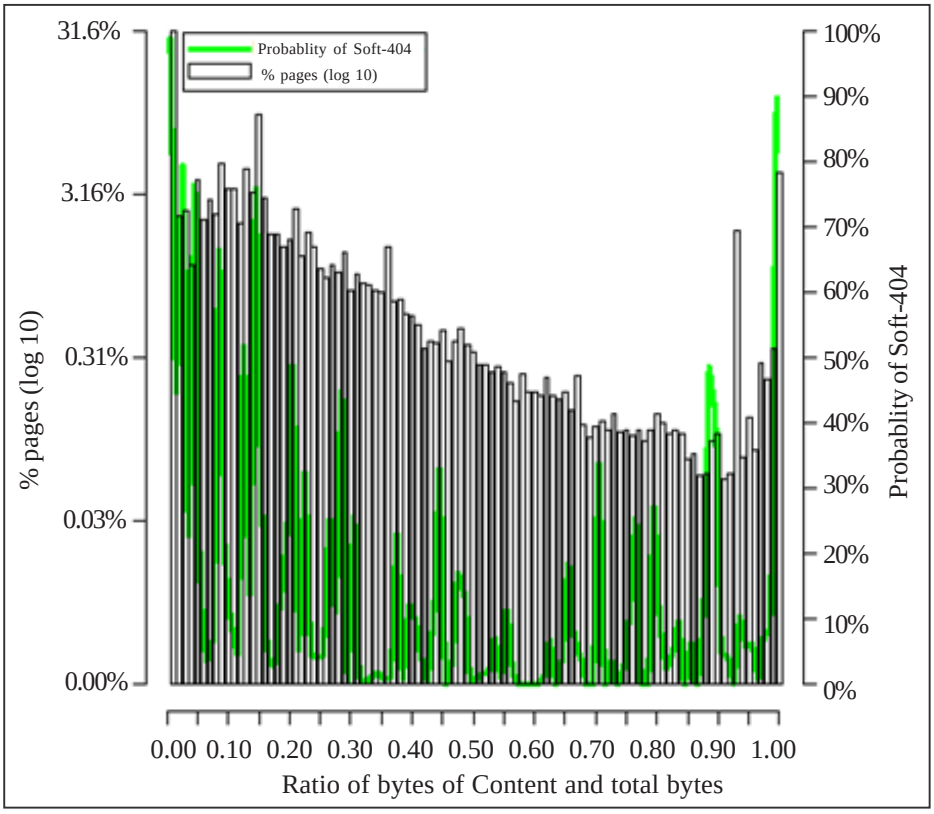
Moreover, to do that the probability of near-duplicates in the training set be less, and therefore, it will not provoke an unreasonably accuracy, we have used several sizes of training set, 50%, 25%, 15%, 10%, 5% (see Section 6). In a training set of 5% of the web pages is very unlikely that exists several near- duplicates documents.

- Other important issue is that due the usage of Soft-404 pages of the .es domain the results could be biased. To avoid these, we have mixed this dataset with webpages of a global set of pages obtained from the "*Stanford WebBase Project*". The studies about Spanish Web (.es) [4] [35] do not indicate that the Spanish Web contains more or less Spam or Soft-404 pages than the .com domain or the Global Web. Moreover, in the study presented by Ntoulas et al. [29], the authors demonstrated

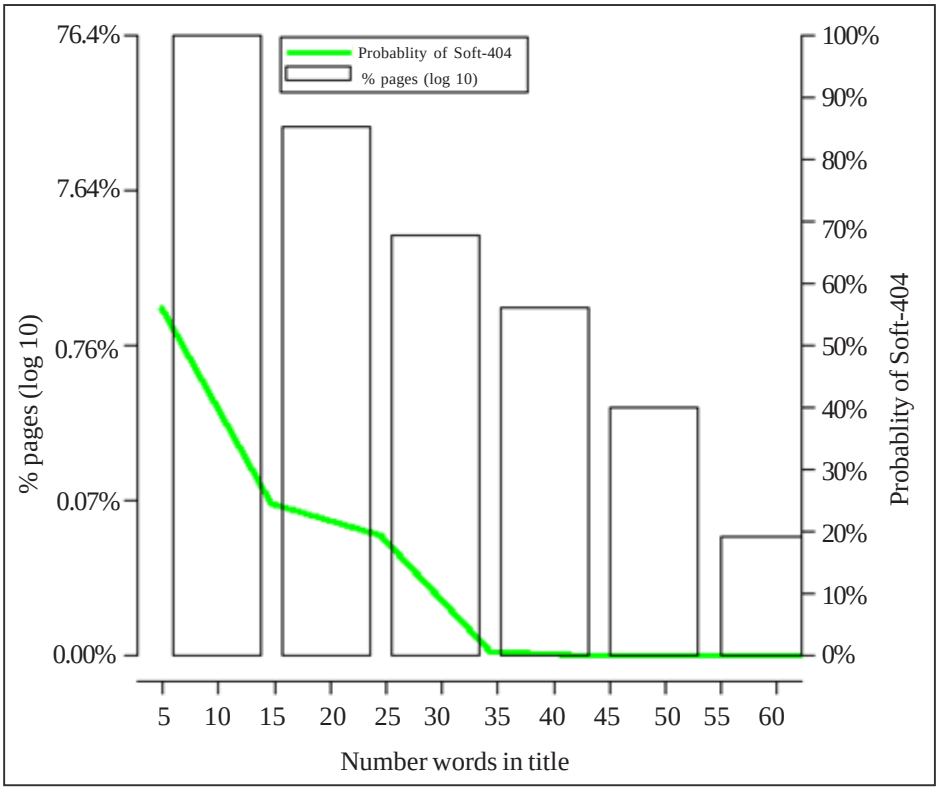
[2] http://www.red.es

[3] http://dbpubs.stanford.edu:8091/~testbed/doc2/WebBase/

[4] http://diglib.stanford.edu:8091/



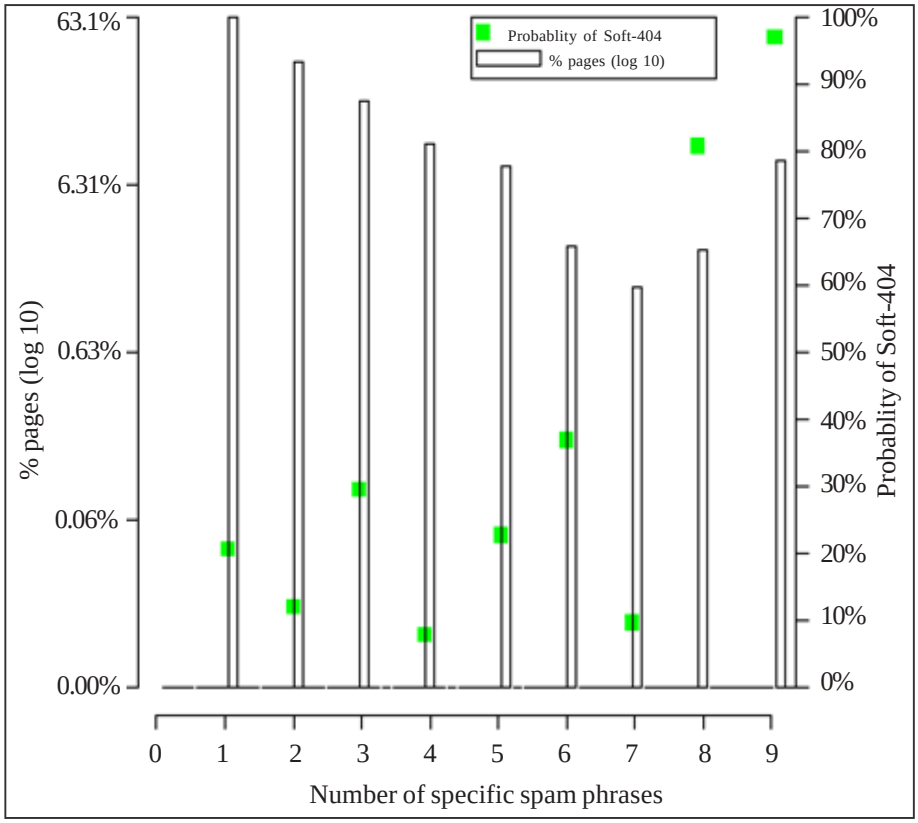
(a)



