Analysing the Effectiveness of Crawlers on the Client-Side Hidden Web

Víctor M. Prieto, Manuel Álvarez, Rafael López-García and Fidel Cacheda

Abstract The main goal of this study is to present a scale that classifies crawling systems according to their effectiveness in traversing the "client-side" Hidden Web. To that end, we accomplish several tasks. First, we perform a thorough analysis of the different client-side technologies and the main features of the Web 2.0 pages in order to determine the initial levels of the aforementioned scale. Second, we submit a Web site whose purpose is to check what crawlers are capable of dealing with those technologies and features. Third, we propose several methods to evaluate the performance of the crawlers in the Web site and to classify them according to the levels of the scale. Fourth, we show the results of applying those methods to some OpenSource and commercial crawlers, as well as to the robots of the main Web search engines.

1 Introduction

The World Wide Web (WWW) is currently the biggest information repository ever built. There are huge quantities of information that is publicly accessible, but as important as the information itself is being able to manage it to find, retrieve and gather the most relevant data according to users' needs in every moment.

The programs that process the Web in order to achieve that goal are called crawlers. A crawler traverses the Web following the URLs it discovers in a certain order and analyses the content of each document to obtain new URLs that will be processed later.

From their origins, crawling systems have had to face a lot of difficulties when they access human-oriented Web sites because some technologies are very hard to analyse (navigation through pop-up menus, different data layers that hide out or

Department of Information and Communication Technologies, University of A Coruña, Campus de Elviña s/n. 15071 A Coruña, Spain, e-mail: {victor.prieto, manuel.alvarez, rafael.lopez, fidel.cacheda}@udc.es

appear depending on users' actions, redirection techniques, etc.). The pages that can only be accessed through these technologies constitute what we call Hidden Web [3] and, particularly, the set of pages that are "hidden" behind client-side technologies are called the "client-side Hidden Web".

The objective of this paper is to present a scale for classifying crawlers according to their treatment of the client-side Hidden Web. To that end, we have accomplished some tasks. First, we have analysed the most important client-side technologies, such as JavaScript, VBScript, AJAX, and Flash, as well as some techniques that are often used for illicit purposes, like Redirection Spam [4] and Cloacking [16]. Once those technologies have been analysed, we have enumerated the difficulties that crawlers can find during their traversal. We have also developed a Web site that generates links dynamically and in accordance with the enumerated difficulties in order to know how existing crawlers behave towards them. For this task, we took into account the crawlers of the main Web search engines, as well as for some OpenSource crawlers [6] [9] and some others with a commercial licence. Then, we have discussed some methods to assess the effectiveness of the crawlers, allowing us to choose the appropriate levels in the scale for them.

The structure of this article is as follows. Section 2 discusses the related works. Section 3 introduces the client-side technologies and how they are used to build dynamic Web sites and defines the scale proposed for classifying crawling systems according to their effectiveness in traversing the client-side Hidden Web. Section 4 shows the experimental results performed with open-source and commercial crawlers. Finally, in Sections 5 and 6 we comment the conclusions we have reached and some possible future works.

2 Related Work

There are many studies about the size of the Web and the characterization of its content. However, there are not so many studies about classifying Web pages taking into account the difficulty for crawlers to process their content. According to the data submitted in *W3techs*[1] and *BuiltWith*[2] currently the 90% of the Web pages use JavaScript. In 2006, M. Weideman y F. Schwenke [15] published a study analysing the importance of JavaScript in the visibility of a Web site, concluding that most of the crawlers do not deal with it appropriately.

From the point of view of crawling systems, there are many works oriented to create programs that are capable of traversing the Hidden Web [3]. Server-side Hidden Web crawlers deal with a large quantity of Web sites whose content is accessed by means of forms. This kind of content is copious and has excellent quality. There are some researches that tackle the challenges established by the server-side Hidden Web. We highlight HiWE [11] because it is one of the pioneer systems. Google [7]

[1] http://w3techs.com/

[2] http://builtwith.com/

also submitted the techniques that they use to access information through forms. Álvarez *et al.* show DeepBot [2], a prototype of hidden-web crawler able to access hidden content, identifying automatically the web forms and learning to execute queries on them.

