Web Site Metadata

Erik Wilde and Anuradha Roy

School of Information
UC Berkeley

Abstract. Understanding the availability of site metadata on the Web is a foundation for any system or application that wants to work with the pages published by Web sites, and also wants to understand a Web site's structure. There is little information available about how much information Web sites make available about themselves, and this paper presents data addressing this question. Based on this analysis of available Web site metadata, it is easier for Web-oriented applications to be based on statistical analysis rather than assumptions when relying on Web site metadata. Our study of `robots.txt` files and sitemaps can be used as a starting point for Web-oriented applications wishing to work with Web site metadata.

1 Introduction

This paper presents first results from a project which ultimately aims at providing accessible Web site navigation for Web sites [1]. One of the important intermediary steps is to understand how much metadata about Web sites is made available on the Web today, and how much navigational information can be extracted from that metadata. Our long-term goal is to establish a standard way for Web sites to expose their navigational structure, but since this is a long-term goal with a long adoption process, our mid-term goal is establish a third-party service that provides navigational metadata about a site as a service to users interested in that information. A typical scenario would be blind users; they typically have difficulties to navigate Web sites, because most usability and accessibility methods focus on *Web pages* rather than *Web sites*. Following the extended principle of Web engineering as *blending into the Web* rather *building Web front-ends* [2], our approach is to first understand the current state of Web site metadata on the Web, before designing our service and data format. This paper describes our analysis of the current state of Web site metadata available on the Web.

Most information resources on the Web are *Web sites*, informally defined as a set of *Web pages* made available by some information provider. While the concept of a Web site is only loosely defined, it is often associated with all Web pages available under one DNS domain name (this could be generalized to all Web pages using the same URI prefix, but for the purpose of this paper, we look at domain-based sites only). For information gathering, Web sites are usually accessed by *Web crawlers* [3] which systematically retrieve Web pages, in

most cases to drive later stages of indexing them for eventually driving a search engine. To allow Web sites some level of control over crawlers, the informal `robots.txt` format [4] — sometimes also referred to as the *Robots Exclusion Protocol (REP)* — is the established way of how a Web site can control crawlers. This format can only be used on a per-domain basis, and specifies rules for all pages under that domain.

The `robots.txt` format is a simple way of how a site can publish metadata about itself; in that case with the sole purpose of controlling crawler access to the site (most often by limiting access to certain parts of a site). This assumes that crawlers get information about available URIs from other sources; in most cases this happens by following links on already crawled pages. On the other hand, sites often want to be crawled so that their contents are available through search engines, and the *sitemaps* format allows sites to publish lists of URIs which they want to advertise to crawlers. Sitemaps can be made available to crawlers in different ways; they can be directly advertised through user interfaces or an HTTP ping interface to individual crawlers, or they can be specified in the `robots.txt` file. While the name "sitemap" implies a certain structure to the URIs contained in that information, this information, in fact, is not a map. Despite its name, the sitemaps format only supports the publication of unstructured lists of URIs.

Sitemap information can be useful for exposing the *Deep Web* [5,6], for example, those pages that are accessible only through HTML forms. Because search engine crawlers typically discover pages by following links, large portions of the Web can be hidden from crawlers, and thus might never be indexed, and therefore never show up in search results. Hence, without sitemap information, search engine crawlers might not be able to find these pages. Since sitemap information may be incomplete and/or inaccurate, search engines have to rely on other techniques to completely crawl the deep Web.

The current Web site metadata already allows crawlers to get information about a site's structure, they can do so by using a Web site's URIs as they are listed in the `robots.txt` and sitemaps files, and if they are associated with a search engine, they can also use click data to learn about a site's popular pages. In that case, site metadata combined with the click data can be used for approximating a site's navigation structure. Figure 1 shows an example for such an algorithmically computed site map.

Site metadata on the one hand greatly improves the interaction of humans with a site, because many tasks on a site require accessing more than one page on the site. On the other hand, even though explicit navigation often is provided through Web page design, IR techniques can be used to algorithmically infer site metadata for tasks other than direct user interaction with a Web site. Google's search results, for example, occasionally include a small "site map" (called "sitelinks") for highly ranked search results (Figure 1 shows an example). This demonstrates the fact that site metadata can have useful applications beyond crawling, and since most Web sites use content management systems to publish their site anyway, exposing site metadata in a richer format than just sitemaps in many cases could be easily implemented by Web sites.
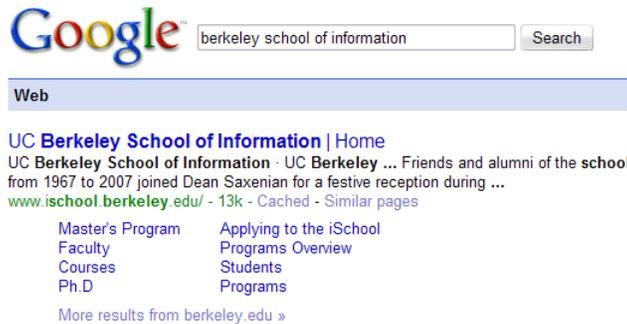
Fig. 1. Algorithmically Computed Site Map