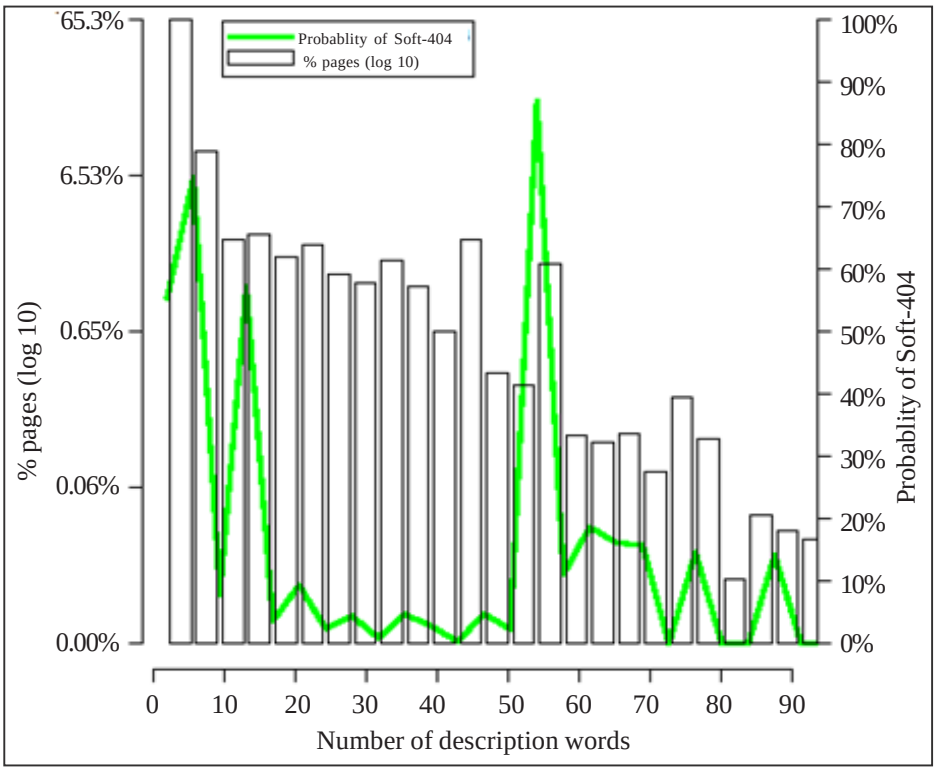
(b)

Figure 2. Probability of Soft-404 relative to ratio of bytes of content and total bytes (a), and probability of Soft-404 relative to number of words in title

that the .com domain contains less Spam than .us domain and similar than .de and .net domains. This study also indicates that the web pages written in French or German (languages of the EU) contains more Spam than pages written in English, language usually used in the .com domain.



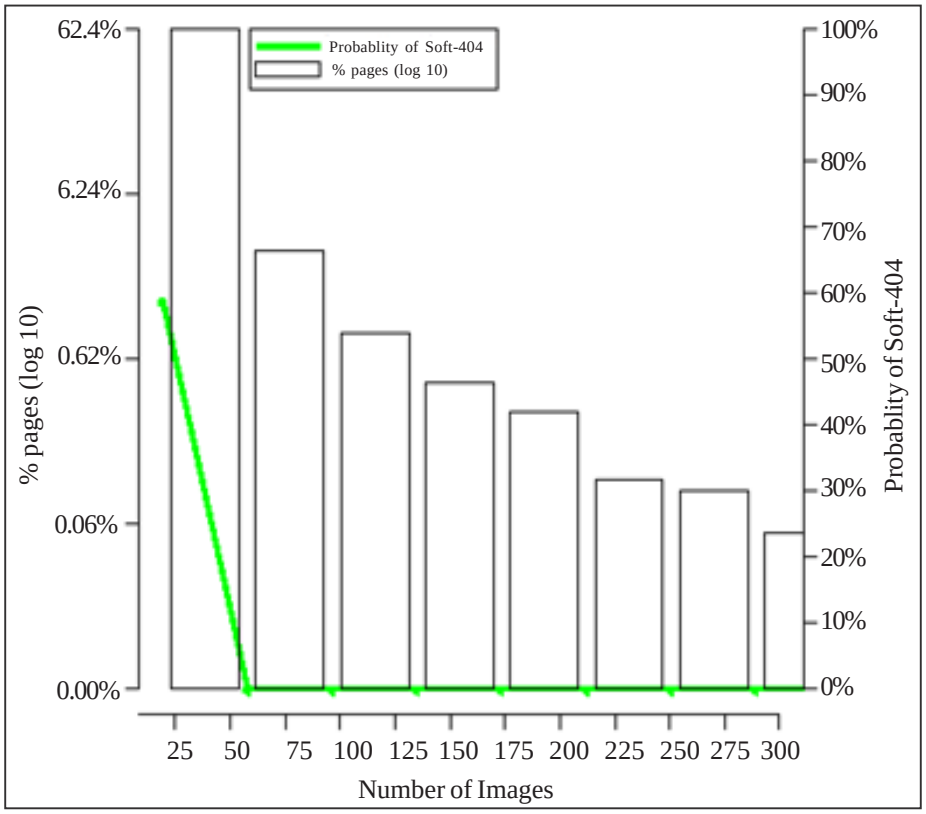
(a)



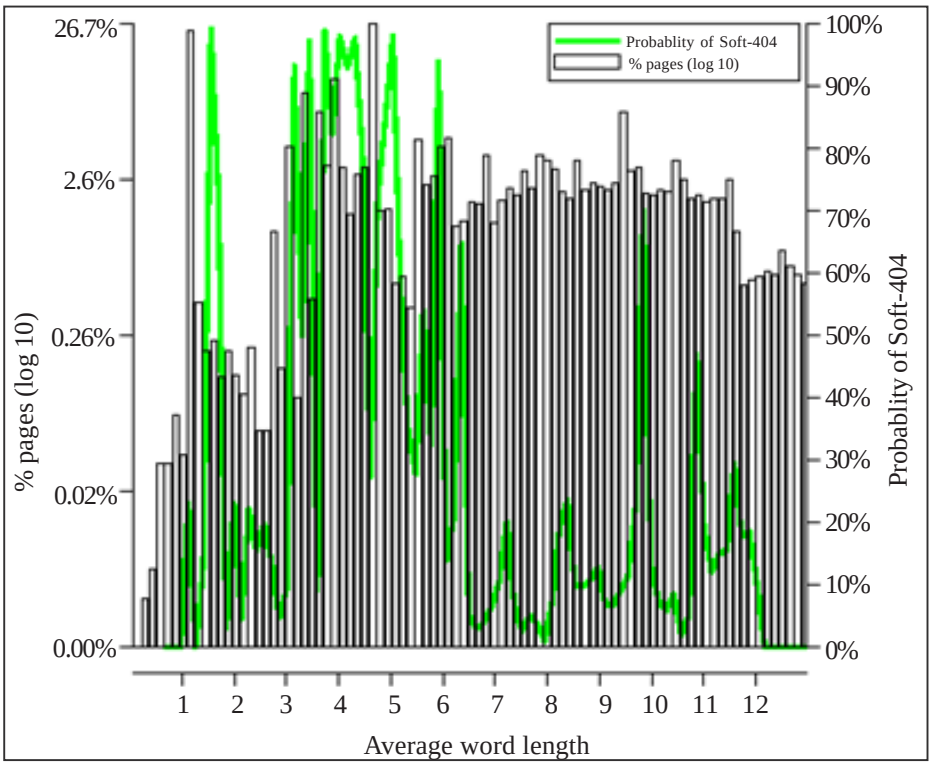
(b)

Figure 3. Probability of Soft-404 relative to number of specific Soft-404 phrases (a), and probability of Soft-404 relative to number of description words (b)

- Finally, the study presented by Bar-Yossef et al. [6] indicates that, in its dataset (mainly .com domains), the 7.5% of the web pages were Soft-404, a value very similar to the detected in the .es domain. For these reasons, we think that it is demonstrated that the proposed method can be used in a dataset containing web pages of the Global Web.