Regarding the client-side Hidden Web, there are less studies. From of our knowledge, this is because the crawlers of major search engines have agreements with companies to access data from their servers. However, they only have agreements with major companies in each sector, and a normal crawler has no such agreements.

However, there have been several studies that analyze the issue. Álvarez *et al.* [1] propose the usage of mini-browsers to execute the client-side technologies and so, have access to the hidden content. In 2008, Mesbah *et al.* show a study [8] about the usage of AJAX on the Web, and as can be processed to have access to the data.

On the other hand, as client-side technologies can be used to "deceive" crawling systems, there are some works about detection of what is known as Web Spam (Cloacking [16] [18] [17] [5] or Redirection Spam [4] [5]).

Nevertheless, there is not any scale that allows researchers to classify the effectiveness of the crawling systems according to their level of treatment of client-side Hidden Web technologies.

3 The Scale for Web Crawlers

The client-side technologies are normally used to improve the users' experience, generating content and links dynamically according to users' actions. Among the most used are: JavaScript; AJAX; Flash;Applet and VBScript. In order to create the scale, we have analysed how designers use the aforementioned technologies to build dynamic sites. The following features have been identified:

- Text links, which constitute the lowest level of the scale.
- Simple navigations, generated with JavaScript, VBScript or ActionScript. This includes links that are generated by means of "document.write()" or similar functions, which allows designers to add new links to the HTML code dynamically.
- Navigations generated with an Applet. We divide them in two types: those which are generated from a URL that is passed to the Applet as an argument and those whose URL is created as a string into the compiled code.
- Navigations generated by means of AJAX.
- Pop-up menus, generated by a script code that is associated to any event.
- Navigations generated with Flash. There are two kinds: those which receive the URL as an argument from the HTML code and those which define it inside the ActionScript code.
- Links that are defined as strings in .java files, .class files or any other kinds of text and binary files.
- Navigations generated in script functions. The script can be embedded inside the HTML or it can be located inside an external file.

- Navigations generated by means of several kinds of redirections: a) those specified in the `<meta>` tag; b) generated by the onLoad event of the `<body>` tag; c)generated by a script when an event in other page (e.g.: the onClick event) is fired; d) embedded in script blocks; e) executed in an applet f) Fash redirections.

In addition, the navigations that are generated with any of the identified methods can create absolute or relative URL addresses. For the addresses that are built with scripting languages, it is possible to recognize the following construction methods: a) a static string inside the script; b) a string concatenation; c) execution of a function that builds the URL in several steps.

On the other hand, the different methods we enumerated before can be combined. For example, some Web sites build pop-up menus dynamically, by means of "document.write()" functions. The number of possibilities is unapproachable. Hence, this study only takes into account a reduced but significative subset. It consists of 70 "scenarios" (see "Description" and "Tested Scenarios" columns in Figure 1) that represent the basic types from which the rest of the cases could be obtained by means of combinations. The number of scenarios taken into account could be higher, but it would not get more information about the use of client-side technologies in the Web or about the methods that crawlers use to discover links. For instance, it is not necessary to check the combination of menus and "document.write()" because we can deduce the result from the two base cases that were included separately.

Starting from the aforementioned 70 scenarios, we proposed an initial grouping based on the technologies, the methods for building strings, the location of the code and if the URLs are absolute or relative. This way, we have grouped the scenarios that presented a similar difficulty for crawlers and we have also sorted them by complexity. Figure 1 shows the 8 step scale. The steps represent the capacity to treat the client-side Hidden Web from lower to higher level of complexity.

Level	Description	Tested Scenarios
1	Text Link	1,2
2	JavaScript/Document.Write/Menu – Static String – Embedded	3, 4, 15, 16, 27, 28
	JavaScript – Concatenated String – Embedded	6
3	Redirect HTML/onBody/JavaScript	51, 52, 53, 54, 55, 56
	JavaScript con # – Static String – Embedded	63,64
	VbScript – Static String – Embedded	67,68
4	VbScript – Special Function - Embedded	70
5	JavaScript/Document.Write – Static String – External/Embedded	9, 10, 18
	Document.Write/Menu – Static String – External	21, 22, 33, 34
	Menu – Concatenated String – Embedded	30
6	JavaScript/Document.Write/Menu – Concatenated String – External	12, 24, 36
	Applet – Static String in HTML	43,44

