Web Site Metadata

Erik Wilde and Anuradha Roy (School of Information, UC Berkeley)

UCB ISchool Report 2009-028
February 2009

Available at `http://dret.net/netdret/publications#wil09b`

Abstract

The currently established formats for how a Web site can publish metadata about a site's pages, the `robots.txt` file and sitemaps, focus on how to provide information to crawlers about where to not go and where to go on a site. This is sufficient as input for crawlers, but does not allow Web sites to publish richer metadata about their site's structure, such as the navigational structure. This paper looks at the availability of Web site metadata on today's Web in terms of available information resources and quantitative aspects of their contents. Such an analysis of the available Web site metadata not only makes it easier to understand what data is available today; it also serves as the foundation for investigating what kind of information retrieval processes could be driven by that data, and what additional data could be provided by Web sites if they had richer data formats to publish metadata.

Contents

1 Introduction

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 domain (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). Web sites are usually accessed by *Web crawlers* [9] 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 [7] — 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 (most often by limiting) crawler access to the site. This assumes that crawlers get information about available URIs from other source; 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[1] 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.

Sitemap information can be useful for exposing the *Deep Web* [6, 8], 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 thus never show up in search results. Thus, 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.

Large search engines crawl different parts of the web with different frequencies. For example, news sites will likely be crawled (and indexed) much more frequently than sites that change infrequently. Sitemap information (in particular, information within `lastmod` tags) can be used by crawlers to set their crawl schedules. How much of this information is used by current search engines is unknown.

With only these two formats, Web sites currently have only a limited way of publishing their site's structure in a machine-readable way, and these ways are mainly intended to steer crawlers. This paper describes the Web site metadata that we found when we were crawling for `robots.txt` files and sitemaps. Beyond that, we envision to provide Web sites with a better way to expose their navigation structure, so that it can be used for improving a site's usability and/or accessibility. The work presented in this paper is the first set of results from a project [14] that attempts to derive information about a Web site's navigational structure from available metadata, and introduces a data format for Web sites (as an extension of the sitemaps format) to make that data available explicitly, if they wish to do so.

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

[1] `http://www.sitemaps.org/`

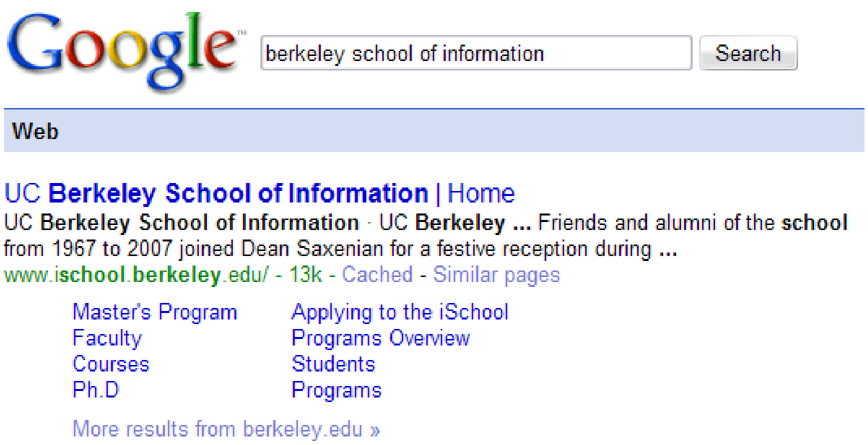

Figure 1: Algorithmically Computed Site Map

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.

This paper first presents the starting dataset for the domains to be crawled (Section 2) and then 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 Domain Statistics

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. Before we describe our use of the domain name dataset, here are some basic statistics about it. The distribution of *top-level domains (TLDs)* in that dataset is shown in Figure 2.

Based on this dataset, our crawling process requests `robots.txt` files from all domains (described in Section 3).