(a)



(b)

Figure 4. Probability of Soft-404 relative to number of images (a), and probability of Soft-404 relative to average length words (b)

4. Heuristics to Detect Soft-404 Pages

After having explained the importance of the Soft-404 pages in the Web, we will discuss a set of heuristics that aim to characterize and detect this type of pages. To justify their efficacy, we tested each heuristic on the dataset #1-A. In each figure, we show values on horizontal axis and on left and rigth vertical axis. The horizontal axis depicts

a set of value ranges for each heuristic. The left scale of the vertical axis applies to the bar graph, and depicts the percentage of pages in our dataset that fell into a particular range. The right scale of the vertical axis applies to the line graph, and depicts the percentage of sampled pages in each range that were judged to be Soft-404 pages. Finally, to improve the presentation of the data, we used a log10 progression on the left scale of the vertical axis that indicates this ratio, but showing the percentage instead of the absolute value.

- **Ratio of content bytes and total bytes:** This heuristic focuses on the fact that Soft-404 pages have little useful content. To demonstrate this, we propose to analyze the relationship between bytes of useful content and the total bytes of the page. We can see in Figure 2a that with ratios lower than 0.30 the probability grows progressively reaching points with 60%, 80% and 100%.

- **Number of words in the title:** Analyzing the set of Soft-404 pages, we observed that many pages have no title or its title is shorter than that in normal pages. In Figure 2b we observe that in pages with less than 10 words in the title, the probability of being a Soft-404 page is 40%, and with values less than 5 the probability of being Soft-404 page rises to 60%.

- **Specific phrases/words:** We have observed that it is usual for Soft-404 pages to contain common terms and phrases. After analysing our dataset, we created a list of 30 terms, which contains words like "*urgent*", "*removed*", "*error*" and "404", among others. This subset was removed from the dataset for not biasing the experimental results. In Figure 3a we observe the results obtained by means of the list described above. We can see that, from 5 or more typical Soft-404 words on, the probability of Soft-404 pages is progressively increasing between 30% and 100%.

- **Number of "*description words*":** As shown in the previous heuristic, Soft-404 pages are pages with little content and poor quality. So, we analyze the number of words in the attribute description of the HTML "meta" tag. Soft-404 pages tend to contain fewer words in that attribute than normal pages. Analyzing the results shown in Figure 3b, we can say that this heuristic works correctly. With values less than 10 the probability of being Spam rises to 75%, and for higher values the probability decreases, in general, to values between 20% and 10%.

- **Number of images:** Starting from the fact that the content of many Soft-404 pages is dynamically generated or not thoroughly detailed, we observed that Soft-404 pages have no images or the number of images is smaller than in normal pages. These results are presented in Figure 4a, showing that in pages with less than 50 images the probability of being Soft-404 page rises to 60%. Although the results for Soft-404 pages are not as conclusive, it is useful to determine whether the page is a normal page, since with values higher than 50 the probability of being Soft-404 is 0%.

- **Word length:** We observed that Soft-404 pages contain words that are shorter than those in normal pages. This is because they do not contain elaborate or complex sentences. In Figure 4b, for values between 1.5 and 3, and between 3 and 6.5, the probability of being Soft-404 rises to 90% or 100%, respectively. For values bigger than 6.5, the probability of being Spam is around 20%.

- **Number of "*keywords*":** Following with the idea of the "*description words*" heuristic, we also studied the contents of the attribute "*keyword*" in the HTML "*meta*" tag of Soft-404 pages. The results in Figure 5a show that with less than 10 "*keywords*" per page, the probability of being Soft-404 rises to 60%. For values between 30 and 45, and bigger than 55, the probability of being Soft- 404 peaks 90%, perhaps due to parking pages trying to do "*keyword stuffing*" to get more traffic, and to sell the domain or make a profit through advertising.

- **Size of the web page:** We observed that many Soft-404 pages tend to be smaller in size than the average normal pages. Figure 5b points out that with values less than 50,000 bytes the probability of being Soft-404 rises to 60%.

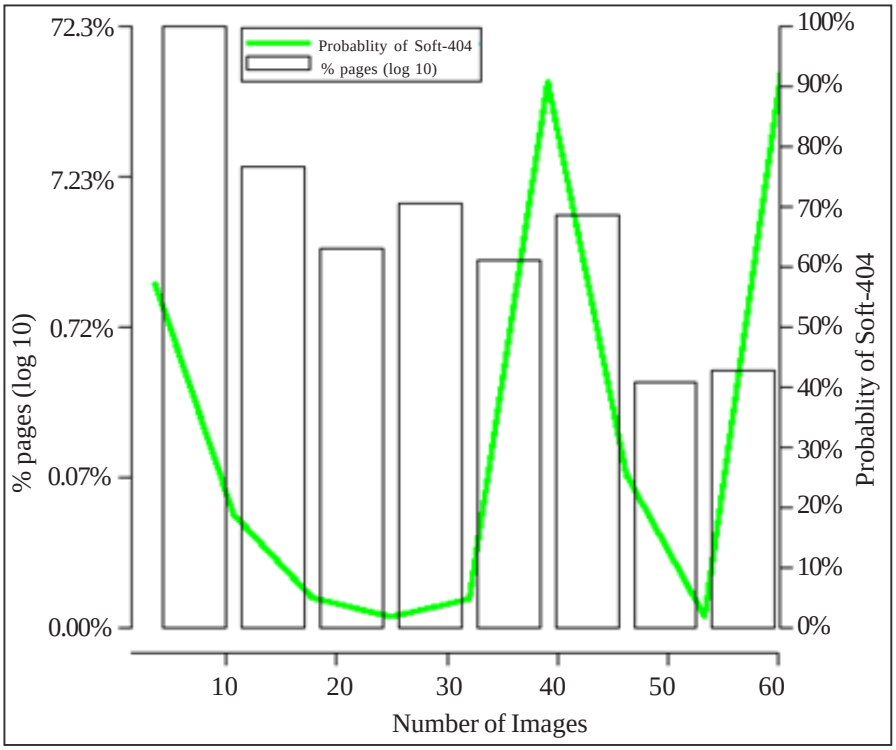
5. Method for Soft-404 Page Detection

In this section we describe the proposed method for Soft-404 page detection. It is based on the analysis of the web page content, where we apply the different heuristics described in the previous Section. So, we characterize a page, and then we decide if it is Soft-404 or not.

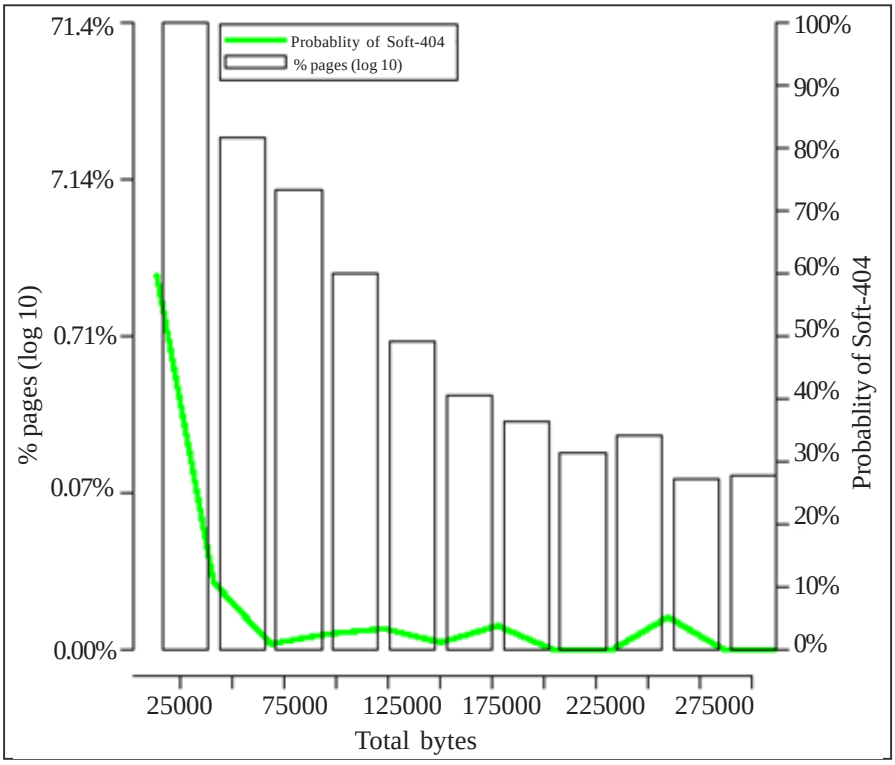
For the appropriate combination of the heuristics, we tried different classification techniques (decision trees, techniques based on rules, neuronal networks, etc.). We conclude that the best results are achieved when we use decision trees. More specifically, we chose the C4.5 algorithm [32] [31] [33].