Level	Description	Tested Scenarios
7	JavaScript – Concatenated String – Embedded – Relative	5
	JavaScript – Special Function – Embedded	7,8
	JavaScript – Concatenated String – External – Relative	11
	JavaScript – Special Function – External – Relative	13,14
	Document.Write – Concatenated String – Embedded/external – Relative	17,23
	Document.Write – Special Function – Embedded/External	19, 20, 25, 26
	Menu – Concatenated String – Embedded/external – Relative	29,35
	Menu – Special Function – Embedded/External	31, 32, 37, 38
	Link in .class	41,42
	Ajax Link – Absolute	62
8	Link in .java	39,4
	Applet – Static String in .class	45,46
	Flash – Static String in HTML/SWF	47, 48, 49, 50
	Redirect Applet/Flash	57, 58, 59, 60
	Ajax Link – Relative	61
	JavaScript with # – Special Function – Embedded	65,66
	VbScript – Special Function – Embedded	69

Fig. 1 Link classification by difficulty

Once the scale has been defined, in order to classify the different crawling systems according to the level of complexity of the "links" they process, we propose the following evaluation methods:

- Simple Average: it treats all the scenarios in the same way, without taking into account their difficulty. It shows the crawlers which treat the highest number of scenarios, so they pay more attention to the Hidden Web in general.
- Maximum Level: this model sorts crawlers according to the highest level of difficulty they can process. A crawler obtains a score i if it has the capacity of processing the scenarios of that level and the levels below. There are some crawlers that process a certain level, but they cannot obtain pages from scenarios of lower level. This could be due to some problems like the low PageRank of a Web page and so on. However, this evaluation method assumes that if a crawler is capable of dealing with a level i, it should be able to deal with lower ones.
- Weighted Average: each scenario is assigned a value between 0 and 1, which depends on the number of crawlers that have been able to process it (0 if every crawler has been able to deal with it). This method shows what crawlers can obtain the highest number of difficult resources in the client-side Hidden Web, or resources that most crawlers do not reach.
- Eight Levels: in this model each level has a value of one point. If a crawler processes all the scenarios of one level it obtains that point. For every scenario that the crawler processes successfully, it gets $1/n$ points, where n is the total number of scenarios that were defined for that level.

4 Experimental Results

In order to check how crawling systems deal with the different scenarios, and for ranking them using the scale defined, we created a web site for performing experiments. The "jstestingsite"[3] web site contains 70 links, one for each scenario defined in the scale. We employed it to test the crawlers of the main Web search engines (Google, Bing, Yahoo!, PicSearch and Gigablast) and other OpenSource and commercial crawlers (Nutch [6], Heritrix [9], Pavuk [10], WebHTTrack, Teleport [12], Web2Disk [14], WebCopierPro [13]).

4.1 Summary Results

The left side of Figure 2 shows the results obtained for OpenSource and commercial crawlers. The crawler that achieves the best results is WebCopierPro, which processed 57,14% of the levels, followed by Heritrix with 47,14% and Web2Disk with 34,29%. Only a few get values beyond 25% in most of the levels. It is also

[3] http://www.tic.udc.es/∼mad/resources/projects/jstestingsite/

important to notice the poor results that they obtained for redirections, especially in the case of WebCopierPro which was not able to deal with any of them, although it gets results of 100% in harder levels. None of the crawlers reached the 100% in redirections. This happens because none of them has been able to process pages with redirections embedded in Applets or Flash, since these technologies were not executed.