This paper first presents a brief overview of how Web site metadata is managed from a Web engineering perspective (Section 2). We then describe the crawling process for `robots.txt` files and the results from that process (Sections 3 and 4). We continue by describing the crawling process for sitemaps files and the results from that process (Sections 5 and 6). We conclude the paper by describing related and future work (Sections 7 and 8).

2   Web Site Metadata on the Web

The *Robots Exclusion Protocol (REP)* or `robots.txt` format was never published as a formal document, the only "official" reference is an Internet draft [4] and various Web sites. The `robots.txt` has a well-defined discovery method: if a Web site is publishing such a file, it must be available at the URI path `/robots.txt` on that site. The file format itself is very simple; it started out as a way to "protect" certain parts of a Web site (defined by URI prefix) to be excluded from crawler access. And because it is possible that access should only be limited for certain crawlers, these exclusion rules can be made specific for certain user agents (which in case of `robots.txt` are not really user agents, but search engine crawlers).

The informal specification of the `robots.txt` file format only defines the three fields `User-Agent`, `Disallow`, and `Allow`. However, the specification does allow other fields as well, as long as they are based on the same basic syntax. Some other fields that are used are `Noindex`, `Crawl-Delay`, `Request-Rate`, and `Visit-Time` fields, which are defined by specific crawlers, and apparently Web site administrators seem to believe these fields are (at least potentially) interpreted by crawlers. Section 4 contains a more complete list of the fields in our sample of `robots.txt` files, as well as other statistics about that data set.

One additional field that can occur in a `robots.txt` file is `Sitemap`, which points to the URI of a sitemaps file as defined in the sitemaps protocol. While discovery through `robots.txt` is one possible way for a site to publish a sitemap, the protocol also defines the ability to submit a sitemap file to a search engine through a submission interface, or by HTTP ping. In these latter cases, the

sitemap file is only known to the search engine it has been submitted to, as there is no well-defined discovery method for it.

Sitemaps can use XML (using a simple schema), plain text, or feed formats (RSS 2.0 and Atom [7]) as their syntax, and it is allowed to compress them on the server side using *gzip* (HTTP transfer encoding works regardless of that, but sitemaps can be served as compressed documents). There are size limitations limiting a sitemap file to no more than 50'000 URIs and no more than 10MB in size. Furthermore, there are size limitations limiting an index file to no more than 1'000 URIs and no more than 10MB in size. For compressed files, these size limits apply to the uncompressed files.

Despite of their name, sitemaps are not really maps, because they do not contain any structure. Sitemaps are simply lists of links a site wants to be crawled, and in addition to the URI, the XML format supports parameters to set last modification data, change frequency, and priority for each URI. It is up to a crawler to decide how to use sitemap information, but it is likely that most search engine crawlers will take their data into consideration when computing their crawling process.

3   Crawling for Robots.txt

Our starting point is Alexa's dataset of the most popular 100'000 domains. This dataset has some bias, based on the way this dataset is collected. Even though the exact method of how the dataset is collected is not published, we chose to accept the bias, because our research does not depend on the exact ranking of popular domains, but instead just depends on a reasonably large set of popular domains. Based on this dataset, our crawling process requests `robots.txt` files from all domains.

Using a simple two-step process (trying `http://domain.com/robots.txt` and `http://www.domain.com/robots.txt` for all domain names), our crawl of 100'000 domains for `robots.txt` files yields 44'832 files (i.e., 44.8% of the domains make `robots.txt` files available); more detailed statistics about these files can be found in Section 4. Various error conditions can be encountered when requesting the files. We do not fully implement error recovery (such as trying to fix corrupted `robots.txt` files and retrying failed connection attempts), because error conditions are only encountered in a small fraction of cases. This means that our crawl yields slightly fewer `robots.txt` files than it could with a more robust crawling mechanism, but that is an acceptable compromise allowing a less complex crawler implementation.