To improve the results we assessed two techniques, "*bagging*" [32] and "*boosting*" [32]. These techniques create a set of N classifiers, combine those that obtained the best results and build a composite classifier. In general, "*bagging*" creates N subsets of n random elements with replacement. Thus, N classifiers are obtained. Each web page that wants to be classified has to be evaluated by each one of the N classifiers. The class in which the web page will be added (Soft- 404 or normal) depends on the votes of most of the N classifiers. The "*boosting*" technique works in a similar way. Each item has an associated weight. This weight reflects the probability of occurrence in the set. N classifiers are generated in N iterations, but for each misclassified item its weight is increased. Once again, the final decision of marking it as Soft-404 or not will be brought by a majority of results of the N classifiers.

Figure 6 shows a portion of the decision tree generated to classify a page as Soft-404. Each node uses one of the heuristics presented in Section 4 and, according to different



(a)



(b)

Figure 5. Probability of Soft-404 relative to number of keywords (a), and probability of Soft-404 relative to total size of a web page (in bytes) (b)

thresholds, decides or delegates to another node to refine the classification.

6. Experimental Results

In this Section, we discuss all the issues we found for both the execution and the assessment stages, and we show and analyze the results obtained with each dataset. We also compare the effectiveness of our system and the systems proposed by Bar-Yossef et al. [5] and Lee et al. [27]. Finally, we analyse the performance of the proposed heuristics, and compare the performance of our

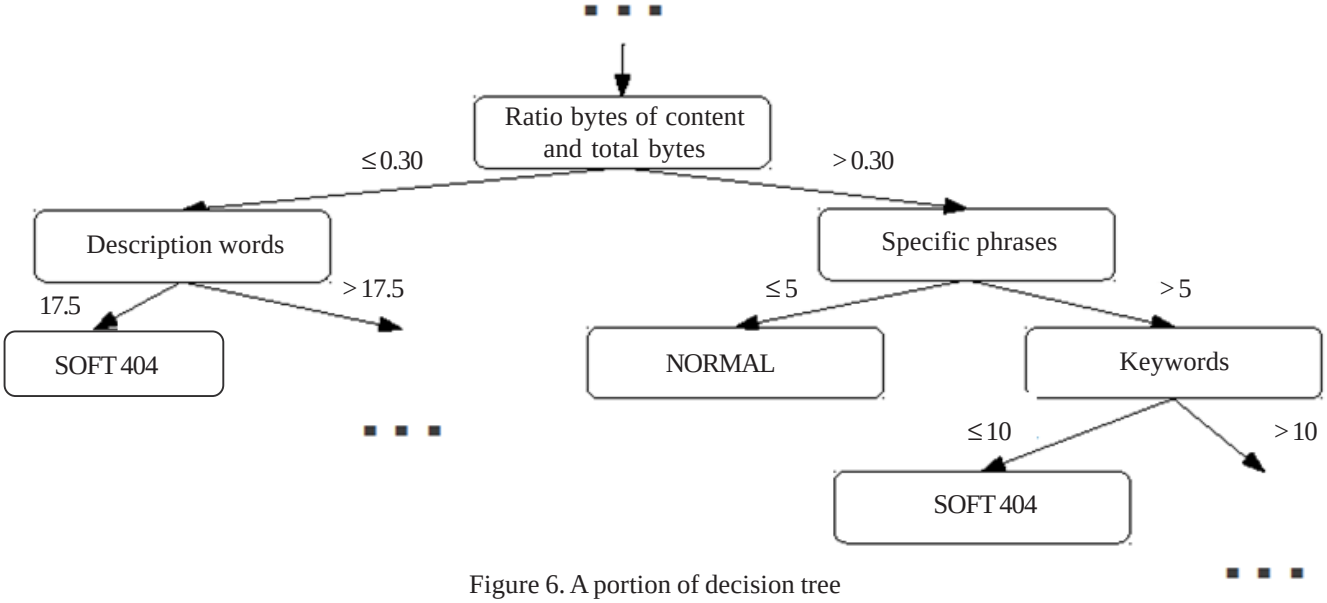


Figure 6. A portion of decision tree

system and the system proposed by Bar-Yossef et al. in [5].

6.1 Experimental Setup

We developed a tool to run and store the results of each of the proposed heuristics for each web page of the datasets presented in Section 3, as well as the processing time for each of the heuristics.

In order to choose the most appropriate algorithm for our heuristics, we used WEKA [21], a tool for automatic learning and data mining, which includes different types of classifiers and different algorithms for each classifier. Once the heuristics were chosen, our experiment was executed and we obtained different results for several classification algorithms. After analysing the aforementioned results we chose the C4.5 classification algorithm as the best one.

For the evaluation of the classifier we used the cross validation technique [25], that consists in building k data subsets. In each iteration a new model is built and assessed, using one of the sets as test set and the rest as training set. We used 10 as the value for k (ten-fold cross validation).

On the other hand, we have performed several experiments using the datasets #1 and #2, but using different training sets, to demonstrate that the obtained results do not depend on the number of web pages used in the training set. We have carried out experiments using 50%, 25%, 15%, 10%, and 5% of the dataset as training set.

In summary, to obtain the results on the datasets, we have combined: a) different sizes of datasets, b) web pages on the global and Spanish Web, c) web pages of different countries, d) different techniques to obtain results, and e) different sizes of the training set.

	Soft-404		Normal page	
	Precision	Recall	Precision	Recall
Dataset #1-A				
C4.5	0.806	0.947	0.983	0.932
C4.5 - Bagging	0.813	0.937	0.981	0.936
C4.5 - Boosting	**0.807**	**0.947**	**0.983**	**0.937**
Dataset #1-B				
C4.5	0.922	0.980	0.979	0.918
C4.5 - Bagging	0.927	0.980	0.980	0.924
C4.5 - Boosting	**0.928**	**0.983**	**0.983**	**0.952**

Table 1. Results for dataset #1

	Soft-404		Normal page	
	Precision	Recall	Precision	Recall
Dataset #1-A				
C4.5	0.980	0.960	0.979	0.918
C4.5 - Bagging	0.986	0.963	0.996	0.999
C4.5 - Boosting	**0.994**	**0.973**	**0.997**	**0.997**
Dataset #1-B				
C4.5	0.992	0.987	0.987	0.992
C4.5 - Bagging	**0.996**	**0.987**	**0.987**	**0.996**
C4.5 - Boosting	0.992	0.98	0.979	0.998

Table 2. Results for dataset #2

6.2 Effectiveness of the System

In this Section we show and discuss the results obtained in each dataset presented in Section 3. Tables 1 and 2 show the results on dataset #1 and #2, respectively. We also compare the results obtained by the system using several amounts of web pages to train the classifier.

			Soft-404		Normal page	
Training %	Dataset		Precision	Recall	Precision	Recall
50%	#1-A	C4.5 - Bagging	0.821	0.913	0.973	0.941
		C4.5 - Boosting	0.815	0.922	0.973	0.937
	#1-B	C4.5 - Bagging	0.926	0.978	0.977	0.924
		C4.5 - Boosting	0.924	0.980	0.980	0.921
	#2-A	C4.5 - Bagging	0.978	0.958	0.996	0.998
		C4.5 - Boosting	0.989	0.970	0.997	0.999
	#2-B	C4.5 - Bagging	0.994	0.984	0.984	0.994
		C4.5 - Boosting	0.995	0.988	0.988	0.995

Table 3. Results of the system using 50% of each dataset to train the classifier

			Soft-404		Normal page	
Training %	Dataset		Precision	Recall	Precision	Recall
50%	#1-A	C4.5 - Bagging	0.825	0.900	0.969	0.941
		C4.5 - Boosting	0.805	0.937	0.980	0.933
	#1-B	C4.5 - Bagging	0.926	0.971	0.970	0.924
		C4.5 - Boosting	0.924	0.974	0.973	0.921
	#2-A	C4.5 - Bagging	0.971	0.950	0.995	0.997
		C4.5 - Boosting	0.990	0.948	0.995	0.999
	#2-B	C4.5 - Bagging	0.993	0.980	0.980	0.993
		C4.5 - Boosting	0.993	0.985	0.985	0.993

Table 4. Results of the system using 25% of each dataset to train the classifier

Looking at the results of dataset #1-A (Table 1), we can see that the best recall, 0.947, is achieved by applying the C4.5 algorithm and Boosting. However, the best precision, 0.813, is achieved by applying Bagging, although this value is smaller than other values we will show in the next datasets. In the case of dataset #1-B (Table 1) the results show an improvement, since the classifier has been trained with more Soft-404 pages than in the previous case. In this dataset we obtained a precision of 0.928 and 0.983 on Soft-404 pages and normal pages, respectively, and a recall of 0.983 and 0.952 on Soft-404 pages and normal pages. These results are obtained by means of C4.5 algorithm and Boosting. We can see the results are very good both in dataset A as in B. This shows that the heuristics are correct for detecting Soft-404 pages.