	Heritrix	Nutch	Pavuk	Teleport	Web2Disk	WebCopierPro	WebHTTrack	Google	Yahoo
Text Link	2 – 100%	2 – 100%	2 – 100%	2 – 100%	2 – 100%	2 – 100%	2 – 100%	2 – 100%	1 – 50%
Href="javaScript Link	6 – 50%	3 – 25%	0 – 0%	4 – 33%	5 – 42%	12 – 100%	3 – 25%	6 – 50%	0 – 0%
Document.write Link	6 – 50%	3 – 25%	0 – 0%	3 – 25%	4 – 33%	12 – 100%	2 – 17%	6 – 50%	0 – 0%
Menu Link	6 – 50%	3 – 25%	0 – 0%	3 – 25%	4 – 33%	12 – 100%	2 – 17%	6 – 50%	0 – 0%
Flash Link	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	2 – 50%	0 – 0%	0 – 0%
Applet Link	2 – 50%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	2 – 50%	0 – 0%	0 – 0%
Redirects	6 – 60%	6 – 60%	2 – 20%	6 – 60%	4 – 40%	0 – 0%	6 – 60%	6 – 60%	2 – 20%
Class or Java Links	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	2 – 50%	0 – 0%	0 – 0%	0 – 0%
Ajax Link	0 – 0%	1 – 50%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%	0 – 0%
Links with #	2 – 50%	2 – 50%	0 – 0%	3 – 75%	2 – 50%	0 – 0%	2 – 50%	4 – 100%	0 – 0%
VbScript Link	3 – 75%	3 – 75%	0 – 0%	3 – 75%	3 – 75%	0 – 0%	2 – 50%	1 – 25%	0 – 0%
Static String Link	26 – 62%	19 – 45%	4 – 10%	18 – 43%	22 – 52%	16 – 38%	22 – 52%	17 – 40%	3 – 7%
Concatenated String Link	6 – 50%	2 – 16%	0 – 0%	1 – 8%	1 – 8%	12 – 100%	1 – 8%	6 – 50%	0 – 0%
Special Function String Link	1 – 6%	2 – 13%	0 – 0%	5 – 31%	1 – 6%	12 – 75%	0 – 0%	8 – 50%	0 – 0%
Tests passed	33 – 47%	23 – 33%	4 – 6%	24 – 34%	24 – 34%	40 – 57%	23 – 32%	31 – 44%	3 – 4%

Fig. 2 Summary of results of OpenSource and Commercial crawlers (left side) and the crawlers of the main search engines (right side)

The right side of Figure 2 contains the results obtained for the main Web search engines. It does not show the results for Bing, Gigablast and PicSearch, since they have not indexed the testing Web site. Only Google and Yahoo! have indexed it. Google has processed 44.29% of the links. GoogleBot has processed 50% of many of the proposed levels. The other half has not been processed because the crawler has not analysed some external files. If it was the case, GoogleBot would achieve much better results. The links that GoogleBot has not processed included technologies like Flash, Applets and AJAX or files like .class and .java.

Regarding the types of links that in general were processed by the crawlers analyzed, we found that only 42.52% of statics links were retrieved, 27.38% of links generated by concatenation of strings and 15.18% for links that were generated by functions. These results show that the crawlers try to discover new URLs by processing the code as text, using regular expressions, instead of using scripting interpreters and/or decompilers. So, they cannot extract those links which were generated by a complex method.

4.2 Ranking the crawlers according to the Scale

Figure 3 shows the result of classifying the crawling systems according to the scale and the assessment levels we had proposed.

We can see that WebCopier, Heritrix and Google get the best results in the Simple Average method. For Maximum Level, Google places first since it processes level 8 links. It is followed by WebCopier (7 points) and Heritrix (6 points). As Google

	Heritrix	Nutch	Pavuk	Teleport	Web2Disk	WebCopier	WebHTTRack	Google	Yahoo
Simple Average	3,77	2,63	0,46	2,29	2,74	4,57	2,40	3,54	0,34
Maximum Level	6,00	7,00	3,00	4,00	5,00	7,00	6,00	8,00	3,00
Weighted Average	1,63	0,99	0,14	0,75	1,03	3,42	0,86	2,34	0,11
Eight levels	6,00	4,30	1,20	4,00	4,55	4,55	3,40	3,70	0,70

Fig. 3 Results according to the proposed scales

achieves the maximum level in this model but not in others, we can conclude that it does not try some scenarios because of its internal policy. Once again, WebCopier, Google and Heritrix have obtained the best results in the Weighted Average model. Very similar results have been obtained in the Eight Levels method. This means that the three top crawlers have dealt with a big quantity of levels in each group or they have gone through links that were part of a group with few links, which makes each link more valuable. We conclude that the best crawlers in both quantity and quality are Google and WebCopier, followed by Heritrix, Nutch and Web2Disk. It is important to highlight the results of GoogleBot. Although it is oriented to traverse all the Web and it has a lot of performance and security requisites, it takes into account a wide range of technologies.