4   Robots.txt Data Analysis

The `robots.txt` files crawled as described in Section 3 are mainly intended as a starting point to find sitemap information, as described in Section 5. However, because the available literature does not present a lot of data about large-scale

collections of `robots.txt` files, we first present some statistics about the dataset obtained in the first step of our study.
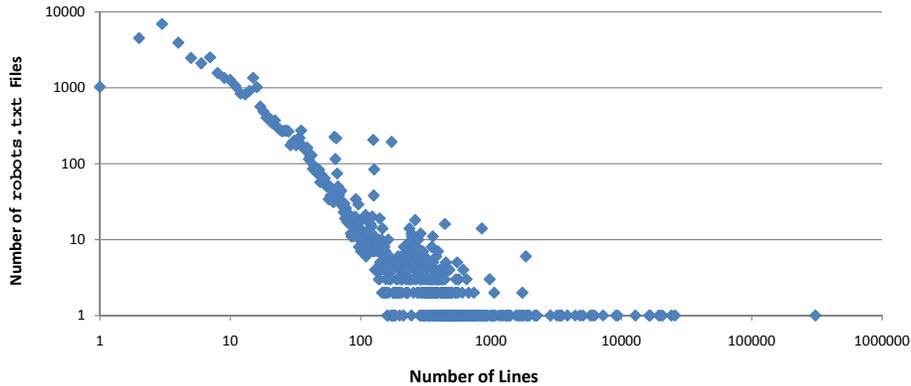

Fig. 2. Distribution of `robots.txt` Size

Figure 2 shows the distribution of the size of `robots.txt` files (in lines) over the number of `robots.txt` files. It is a heavy-tail distribution with the average size being 29.8 lines ($\sigma = 293.4$) with a median of 7 lines. Since there is a fair number of rather large `robots.txt` files in our dataset, we want to understand the reasons for these sizes. `robots.txt` files can become large for two reasons: because they contain individual configurations for a large number of user agents, or because they contain a lot of instructions for one user agent (or a combination of these two reasons). We therefore looked at how many individual configuration sections for specific user agents the `robots.txt` files contain.

Figure 3 shows the result of this analysis. Again, it is a heavy-tail distribution with an average of 6 sections ($\sigma = 29.5$) and a median of 2. However, in this case there is a noticeable peak in the long tail, with the center at `robots.txt` files having 120 user agent configuration sections.

Our assumption is that this peak has its origin in some widely used and reused template that originally had 120 configuration sections, and then was adapted for various sites by adding or removing some of these sections. There is a variety of templates and generators for `robots.txt` files available on the Web, so assuming that one of these gained popularity is a reasonable explanation of the peak around 120 configuration sections.

To better understand how current `robots.txt` files are using fields to steer crawlers, we looked at the overall usage of fields. As stated in Section 2, only the three fields `User-Agent`, `Disallow`, and `Allow` are defined by the `robots.txt` file format, but some other fields also have gained some acceptance. Table 1 contains a list of the ten most popular fields we found (sorted by the number of files containing this field, based on the dataset of 44'832 files), also listing how
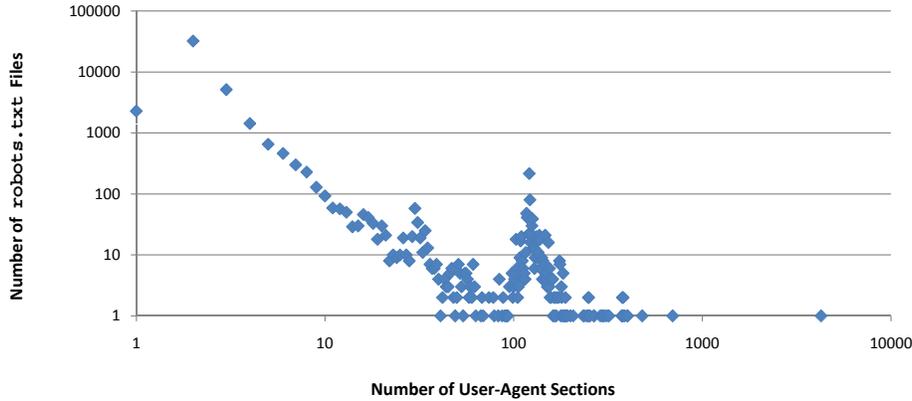
Fig. 3. `User-Agent` Sections per `robots.txt` File

many occurrences were found in total, and the average number of occurrences per file based on the number of files in which this field was used.

Table 1. Popular Fields in `robots.txt` Files

Field Name	#Files	#Fields	Fields/File
1. `User-Agent`	42'578	225'428	5.29
2. `Disallow`	39'928	947'892	23.74
3. `Sitemap`	6'765	10'979	1.62
4. `Allow`	3'832	23'177	6.05
5. `Crawl-Delay`	2'987	4'537	1.52
6. `Noindex`	905	2'151	2.38
7. `Host`	728	758	1.04
8. `Request-Rate`	121	127	1.05
9. `Visit-Time`	89	102	1.15
10. `ACAP-Crawler`	71	234	3.30

The three standard `robots.txt` fields are among the most frequently used ones, and the popularity of fields drops significantly after the top five. The `Sitemap` field points to a sitemap and is what we use for the second step of our crawling process (described in Section 5). Most of the other fields we found are fields only supported by particular crawlers, so if they do appear in an appropriate `User-Agent` section, they can control that particular crawler. One exception to these crawler-specific fields are `ACAP`-prefixed fields, which are part of the *Automated Content Access Protocol (ACAP)*. ACAP is an initiative of content providers to extend the `robots.txt` format so that it is possible to

express more specific policies about the crawled content, mostly about access and usage permissions for copyright-protected content.