Table 2 shows the results for dataset #2. In the dataset #2-A, the recall rises to 0.973, a value higher than that obtained in the dataset #1-A. Once again, the results of both datasets (#2-A and #2-b) are very similar, which reaffirms the idea that the heuristics are correct and do not depend on the number of Soft-404 pages in the set.

The precision obtained in dataset 1 is a bit lower, because the normal pages (not Soft-404) contain a number of home pages higher than in dataset 2. The home pages are more difficult to characterize because usually they are more similar among them, that is, they share similar results of the proposed heuristics, which does not occurs in the common web pages. Therefore, the system needs to train with more web pages. For that, the dataset 2 has obtained better precision.

In short, we can see that the best results are obtained in the dataset #2, achieved by applying C4.5 and Boosting.

Analyzing the results of Tables 1 and 2 we can see that there is a small set of Soft-404 pages that is not detected. We have analysed these web pages, and we have concluded that this fact occurs because a web server does not send a typical Soft-404 page, but sends the content of the root page of the web site. However, this would not be a problem for crawlers, since its similarity detection module will alert of a similar content between this type of Soft-404 pages and the corresponding root page of the web site. After that, the crawler will remove this repeated content.

In both datasets we see that the results do not improve by increasing the amount of web pages of the dataset. Therefore, the proposed heuristics characterize correctly

			Soft-404		Normal page	
Training %	Dataset		Precision	Recall	Precision	Recall
50%	#1-A	C4.5 - Bagging	0.817	0.896	0.968	0.940
		C4.5 - Boosting	0.813	0.897	0.968	0.939
	#1-B	C4.5 - Bagging	0.925	0.965	0.964	0.924
		C4.5 - Boosting	0.921	0.967	0.965	0.919
	#2-A	C4.5 - Bagging	0.965	0.954	0.995	0.997
		C4.5 - Boosting	0.980	0.955	0.995	0.998
	#2-B	C4.5 - Bagging	0.989	0.970	0.970	0.989
		C4.5 - Boosting	0.991	0.980	0.980	0.993

Table 5. Results of the system using 15% of each dataset to train the classifier

			Soft-404		Normal page	
Training %	Dataset		Precision	Recall	Precision	Recall
50%	#1-A	C4.5 - Bagging	0.815	0.886	0.965	0.940
		C4.5 - Boosting	0.813	0.873	0.961	0.940
	#1-B	C4.5 - Bagging	0.926	0.959	0.958	0.925
		C4.5 - Boosting	0.919	0.962	0.961	0.917
	#2-A	C4.5 - Bagging	0.954	0.921	0.992	0.996
		C4.5 - Boosting	0.969	0.935	0.993	0.997
	#2-B	C4.5 - Bagging	0.989	0.970	0.970	0.989
		C4.5 - Boosting	0.991	0.978	0.978	0.991

Table 6. Results of the system using 10% of each dataset to train the classifier

			Soft-404		Normal page	
Training %	Dataset		Precision	Recall	Precision	Recall
50%	#1-A	C4.5 - Bagging	0.830	0.836	0.951	0.949
		C4.5 - Boosting	0.805	0.902	0.970	0.935
	#1-B	C4.5 - Bagging	0.918	0.948	0.947	0.917
		C4.5 - Boosting	0.912	0.956	0.954	0.909
	#2-A	C4.5 - Bagging	0.878	0.896	0.989	0.987
		C4.5 - Boosting	0.927	0.895	0.989	0.984
	#2-B	C4.5 - Bagging	0.985	0.966	0.966	0.985
		C4.5 - Boosting	0.988	0.971	0.971	0.988

Table 7. Results of the system using 5% of each dataset to train the classifier

Soft-404 pages without depending of the number of web pages in the training set.

In order to test the correct characterization of the proposed heuristics and to not depend on the number of web pages in the training set, we performed several experiments using the datasets described above (#1 and #2), but using different training sets sizes. Concretely, we conducted experiments using 50% of the dataset as training set, and subsequently we have reduced the training set to 25%, 15%, 10%, and 5%. The results are shown in tables 3, 4, 5, 6 and 7, respectively.

Analyzing the results, we can say that they are very similar to the results discussed above (Table 1 and 2). To compare the results we will show the results obtained using C4.5, Bagging and Boosting. The system using 50% of each dataset to train the classifier obtains a precision

of 0.989 and a recall of 0.970. In the case where the system uses a 5% of each dataset to train the classifier, the system achieves a precision of 0.927 and a recall of 0.895. The difference is not significant, concretely, the precision decreases a 0.062 and the recall a 0.075 We can see that the system maintains very good results independently of the amount of pages used to train the classifier.

	Soft-404		Normal page	
	Precision	Recall	Precision	Recall
Dataset #2-A				
Bar-Yossef *et al.*	0.787	0.945	0.914	0.768
Soft404Detector	**0.994**	**0.973**	**0.997**	**0.997**
Dataset #2-B				
Bar-Yossef *et al.*	0.797	0.935	0.920	0.757
Soft404Detector	**0.992**	**0.980**	**0.979**	**0.998**

Table 8. Results of Bar-Yossef et al. and Soft404 Detector on dataset #2

6.3 Comparison with the Effectiveness of other Systems

In this section we compare the effectiveness of our system with the systems proposed by Bar-Yossef et al. [5] and Lee et al. [27], respectively.

The comparison between our system and the system proposed by Bar-Yossef et al. [5], has been carried out from a theoretical and empirical point of view. The experiments have been performed on the dataset #2, because it contains web pages of the global web and therefore it is more similar to the dataset used by Bar-Yossef et al. [5].

For the theoretical comparison, we define an upper bound of the types of Soft- 404 pages that each system can identify. Bar-Yossef et al. state in their article that they do not detect many cases of Soft-404 pages: a) a root pages of the web server can never be a Soft-404 page, and b) parking pages. We randomly selected a subset of 20% of the set of Soft-404 pages, and we identified those pages that are parking pages. We found that 47.2% of the Soft-404 pages are parking pages. Therefore, in the best case, the system proposed by Bar-Yossef et al., will be able to detect only 52.8% of the Soft-404 pages. Instead, our system detects more than 98% of Spft-404 pages in all the datasets analyzed.

For the empirical comparison, we have developed the algorithm explained by Bar-Yossef et al. This developing has been made following exactly the pseudocode included in the article [5]. Thus, the obtained results should be the same ones. Table 8 shows the results obtained by their algorithm and the results obtained by our system.

Focusing on results obtained for Soft-404 pages, we can see that in the dataset #2-A, the system of Bar-Yossef et al. has achieved a precision and a recall of 0.787 and 0.945, while our system has achieved 0.994 and 0.973, respectively. On the dataset #2-B, our system has obtained a precision of 0.992 and a recall of 0.980, and the system proposed by Bar-Yossef et al. has obtained a precision of 0.797 and a recall of 0.935. In short, our system improves the recall obtained by Bar-Yossef et al. to detect Soft-404 approximately in a 5% and the precision approximately a 25%. In the case of the Normal pages (no-soft-404 pages), our system improves the precision and the recall approximately in a 6% and in a 32%, respectively.

Remark that the results obtained by Bar-Yossef et al. are against web server that responds with 404 HTTP code when receives requests to unknown documents, its precision and recall would be 1. This is because the algorithm proposed by the Bar-Yossef et al. sends a request to a random web page in order to detect whether the corresponding web server sends Hard-404. Whether the web server sends a Hard-404, this web page is considered as no-Soft-404, that is, a normal web page.

In our opinion, it is unnecessary and inefficient because: a) as we will discuss in sections 6.4 and 6.5, to make HTTP requests to detect Soft-404 pages is inefficient and causes a great loss of resources and time and b) the precision and recall obtained would be 1, and our system achieves results very close to 1 (precision 0.994 and recall 0.997), without having to make HTTP requests.