5 Conclusions

This article proposes a scale that allows us to classify crawling systems according to their effectiveness accessing the client-side Hidden Web. In order to classify the different crawling systems, we have created a Web site implementing all the difficulties that we had included in the scale.

Analyzing the results we can make the following recommendations about creating a web page: use JavaScript, embed JavaScript code in HTML, avoid dynamic creation of URLs and use HTTP or HTML code for the redirects.

The best crawlers are Google and WebCopier, followed by Heritrix, Nutch and Web2Disk. We can say that most of the times they try to discover new URLs processing the code as text, using regular expressions. This allows them to discover a big amount of scenarios. In addition, the major crawlers have agreements with companies in each sector, allowing them direct access to data. With these policies, the crawlers can treat directly the data and save resources. However, only the major search engines have these agreements. We conclude that in present, most of the URLs which are located in the client-side technologies are not discovered.

6 Future Work

Among the studies that we propose as a continuation of this work, we have the improvement of the assessment methods in order to take into account the frequency

of use of each technology on the Web. This will allow us to know the volume of information that is beyond the scope of the crawlers, since nowadays they do not treat all the scenarios. We can use this information to know the convenience of analysing the client-side Hidden Web. We also are interested in in studying how the features (the topics, the number of visits, etc.) of a Web site can affect the crawling process and the indexation of its pages.

7 Acknowledgments

This work was partly supported by the Spanish government, under projects TIN 2009-14203 and TIN 2010-09988-E.

References

1. M. Álvarez, A. Pan, J. Raposo, and Justo Hidalgo. Crawling Web Pages with Support for Client-Side Dynamism, 2006
2. M. Álvarez, J. Raposo, A. Pan, F. Cacheda, F. Bellas, and V. Carneiro. Crawling the content hidden behind web forms In *Computational Science and Its Applications - ICCSA 2007*, Vol. 4706 *Lecture Notes in Computer Science*, pages 322–333, 2007.
3. M. K. Bergman. The deep web: Surfacing hidden value, 2000.
4. K. Chellapilla and A. Maykov. A taxonomy of javascript redirection spam. In *Workshop on Adversarial information retrieval on the web*, AIRWeb 2007, pages 81–88.
5. Z. Gyongyi and H. Garcia-Molina. Web spam taxonomy, 2005.
6. R. Khare and D. Cutting. Nutch: A flexible and scalable open-source web search engine. Technical report, 2004.
7. J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's deep web crawl. *Proc. VLDB Endow.*, 1:1241–1252, August 2008.
8. A. Mesbah, E. Bozdag, and A. van Deursen. Crawling ajax by inferring user interface state changes. In *Web Engineering, 2008. ICWE '08.*, pages 122–134, 2008.
9. G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to heritrix, an archival quality web crawler. In *4th International Web Archiving Workshop (IWAW04)*, 2004.
10. Pavuk Web page. http://www.pavuk.org/, 2011.
11. S. Raghavan and H. Garcia-Molina. 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, 2001. Morgan Kaufmann Publishers Inc.
12. Teleport Web page. http://www.tenmax.com/teleport/pro/home.htm, 2011.
13. Web Copier Pro Web page. http://www.maximumsoft.com/products/wc_pro/overview.html.
14. Web2Disk Web page. http://www.inspyder.com/products/Web2Disk/Default.aspx, 2011.
15. M. Weideman and F. Schwenke. The influence that JavaScript has on the visibility of a Website to search engines - a pilot study. *Information Research*, 11(4), Jul 2006.
16. B. Wu and B. D. Davison. Cloaking and redirection: A preliminary study, 2005.
17. B. Wu and B. D. Davison. Identifying link farm spam pages. In *Proceedings of the 14th International World Wide Web Conference*, pages 820–829. ACM Press, 2005.
18. B. Wu and B. D. Davison. Detecting semantic cloaking on the web. In *Proceedings of the 15th International World Wide Web Conference*, pages 819–828. ACM Press, 2006.