The idea of `robots.txt` most often is to restrict crawlers from certain pages and paths on a site. This can make sense because of pages that are frequently updated, because of pages that contain content that should not be indexed (e.g., because of copyright issues), or because of crawlers that interact with the server in unfortunate ways when retrieving pages. This means that while some configurations in `robots.txt` files are global (i.e., apply to all crawlers), there are also some which are for specific crawlers only. We looked at the `User-Agent` fields in our dataset and counted the various strings listed there, trying to adjust for minor variations such as capitalization, whitespace, or version numbers.

Table 2. Popular User-Agents in `robots.txt` Files

User Agent	Occurrences	
1. `*`	46'645	20.70%
2. `Mediapartners-Google`	3'825	1.70%
3. `wget`	3'787	1.68%
4. `WebZIP`	3'014	1.34%
5. `Mozilla`	2'696	1.20%
6. `GoogleBot`	2'694	1.20%
7. `Microsoft URL Control`	2'647	1.17%
8. `WebBandit`	2'271	1.01%
9. `lwp-trivial`	2'227	0.99%
10. `MIIxpc`	2'180	0.97%

Table 2 lists the top ten `User-Agent` field values we found in our dataset (the total number of all fields was 225'304, the distribution of those fields across `robots.txt` files is shown in Figure 4). `*` is the catch-all value which is used to define rules applying to all crawlers; it is by far the most popular value. `Mediapartners-Google` is the crawler for sites participating in Google's *AdSense* program, and is the most frequently listed named crawler. `wget` and `WebZIP` are two similar "crawlers" which usually do not really crawl the Web, but instead are used to download the contents of a site; they are often used to download site contents for offline browsing or post-processing.

Many crawlers do not reveal their identity and use fake `User-Agent` field values to cloak themselves as browsers. The `Mozilla User-Agent` value is the most frequently used one and thus is listed in many `robots.txt` files; but if a crawler is misbehaving in the sense that it does not properly reveal its identity, it is unlikely that it will be sufficiently well-behaving to respect `robots.txt` configurations. `GoogleBot` is Google's search engine crawler (it is using a different identity than the AdSense crawler mentioned earlier). `Microsoft URL Control` is a default identity used within various Microsoft Web tools, and developers can either change that when they develop software using these tools, or leave it at

its default value. `WebBandit` is a tool similar to `wget` and `WebZIP`, in most cases not used as a crawler, but for targeted downloads of Web content. `lwp-trivial` is the default name used by the Perl module LWP::Simple. `MIIxpc` is a crawler about which there is no public information available, but apparently it is active enough to be listed in many `robots.txt` files.
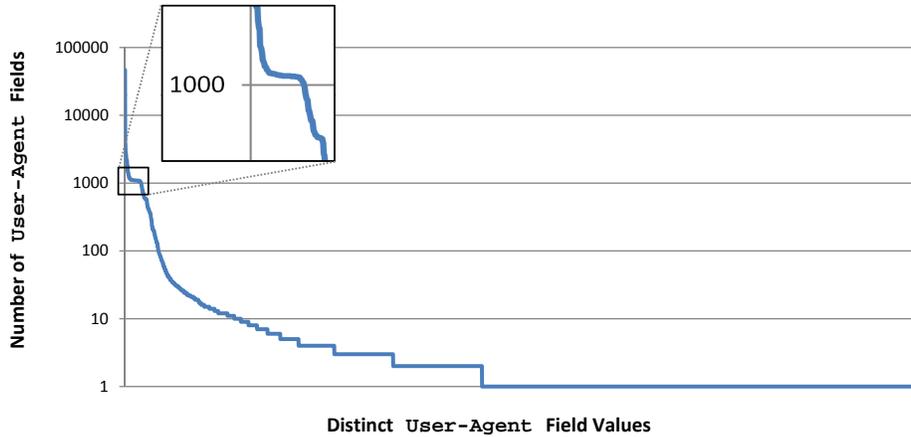

Fig. 4. Distribution of `User-Agent` Field Values

Figure 4 shows the distribution of occurrences of `User-Agent` fields. The horizontal axis linearly lists all 4'483 distinct `User-Agent` fields we found (Table 2 lists the top ten) sorted by the number of occurrences. It can be seen that more than half of the `User-Agent` values only occur once. The tableau in the distribution at about 1'000 occurrences (as magnified in the figure) is something that we believe to be caused by `robots.txt` files being created using templates or generators, which usually just present a list of predefined `User-Agent` values, and therefore the values available there will show up in many template-based or generated files.