3 Crawling for Robots.txt

Starting from Alexa's dataset of the most popular 100'000 domains, we collect `robots.txt` files. The strategy for that is a simply two-step process. First, the domain name itself is used to request the file, for example sending a request for `http://domain.com/robots.txt`. In most cases, if there is a `robots.txt`, it is either served directly, or there is an HTTP redirect to the `www`-prefixed URI, such as `http://www.domain.com/robots.txt`. However, some servers are setup to redirect *any* requests for `http://domain.com/*` to `http://www.domain.com/`, in which case the request for `robots.txt` is redirected to the site's home page. Hence, we check the result of requesting `http://domain.com/robots.txt` with simple heuristics whether it is HTML, and if it is, we request `http://www.domain.com/robots.txt`.

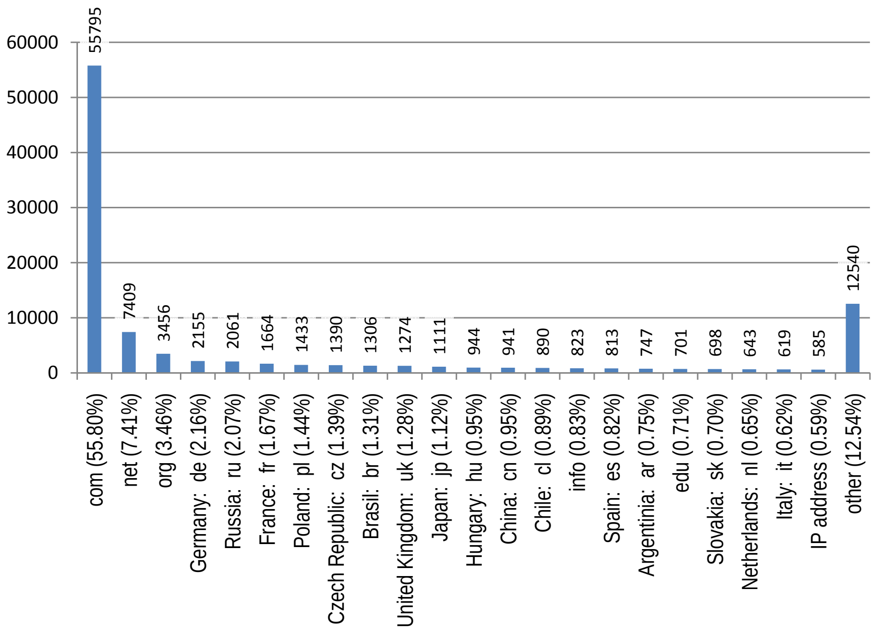

Figure 2: Distribution of TLDs in Popular Domains

Based on this simple two-step process, our crawl of 100'000 domains for `robots.txt` files yields 44'832 files; more detailed statistics about these files can be found in Section 4. Various error conditions can be encountered when requesting the files. Here is a collection of the most frequent error conditions when requesting or processing `robots.txt` files:

- *Server Errors:* Some servers encounter internal misconfigurations and return server errors (we received `500`, `503`, and `504` responses). Some have badly configured redirects, in which case the redirect points to unintended resources (such as `http://www.domain.comrobots.txt`, omitting the slash). In all of these cases it is impossible to retrieve a `robots.txt` file.
- *Media Type Errors:* Some servers respond with a `text/html` entity to the request. In most cases, they still serve a plain text file, but it is mislabeled as a different media type.
- *Charset Issues:* If the server does respond with a `robots.txt` file, there can be character set issues. These may manifest themselves as references to non-existing character sets (for example, `windwos-1251`, `win-1251`,[2] and `Latin-1`[3]), character sets not supported on the client side, and `robots.txt` files containing errors with respect to the signaled character set.
- *Size:* The biggest `robots.txt` file is 12.3MB. While this is not a very big file, it is quite a bit larger than anticipated (see Figure 4 for a distribution of files sizes).[4]
- *Connection Problems:* A number of servers did not properly close the connection, so that connections would remain open for a long time. Another problem were dropped connections. However, in the vast majority of cases, connections were handled properly by the crawled servers.