Analyzing the algorithm proposed by Bar-Yossef et al. and obtained results, we concluded that their system misclassified many web pages. The weaknesses of their algorithm have been identified in several points:

- Their system does not work in web servers that send Soft-404 pages without making any redirections. Their algorithm does not receive a 404 HTTP code, nor any redirection, since the web server has sent a 200 HTTP code, but with other content.

- There are a lot of cases where the number of redirections and/or the set of URLs are different between the original web page and the random web page requested to the web server. Concretely, the 6.5% of the Soft-404 pages were misclassified due to this fact.

- Finally, another important weakness is that the detection is based on the comparison of the web content. To compare the content of the web pages is computationally expensive and unreliable. The authors consider a page is Soft-404 whether the original page and the random page

have "*almost-identical content*". The problem arises in interpreting the numerical value of the word "*almost*". We have considered that two web pages are "*almost-identical*", when the 80% of their contents are equals. This weakness does the 93.5% of the Soft- 404 pages were misclassified.

For comparison of the content, we were strict, because in other case (allowing that the system can fail in the comparison process of the contents) the system of Bar-Yossef et al. would fail in a important part of their algorithm and therefore, it would not detect a lot of Soft-404 pages. The first step to comparison process is to extract the useful content of each page (remove HTML tags, scripts, images, etc.). The process of extraction of the useful content of the web pages is very complicated. For this, we have followed the approach developed by Donghua et al. in [14]. This study is based on that the location of the main content is very centralized and has a good hierarchical structure. The authors found that the threshold values of the node containing content are obviously different from that of other nodes in the same level. With these values, they have proposed an algorithm that judges the content by several parameters in the nodes (Link Text Density, Link Amount, Link Amount Density and Node Text Length).

	Soft-404	
	Precision	Recall
Dataset #2-A		
Lee	X	0.695
Soft404Detector	**0.994**	**0.973**

Table 9. Results of Lee et al. system and Soft404 Detector on dataset #2-A

To compare our results with the ones obtained by Lee et al. [27], we would have to develop their algorithm and test the two systems on our datasets or on their dataset. However, there are three problems for doing it: a) the paper of Lee et al. [27] does not contain a pseudocode of their proposed algorithm and neither enough information for doing this, therefore, the comparison between both algorithms would be incorrect, b) the system of Lee et al. follows a perspective completely different from ours (performs a complete crawling on the analysed host and on their related hosts), so we could not validate our system on their dataset, nor their system on our dataset, and c) the dataset analyzed in their article cannot be obtained because it is not public. Because of these problems, we have compared both systems by analyzing the disadvantages existing in the system of Lee et al., and comparing the results shown in his article with those obtained by our system. The disadvantages of the system of Lee et al. are the following:

- The detection is not done in real time, which means that all these pages will be processed and subsequently the system will detect them. This is because, to decide whether a page is Soft-404, the system must perform a previous crawling of the host of this page and the hosts reached by redirections from this host. It provokes a loss of resources of the crawler system and of the search engine, since will process, index and follow links of web pages, that subsequently, the system will detect them as Soft-404.

- It is based on 3 simple heuristics that can fail too many times, and it does not apply knowledge previously learned. For example, a page with a lot of redirects, or a host with many web pages and a little number of target URLs redirected from URLs in this host. As occurs in the system proposed by Bar-Yossef et al., the system needs HTTP redirects, otherwise it does not detect that a page is Soft-404.

Unlike the Lee et al. system, our system:

- Detects Soft-404 pages in real time and it is not necessary to carry out a crawling and an analysis of related hosts. So, It prevents loss of resources of the crawling system and search engine, preventing the processing, indexing of web pages that should not be indexed.

- It is based on previously learned knowledge and a set of tested heuristics. Also, it is not based on the type of redirects performed by the web pages, so our system will detect Soft-404 independently of the type of redirections.

Table 9 shows the results obtained by Lee et al. in their article [27]. To compare their results with our results, we use the dataset #2-A, since it contains web pages of the global Web, like contains the one used by Lee et al. The system of Lee achieves a recall of 0.695 detecting Soft-404 pages. Analysing the results obtained by Soft404Detector, we see that achieves a recall of 0.973 and a precision of 0.994. That is an improvement of 40% in the recall of Soft-404 page. The information about the precision is not provided in the article, but the authors analyse the precision of their system to detect a set of soft errors (5xx, 403, Spam, etc.). They achieve a precision of 0.866. Whether we compare the precision of our system, 0.994, with their results, 0.866, we achieve an improvement of 14.78%.

6.4 Performance of the Heuristics

To verify the computational cost of the proposed heuristics, we analyzed the execution time of each one of them. In addition, we compared the performance of our system with that of the one proposed by Bar-Yossef et al. in [5]. First, we performed a quantitative analysis of the performance of our system. To do this, we executed our heuristics, and the detection system, on the dataset #2.

Figure 7 shows a box and whisker plot with the results of the average execution time per page for each discussed heuristics in Section 4.

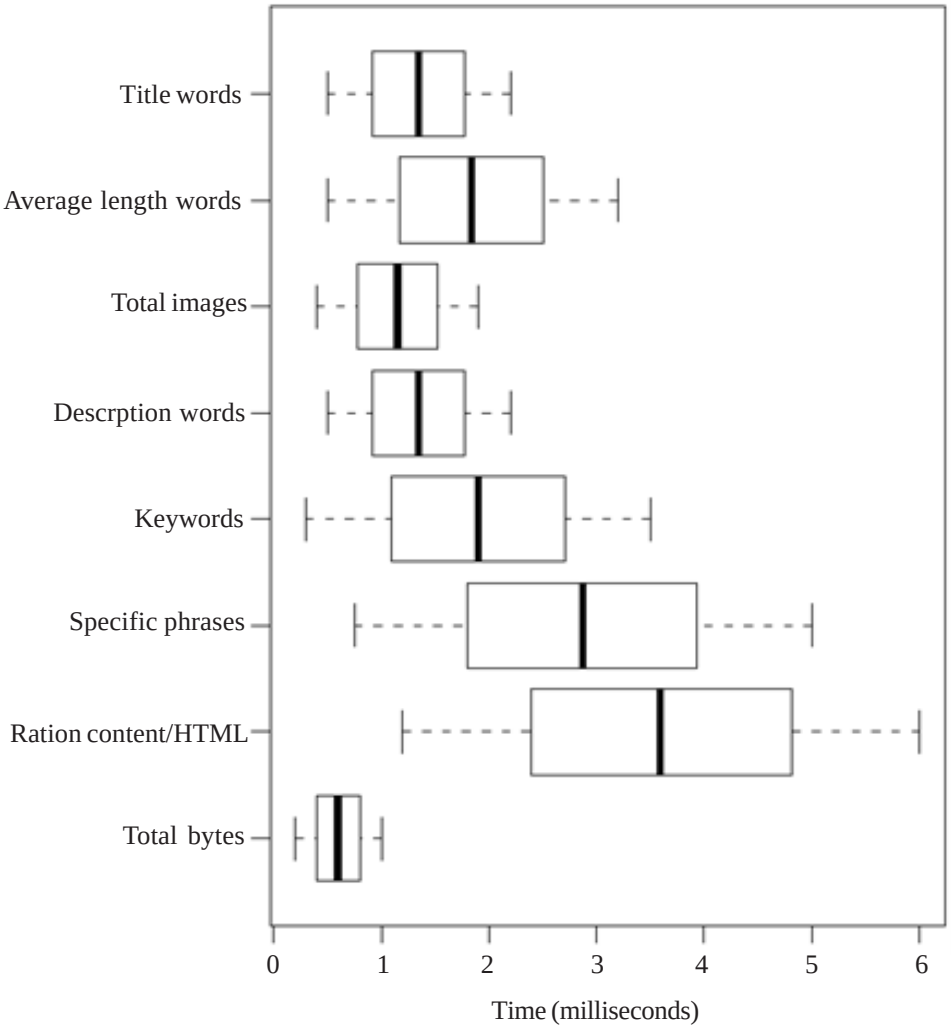


Figure 7. Execution times of the heuristics in dataset #2

Analyzing the results, we note that the execution time of the heuristics "*Title words*", "*Total images*" and "*Description words*" are very similar, since the logic of each heuristic and the executed code is very similar. Concretely, these heuristics have a maximum run time of approximately 2 milliseconds and a median next to 1 millisecond. The heuristic "*Average length words*" has obtained an execution time higher than the previous one, due to its computational complexity is greater. The "*Average length words*" has a maximum approximately of 3 milliseconds with a median of 2 milliseconds. In the case of "*Specific phrases*" and "*Ratio content/HTML*" the execution times obtained are higher.