5   Crawling for Sitemaps

Starting from the `robots.txt` files obtained as described in Section 3, the next step to get more complete Web site metadata is to crawl for the second Web site metadata format, the sitemaps format. The likelihood of a Web site providing sitemaps is substantially lower than that of it providing a `robots.txt` file, but on the other hand, the information found in sitemaps typically is more valuable, because it is much more specific in listing a Web site's actual page URIs, whereas `robots.txt` files typically only specify a small number of URI prefixes.

While we depend on sitemaps being available through `robots.txt` files, this only provides access to a subset of available sitemap information. Web sites can

also directly make sitemaps available to crawlers by uploading them or pinging crawlers to download a sitemap. However, these two methods depend on the Web site explicitly cooperating with the crawler, and therefore is not available to crawlers which have to depend on publicly available information.
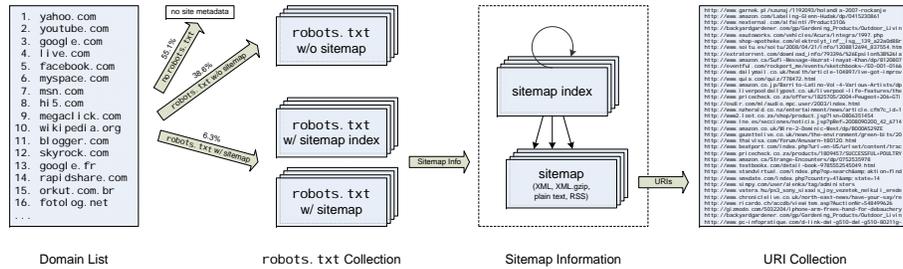

Fig. 5. Overview of the Crawling Process

Figure 5 shows an overview of the complete crawling process as it starts with the domain dataset and eventually creates a dataset of Web page URIs from those domains. In the starting dataset of 44'832 `robots.txt` files, 6'268 files (14%) contained `Sitemap` fields, for a total of 10'029 fields (it is legal for `robots.txt` files to reference more than one sitemap file; we found one pointing to 530 sitemap files).
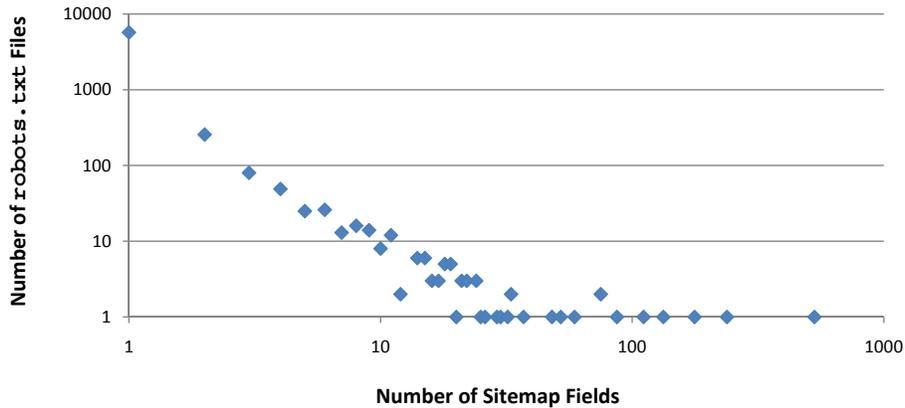

Fig. 6. `Sitemap` Fields per `robots.txt` File

Figure 6 shows the distribution of `robots.txt` files according to how many references to sitemaps they contained (`robots.txt` files with no references are

not shown in this figure). The vast majority of `robots.txt` files (5'710 or 91%) specify only one sitemap reference, but there also is a considerable number of `robots.txt` files pointing to more than one sitemap file.

The first task when crawling for sitemaps is to navigate sitemap indices and sitemap files, so that all sitemap information for a given site can be retrieved. The sitemaps specification is silent on whether index files may point to index files, but since it is not specifically disallowed, it is probably allowed, and there are sites that make use of that assumption. As one example of sitemap information crawled from one company, Table 3 shows the number of sitemaps/sitemap indices for various `amazon` domains. It also shows the total number of URIs contained in these sitemaps.

Table 3. Sitemap Information about `amazon` Domains

Domain	#Sitemaps	#URIs
`amazon.com`	4'945	119'346'271
`amazon.ca`	2'980	96'476'534
`amazon.co.jp`	2'933	54'487'651
`amazon.co.uk`	3'177	44'668'202
`amazon.fr`	378	15'571'351

Amazon is a good example for the *Deep Web* motivation described in Section 1. Amazon has a large number of products available through its Web site, but most pages are dynamically generated and not statically linked from anywhere. Thus, to make all of these pages available to crawlers, all of these product pages must be listed in sitemaps.