[2] These two should be `windows-1251`.

[3] This should be `latin1` or `ISO-8859-1` (preferred).

[4] Most bigger files, though, are either HTML (caused by improper redirects) or text-based sitemaps, which are (usually long) line-by-line lists of a site's URIs.

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.

One interesting case is a 380'000 lines, 12.3MB `robots.txt` file from `cityweekend.com.cn`. Apparently, this `robots.txt` file lists many user accounts on that site, and specifically sets `Disallow` rules for them. Strictly speaking, this might be necessary if crawlers are to be controlled in specific subsets of user-specific pages, because `robots.txt` rules do not support *patterns* to be specified for URIs, they only support *prefixes*. On the other hand, it seems unlikely that crawlers will interpret files of that size, and there also is the issue of potentially unintended disclosure of account data (discussed in more detail in Section 5.1).

The 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.[5] 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.

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.

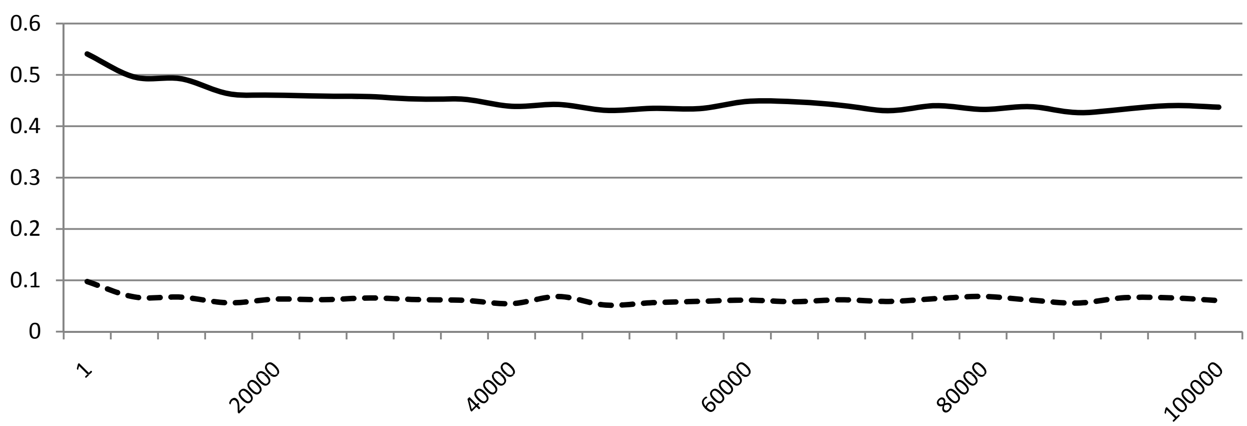

Figure 3: Distribution of Site Metadata

Figure 3 shows the distribution of site metadata in terms of domain ranking. It shows the likelihood of a `robots.txt` file and of sitemap information being available depending on the popularity of the domain (the overall averages are 45.1% for `robots.txt` files and 6.3% for sitemap information). The solid line shows the likelihood of a `robots.txt` file being available, and the dashed line shows the likelihood of sitemap information being available (because in our setup sitemaps are only discovered through `robots.txt` files, the second line can never be higher than the first line, and for our data always is considerably lower than the

[5] `Noindex` has been introduced by Google and `Crawl-Delay` has been introduced by Microsoft.

first). It can be seen that there is a slight correlation between domain popularity and metadata availability (but mostly so for the most popular domains), which makes sense, because more popular sites probably invest more effort in being as visible and as accessible as possible.