This is due to that the computational cost, of searching for specific words in the content, and separating the useful content of the HTML code to get the relation between them, is higher than the other heuristics. On one hand, the heuristic "*Specific phrases*" is executed in the worst case in 5 milliseconds and has a median close to 3 milliseconds.

On the other hand, the system uses approximately 6 milliseconds in the execution of the heuristic "*Ratio content/HTML*" with a median of about 3 milliseconds. Finally, the fastest heuristic is "*Total bytes*", which achieve a maximum of 1 millisecond and a median of about 0.5 milliseconds.

The results show that there is a distance between the median and maximum. In our opinion this is due to: a) the execution time depends on the size of the analysed page and b) the complexity of the HTML code.

6.5 Comparison with the Performance of other Systems

In this section, we discuss the performance of our system when it is compared with other systems.

In this case our results were not compared with those obtained by Lee et al. [27], since their technique uses an approach completely different from ours. Concretely, the system proposed by Lee et al. [27], needs to make a complete crawling of a host and the hosts reached by redirections from this host, before it can decide whether or not a page is Soft-404. This makes the process before deciding whether a page is Soft-404 slower than our system or that the proposed by Bar-Yossef et al.

However, when the system of Lee et al. has carried out the crawling of a host, and its related hosts, it will be able to decide which pages are Soft-404 faster than our system, but with a great disadvantage with respect to it. The system proposed by Lee et al. causes that the crawling system, used by the search engine, has to process, index and extract the links from a web page that in the future can be a Soft-404 page. This fact will cause a great loss of resources and money.

To analyse the algorithm proposed by Bar-Yossef et al., we have separated it into its basic processes and obtaining its execution time and its complexity. In Table 10, we can see the results of this analysis. We did the same with our system. So, we were able to compare the performance of the two systems. The first column indicates the name of each of the processes used by the algorithm of Bar-Yossef et al. [5] or by our system. The next columns indicate a brief description of the steps of each process, their computational complexity, the processing time, and which of the systems use this process.

We can see that the system proposed by Bar-Yossef et al. always has to make two requests (P1 and P1'). One request is to a legitimate web page and another request is to a random page, to check if the server returns a 200 HTTP code when the web server receives a request for an unknown page. They always need to make two requests because although one request is sufficient per "*parent*" URL path, the system should not rely since the content of the "parent" can change, which is very common due to the use of content dynamically generated. For the P1 process, Bar-Yossef et al. use a maximum execution time of 10 seconds. We are aware that to use timings from the 2004 for the latency of the network is not correct, but we did not exist any other solution, and we have not obtained the benefit of this, since we have also used this time for our system. We will show the comparative analysis of execution times below.

Also, the system of Bar-Yossef et al. always needs to check the number and the destination of the redirections of each performed request. If the system cannot decide if a page is a Soft-404 page or not, the system will have to perform a content analysis of each of the two obtained pages. The P3 process compares identical content, but also similar content (nearly-identical) by a process of "*shingling*" [9], which has a high computational cost.

In our case the Soft404Detector only runs one process, P1, and does not do anything if the server returns a Hard 404. In other cases, it only extracts the results of the heuristics, P4, and executes the classification process, P5. It does not have to make any other checks.

P.	Steps	Complexity	Time (sec)	Bar-Yossef et al.	Soft404 Detector
P1	Parse the URL			Yes	Yes
	DNS resolution			Yes	Yes
	HTTP Get	O(1)	10	Yes	Yes
	Check HTTP code			Yes	Yes
	Follow redirects			Yes	Yes
P1'	Same steps as P1				
	for a random generated page	O(1)	10	Yes	No
P2	Check num. redirects				
	Check redirects	O (N)	$t1$	Yes	No
				No	No
P3	Compare content	O (N^2)	$t2$	Yes	No
P4	Execute the heuristics	O (N)	0.0001	No	Yes
P5	Classify the URL	O (log N)	0.00018	No	Yes

Table 10. Processes, complexity and processing time of each detection system

The time of the P4 and P5 processes was obtained by measuring the time taken for the execution of all the proposed heuristics on a web page and the time taken for the classification algorithm, respectively.

The results of the analysis are shown in Table 11. The complexity of P1 and P1' is constant, i.e., $O(1)$. However, the complexity of $P2$ depends of the number of redirections, therefore its complexity is lineal, $O(N)$, where N is the number of redirections. The complexity of P3 is $O(N^2)$, where N represents the number of characters of the web page, because the system of Bar-Yossef et al. has to

	Complexity		Processing time (sec)	
	Best case	Worst case	Best case	Worst case
Dataset #1-A				
Bar-Yossef *et al.*	O(1)	$O(N^2)$	20	$20 + t1^5 + t2^6$
Soft404Detector	O(1)	O(N)	10.00028	10.00028

Table 11. Complexity and processing time of each Soft-404 detection system

Bar-Yossef *et al.*

	P1 O(1)	P1 O(1)	P2 O(1)	P3 $O(N^2)$
time (sec)	10	10	t1	t2

Soft404Detector

	P1 O(1)	P4 O(1)	P5 O(1)
time (sec)	10	0.0001	0.00018

Figure 8. Execution sequence of each system

compare all of the content (character by character) of two web pages, and this is very costly. On the other hand the complexity of the process P4 is also $O(N)$, where N represents the size of the web page. Finally, the process P5 traverses the corresponding decision tree, which is a binary tree and its traversal has a complexity of $O(\log N)$ where N is the depth of the tree. However, in this case, we know the depth of the tree because it was created previously in the training process. For that, the complexity of the process P5 is constant, i.e., 0(1). In summary, our system, in the best case, has the same complexity as the proposed by Bar-Yossef et al., O (1), but in the worst case, has a complexity of $O(N)$, while Bar-Yossef et al. has $O(N^2)$.

Regarding the execution time analysis, although we do not have the value of t1 and t2, since it was not published in their work, we can see that their system always has to make two requests (P1 and P1'). Our system will always be faster, since it uses just one request ($P1$). In the best case, we improve from 20 seconds to 10 seconds, and in the worst case, our system, takes only 10.00028 seconds versus $20 + t1 + t2$ seconds of the Bar-Yossef et al. system.

To summarize the analysis, we can see in Figure 8 a sequence diagram of the processes (including time and complexity information) of both the systems.

7. Conclusions

In this article, we discuss the importance of minimizing the use of resources in high performance environments such as crawling systems. Crawlers should not waste resources by accessing, processing and indexing pages without content or whose content does not represent the content that was assumed when the link was followed.

We have shown that 7.35% of the servers return Soft-404 pages when we do a request for unknown pages. In our opinion, it is a significant number, indicating that the problem should be studied.

We have proposed a set of content-based heuristics that characterize Soft-404 pages and help to identify them. We have not found studies addressing this issue from the content analysis. These results were used as input to several classification algorithms. Finally, we chose the C4.5 algorithm and improved the results by using Boosting and Bagging.

Because there was no similar study in the state of the art, we built two datasets of Soft-404 pages, which allowed us to test our heuristics and to obtain the results of the classifier. In the dataset #1, we achieved a precision of 0.928 and a recall of 0.983 on Soft-404 pages and a precision of 0.983 and a recall of 0.952 on normal pages. In the dataset #2, we achieved a precision of 0.996 and a recall of 0.987 on Soft-404 pages and a precision of 0.987 and a recall of 0.996 on normal pages.

We have compared the results of our system with existing methods in the supported literature. Concretely, we have compared our work with the presented by Bar-Yossef et

[5] Processing time of a process with complexity $O(N)$

[6] Processing time of a process with complexity $O(N^2)$

al. [5], and Lee et al. [27]. On the one hand, we have analyzed the advantages of our system with respect to the limitations of the above, since they do not detect all kinds of Soft-404 pages and there are scenarios in which their methods fail. On the other hand, we have developed the algorithm of Bar-Yossef et al., and we have tested it against datasets presented in this article.

Summarising the obtained results, our system improves the recall obtained by Bar-Yossef et al., to detect Soft-404 pages, approximately in a 5%, and the precision approximately a 25%. In the case of the Normal pages (no-soft-404 pages), our system improves the precision and the recall approximately a 6% and a 32%, respectively.

Due the system of Lee et al. [27] follows a perspective completely different from ours, we have compared both systems from a theoretical point of view. In the article, we have demonstrated that our system works better and in more cases than the proposed by Lee et al. [27].