6 Sitemaps Data Analysis

A somewhat surprising discovery is that some big sites do not have any sitemap information. `ebay` and `yahoo` are two examples. Contrast `ebay` to `amazon`, which has by far the largest number of page URIs in its sitemaps. Furthermore, many big sites are only marginally present: Our crawler discovered only 147 URIs for `microsoft`. The reason for this is that Microsoft specifies sitemaps for only a small fraction of its site.

To better understand the usage of sitemap files, it is interesting to look at how many sitemap files an average domain has, and what the distribution is of the number of sitemap files for those domains using sitemaps. Figure 7 shows this distribution. The horizontal axis shows the rank of a domain in terms of the number of sitemap files this domain uses. The vertical axis shows the number of sitemap files for that domain. Of the 5'303 domains included in that figure, the majority (3'880 or 73.2%) use just one sitemap file; but there is a heavy-tail distribution of domains using more than just one sitemap file. Furthermore,
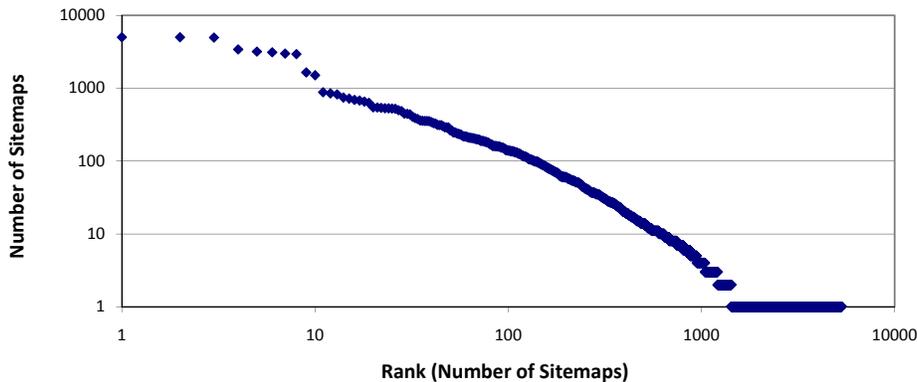
Fig. 7. Distribution of Sitemaps Across Domains

there is a small number of outliers which use an exceptionally high number of sitemap files.

Table 4. Top 10 Domains for #Sitemaps/Domain

Domain	#Sitemaps
1. pricecheck.co.za	5'006
2. ricardo.ch	5'000
3. amazon.com	4'945
4. mailonsunday.co.uk	3'395
5. amazon.co.uk	3'177
6. amazon.de	3'108
7. amazon.ca	2'980
8. amazon.co.jp	2'933
9. alacrastore.com	1'644
10. motofakty.pl	1'505

Table 4 shows the top ten domains in terms of number of sitemaps.[1] While amazon, ricardo (an auction site), and pricecheck are somewhat expected, somewhat surprising is the presence of the news site mailonsunday, which seems to have one sitemap file per calendar day. Each file lists the articles that were published on that day. This example contrasts the variance in sitemap organization: amazon uses a large number of sitemap files because of its sheer size; mailonsunday uses a large number of files in order to better organize its URIs in sitemaps.

To better understand how sitemap files are used on average, it is interesting to analyze the usage of sitemap files for managing large sets of URIs. Figure 8

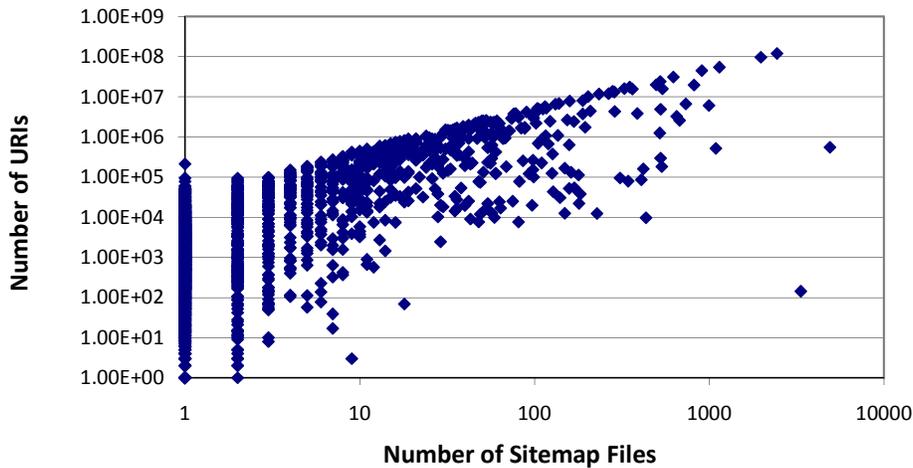[1] The top ten are the easily recognizable outliers visible in Figure 7.