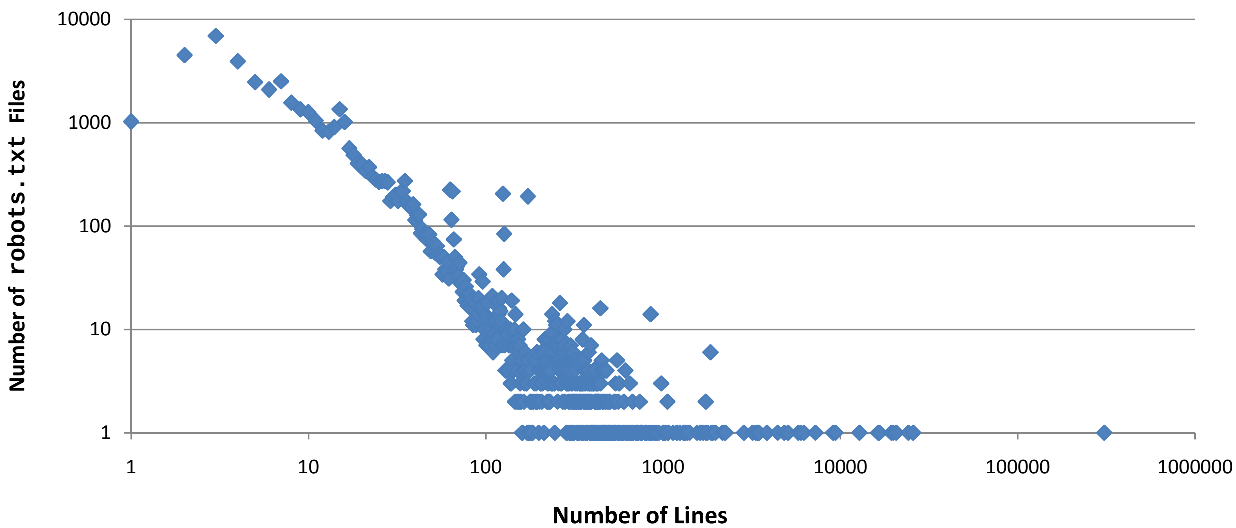

Figure 4: Distribution of `robots.txt` Size

Figure 4 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 5 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 3, 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 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.