We also discuss a performance analysis of the proposed heuristics. We compare the performance of our system with that of Bar-Yossef et al. [5], demonstrating that our system takes less than half the time and has a complexity in the worst case of $O(N)$ versus the complexity $O(N^2)$ of the system proposed by Bar-Yossef et al. [5].

To the best of our knowledge, the results we show are the best within the current literature on this topic. Due to its effectiveness and performance, we consider that it can be used in crawling systems, where the use of resources is a crucial task.

8. Future Works

A future study will be to design and implement the architecture of a crawling system, which allows the inclusion of a module for detection of Soft-404 pages. With this, the crawler avoids the Soft-404 pages, and we can study the real improvement in the performance of the crawler.

Another idea is to study other types of garbage documents (Web Spam, duplicate contents, etc.) and design methods and systems to detect them. This will improve the performance and reduce the number of the indexed documents, to help to improve the user experience.

As a complementary study to the latter, we would like to discuss the possibility of generalising the Soft404Detector to a distributed architecture, based on the Map-Reduce technique [13], which is currently the most used in crawling environments.

9. Acknowledgements

This research was supported by Xunta de Galicia CN2012/211, the Ministry of Education and Science of Spain and FEDER funds of the European Union (Project TIN2009-14203).

References

[1] Akamine, S., Kato, Y., Kawahara, D., Shinzato, K., Inui, K., Kurohashi, S., Kidawara, Y. (2009). Development of a large-scale web crawler and search engine infrastructure. *In*: Proceedings of the 3rd International Universal Communication Symposium, IUCS '09, p. 126–131. ACM, New York, NY, USA.

[2] Amitay, E., Carmel, D., Darlow, A., Lempel, R., Soffer, A. (2003). The connectivity sonar: detecting site functionality by structural patterns. *In*: Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia, p. 38–47. ACM Press.

[3] Baeza-Yates Ricardo; Ribeiro-Neto, B. (2011). Modern information retrieval: the concepts and technology behind search. Addison-Wesley Longman Publishing Co., Inc.

[4] Baeza-Yates, R., Castillo, C., Efthimiadis, E. N. (2007). Characterization of national web domains. ACM Trans. Internet Technol. p. 7.

[5] Bar-Yossef, Z., Broder, A. Z., Kumar, R., 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, p. 328–337. ACM, New York, NY, USA.

[6] Bar-yossef, Z., Keidar, I., Schonfeld, U. (2007). Do not crawl in the dust: different urls with similar text. *In*: Proc. 15th WWW, p. 1015–1016. ACM Press.

[7] Benczur, A. A., Csalogany, K., Sarlos, T., Uher, M., Uher, M. (2005). Spamrank - fully automatic link spam detection. *In*: Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (AIRWeb.

[8] Bergman, M. K. (2001). The Deep Web: Surfacing Hidden Value. *Journal of Electronic Publishing*, 7 (1).

[9] Broder, A. Z., Glassman, S. C., Manasse, M. S., Zweig, G. (1997). Syntactic clustering of the web. *In*: Selected papers from the sixth international conference on World Wide Web, pp. 1157–1166. Elsevier Science Publishers Ltd., Essex, UK. DOI http://dx.doi.org/10.1016/S0169-7552(97)00031-7

[10] Cambazoglu, B. B., Baeza-Yates, R. (2011). Scalability challenges in web search engines. *In*: M. Melucci, R. Baeza-Yates (eds.) Advanced Topics in Information Retrieval, The Information Retrieval Series, 33, p. 27–50. Springer Berlin Heidelberg.

[11] Castillo, C., Davison, B. D. (2010). Adversarial web search, 4 (5) 377–486.

[12] Chellapilla, K., 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, p. 81–88. ACM, New York, NY.

[13] Dean, J., Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. Commun. ACM, 51, p. 107–113.

[14] Donghua Pan Shaogang Qiu, D. Y. (2008). Web page content extraction method based on link density and statistic. *In*: Wireless Communications, Networking and Mobile Computing. WiCOM '08. 4th International Conference, p. 1–4.

[15] Fetterly, D., Manasse, M., 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, WebDB '04, p. 1–6. ACM, New York, NY, USA.

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

[17] Gulli, A., Signorini, A. (2005). The indexable web is more than 11.5 billion pages. *In*: Special interest tracks and posters of the 14th international conference on World Wide Web, WWW '05, p. 902–903. ACM, New York, NY, USA.

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

[19] Gyöngyi, Z., Garcia-Molina, H. (2005). Link spam alliances. *In*: Proceedings of the 31st international conference on Very large data bases, VLDB '05, p. 517–528. VLDB Endowment.

[20] Hafri, Y., Djeraba, C. (2004). High performance crawling system. *In*: Proceedings of the 6th ACM SIGMM international workshop on Multimedia information retrieval, MIR '04, p. 299–306. ACM, New York, NY, USA.

[21] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I. H. (2009). The weka data mining software: an update. SIGKDD Explor. Newsl. 11, p. 10–18.

[22] Henzinger, M. R., Motwani, R., Silverstein, C. (2002). Challenges in web search engines. SIGIR Forum, 36, p. 11–22.

[23] Hidalgo, J. M. G. (2002). Evaluating cost-sensitive unsolicited bulk email categorization.

[24] Kimball, A., Michels-Slettvet, S., Bisciglia, C. (2008). Cluster computing for web-scale data processing. SIGCSE Bull. 40, p. 116–120.

[25] Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. *In*: Proceedings of the 14th international joint conference on Artificial intelligence - 2, IJCAI'95, p. 1137–1143. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

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

[27] Lee, T., Kim, J., Kim, J. W., Kim, S. R., Park, K. (2009). Detecting soft errors by redirection classification. *In*: Proceedings of the 18th international conference on World wide web, WWW'09, p. 1119–1120. ACM, New York, NY, USA.

[28] Meneses, L., Furuta, R., Shipman, F. (2012). Identifying "*soft* 404" error pages: analyzing the lexical signatures of documents in distributed collections. *In*: Proceedings of the Second international conference on Theory and Practice of Digital Libraries, TPDL'12, p. 197–208. Springer-Verlag, Berlin, Heidelberg.

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

[30] Olston, C., Najork, M. (2010). Web crawling. Found. Trends Inf. Retr. 4 (3) 175–246.

[31] Quinlan, J. R. (1993). C4.5: programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

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

[33] Quinlan, J. R. (1996). Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research,* 4, p. 77–90.

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

[35] Ricardo Baeza-Yates, C. C., Lopez, V. (2005). Characteristics of the web of spain.

[36] Sahami, M., Dumais, S., Heckerman, D., Horvitz, E. (1998). A bayesian approach to filtering junk e-mail.

[37] Schmidt, W. C. (1997). World-wide web survey research: Benefits, potential problems, and solutions. Behavior Research Methods, Instruments, and Computers 29, p. 274–279.

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

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

[40] Zhang, H., Goel, A., Govindan, R., Mason, K., Van Roy, B. (2004). Making Eigenvector- Based Reputation Systems Robust to Collusion, Lecture Notes in Computer Science, 3243/2004, p. 92–104. Springer Berlin / Heidelberg.

Author Biographies

Víctor M. Prieto is currently the CEO of his own business in web information services. He received his Bachelor's Degree in Computer Engineering from the Universidade da Coruña in 2007 and a Ph.D. Degree in Computer Science in Department of Information and Comunications Technologies at the same university in 2013. His main research fields are web crawling, Hidden Web and Web Spam. He has also published several papers in international journals and has participated in multiple international conferences.

Manuel Álvarez is an Associate Professor in the Department of Information and Communication Technologies, at the Universidade da Coruña (Spain). He received his Bachelor's Degree in Computer Engineering from the Universidade da Coruña in 1999 and a Ph.D. Degree in Computer Science from the same University in 2007. His research interests are related to information retrieval and data integration. Manuel has managed several projects at national and regional level in the field of data integration and Hidden Web accessing. He has also published several papers in international journals and has participated in multiple international conferences.

Fidel Cacheda is an Associate Professor in the Department of Information and Communications Technologies at the Universidade da Coruña (Spain). He received his Ph.D. and B.S. degrees in Computer Science from the University of A Coruña, in 2002 and 1996, respectively. He has been involved in several research projects related to Web information retrieval and multimedia real time systems. His research interests include Web information retrieval and distributed architectures in information retrieval. He has published several papers in international journals and has participated in multiple international conferences.