Fig. 8. Number of URIs vs. Sitemap Files

plots the number of URIs in sitemaps versus the number of sitemap files used for storing these URIs. In theory, there should be no data point above the 50'000 URI mark on the 1 sitemap file line, because of the 50'000 URI per sitemap file limit specified by the sitemaps format.

There is much diversity in how sites beyond 100'000 URIs divide their URIs into sitemap files. For example, `pt.anuncioo.com` has a sitemap file with more than 200'000 URIs.[2] On the other extreme, `ricardo.ch` divides its 549'637 URIs into 4'911 files. Really large sites tend to use uniformly large (usually close to the maximum size of 50'000 URIs) sitemap files. Some of the outliers in the bottom right part of the figure are most likely caused by domains where we did have a substantial amount of sitemap files, but downloading the actual files (and then counting the URIs) failed due to timeouts.

7 Related Work

Regarding the analysis of `robots.txt` files, there is early work based on a rather small sample [8] (164 sites), and a specific analysis of corporate Web sites [9], also using a small sample (60 sites), and manual analysis of the results. This early work has been limited by much lower adoption of `robots.txt` files, and by the scale of the studies.

More recently, a study of initially only 8'000 sites [10,11] has been extended in the *BotSeer* project and now covers 13.2 million sites [12]. Their finding (in the initial 8'000 site study) of a 38.5% adoption rate of `robots.txt` files is a little bit smaller than our average of 45.1%, which might be explained by the

---

[2] Which is a violation of the sitemaps format that specifies a maximum of 50'000 URIs per file.

study's date (October 2006), and also by the fact that the study did not start with the most popular domains, which probably have a higher adoption rate. At the time of writing, the BotSeer Web page reports 2'264'820 `robots.txt` files from 13'257'110 Web sites, which translates to a 17% adoption rate; this considerably lower number may be explained by the fact that the large set of Web sites necessarily contains many rather small sites, which in many cases do not configure `robots.txt` files. In addition to crawling for `robots.txt` files, BotSeer is able to look at the dynamic behavior of crawler by setting up honeypot sites. These sites use `robots.txt` files and act as regular Web sites. BotSeer then logs how ethically crawlers act, i.e. how much of the restrictions defined in `robots.txt` they actually respect. This study of crawler behavior is something that is outside of our scope.

The *Web Modeling Language (WebML)* [13] is an approach to capture the structure of a Web site in a declarative way; it thus would be an ideal starting point for publishing information about site's structure (we do not know how far WebML provides support for this functionality, though). More generally, almost all *Content Management Systems (CMS)* have metadata about a site's content and structure and many support exposing this as `robots.txt` and/or sitemaps. As a popular example, the Drupal CMS supports a module for publishing sitemaps (initially named *Google Sitemap*, the module has been renamed to *XML Sitemap*).

We believe that once the users of richer Web site metadata are there (in the form of crawlers or browsers), it will be easily possible for many Web sites to automatically make that information available. A study by DANIELSON [14] has shown that a more structured overview of a Web site can help significantly in many tasks when interacting with a Web site; however, most approaches for Web site navigation only look at it as a per-site task, rather than looking at it as a fundamental way of how to interact with Web-based information.

To our knowledge, there is no related work in the overlap of the two areas described above, which is our eventual target area: The overlap of crawler-oriented site metadata often investigated in IR-oriented research, and the HCI-oriented question of how to make site metadata available to support navigational tasks on Web sites. Some prior work about looking at the Web graph in general [15] does discuss some questions relevant for our approach, though (specifically, the "URL split" technique presented in that paper). Surprisingly, even the otherwise detailed *Web Content Accessibility Guidelines (WCAG)* [16] say little about how to implement Web site navigation in an accessible may, they are mostly concerned with looking at individual Web pages.

8   Future Work

The work presented in this paper is the first stage of a research project that aims at making metadata about Web site structure available on the Web, as a service that can be provided by a site itself, or by a third party. We believe that

this metadata should be available so that it can be used by clients, for example to enhance Web site accessibility. Our approach [1] is twofold:

1. *Data Mining:* Based on the sitemap URIs, it is possible to construct a navigational sitemap of a Web site. We intend to employ approaches based on clustering of URI prefixes. This approach assumes that a site's URI structure reflects the site's navigational structure, and the important question is to find out how appropriate this assumption is, and whether it is possible to reliably detect whether the assumption is true for a given Web site or not.
2. *Data Format:* Based on the sitemaps format, we propose a format that can be used by Web sites to expose their navigational structure, if they want to do so. This site metadata can then be picked up by browsers and other clients, and typically will be more reliable than reverse-engineered data.

The next step beyond this is to set up an experimental service that provides access to data-mined navigational metadata, and to make that data available in a browser. A browser could use the two possible data sources mentioned above, first looking for authoritative navigational metadata provided by the site itself, and then accessing a third-party service inquiring about data-mined navigational metadata. This approach supports a transition strategy to a Web where sites can make their navigational metadata available, but if they don't do it, there still is a fallback provided by a third party.

9 Conclusions

This paper presents detailed analyses of the current availability of Web site metadata. The analyses are based on a starting set of the 100'000 most popular domains, and use data these sites make available through their `robots.txt` files and sitemaps. The analyses show that there is a wealth of Web site metadata available, even though currently its sole purpose is to control and steer Web crawlers. Based on these analyses, we conclude that it is a promising research path to take a closer look at the properties of the available Web site metadata, and our future work proposes to do that with the specific goal of extracting navigational metadata (i.e., metadata intended to improve navigational access to Web sites).

A more detailed presentation of the results can be found in a technical report [17], which is an extended version of the results presented here. We would like to thank Alexa for providing us with their dataset of the most popular 100'000 domains.

References

1. Wilde, E.: Site Metadata on the Web. In: Proceedings of the Second Workshop on Human-Computer Interaction and Information Retrieval, Redmond, Washington (October 2008)

2. Wilde, E., Gaedke, M.: Web Engineering Revisited. In: Proceedings of the 2008 British Computer Society (BCS) Conference on Visions of Computer Science, London, UK (September 2008)

3. Pant, G., Srinivasan, P., Menczer, F.: Crawling the Web. In Levene, M., Poulovassilis, A., eds.: Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Springer-Verlag, Berlin, Germany (November 2004) 153–178

4. Koster, M.: A Method for Web Robots Control. Internet Draft draft-koster-robots-00 (December 1996)

5. He, B., Patel, M., Zhang, Z., Chang, K.C.C.: Accessing the Deep Web. Communications of the ACM **50**(5) (May 2007) 94–101

6. Madhavan, J., Ko, D., Kot, L., Ganapathy, V., Rasmussen, A., Halevy, A.: Google's Deep Web Crawl. In: Proceedings of the 34th International Conference on Very Large Data Bases, Auckland, New Zealand, ACM Press (August 2008) 1241–1252

7. Nottingham, M., Sayre, R.: The Atom Syndication Format. Internet RFC 4287 (December 2005)

8. Cobéna, G., Abdessalem, T., Hinnach, Y.: WebWatching UK Web Communities: Final Report For The WebWatch Project. Technical Report British Library Research and Innovation Report 146, British Library Research and Innovation Centre (July 1999)

9. Drott, M.C.: Indexing Aids at Corporate Websites: The Use of Robots.txt and META Tags. Information Processing and Management **38**(2) (March 2002) 209–219

10. Sun, Y., Zhuang, Z., Councill, I.G., Giles, C.L.: Determining Bias to Search Engines from Robots.txt. In: Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence, Silicon Valley, California (November 2007) 149–155

11. Sun, Y., Zhuang, Z., Giles, C.L.: A Large-Scale Study of Robots.txt. In: Poster Proceedings of the 16th International World Wide Web Conference, Banff, Alberta, ACM Press (May 2007) 1123–1124

12. Sun, Y., Councill, I.G., Giles, C.L.: BotSeer: An Automated Information System for Analyzing Web Robots. In Schwabe, D., Curbera, F., Dantzig, P., eds.: Proceedings of the 8th International Conference on Web Engineering, Yorktown Heights, NY (July 2008)

13. Ceri, S., Fraternali, P., Matera, M.: Conceptual Modeling of Data-Intensive Web Applications. IEEE Internet Computing **6**(4) (2002) 20–30

14. Danielson, D.R.: Web Navigation and the Behavioral Effects of Constantly Visible Site Maps. Interacting with Computers **14**(5) (October 2002) 601–618

15. Raghavan, S., Garcia-Molina, H.: Representing Web Graphs. In Dayal, U., Ramamritham, K., Vijayaraman, T.M., eds.: Proceedings of the 19th International Conference on Data Engineering, Bangalore, India, IEEE Computer Society Press (March 2003) 405–416

16. Caldwell, B., Cooper, M., Reid, L.G., Vanderheiden, G.: Web Content Accessibility Guidelines 2.0. World Wide Web Consortium, Recommendation REC-WCAG20-20081211 (December 2008)

17. Wilde, E., Roy, A.: Web Site Metadata. Technical Report UCB ISchool Report 2009-028, School of Information, UC Berkeley, Berkeley, California (February 2009)