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

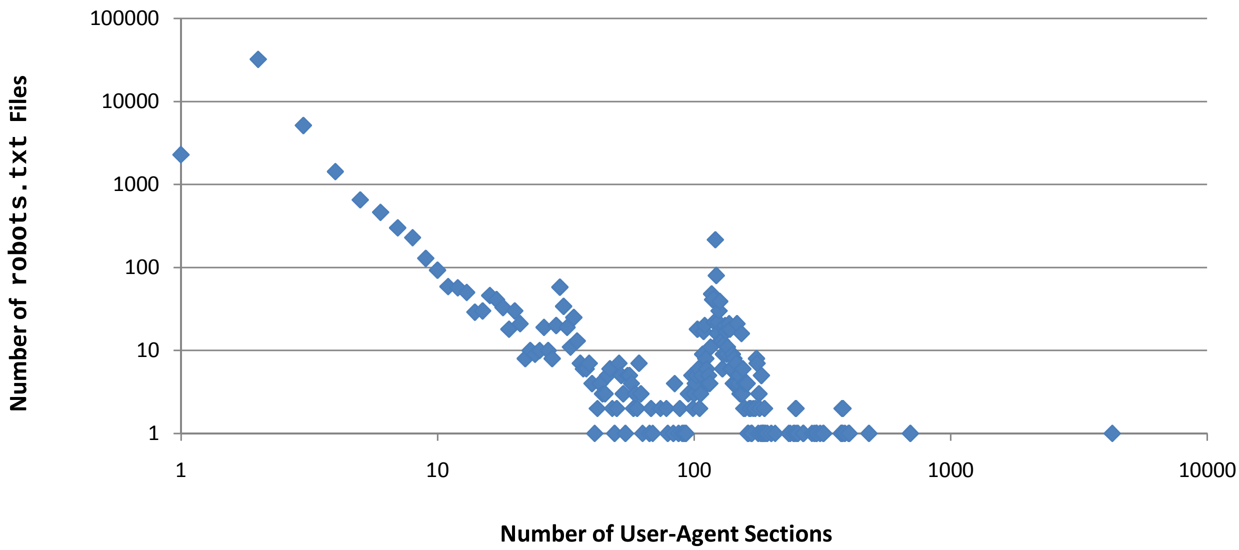

Figure 5: `User-Agent` Sections per `robots.txt` File

	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

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

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 statistics shown in Table 1 are influenced by the fact that typically, for one `User-Agent`, there are a number of `Disallow` rules specifying the URI prefixes that should not be crawled by that particular crawler. To better understand the complexity of these rules, and how much of a site's structure they expose in terms of specifying relevant URI spaces, we looked at the size of `User-Agent` sections, meaning those sections of a `robots.txt` files which are specifying `Disallow` (and maybe other) rules for one specific `User-Agent`. These sections are limited by `User-Agent` fields, or by the end of the `robots.txt` file. Figure 6 shows the results of this analysis.

The total number of `User-Agent` sections we found is 202'332. 33'589 `User-Agent` sections (16.6%) had no `Disallow` rules at all (not shown in Figure 6 because of the logarithmic axis). This may be due to

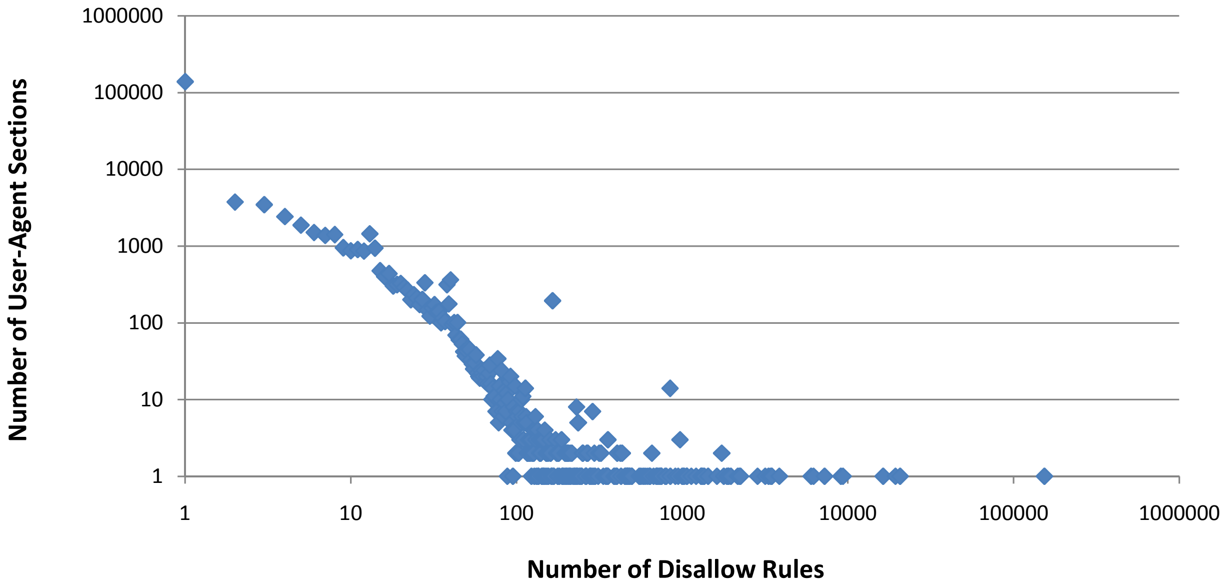

Figure 6: `Disallow` Rules per `User-Agent` Section

the fact that they were actually empty, or contained rules other than `Disallow`, such as the ACAP rules mentioned above. Apart from these `User-Agent` sections containing no `Disallow` rules, the distribution shows a heavy-tail pattern, with the number of `User-Agent` sections containing one `Disallow` rule being an outlier (138'695 or 68.5%). There also is one outlier in the other direction with one `User-Agent` section containing 153'540 `Disallow` rules.

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 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 7). `*` 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`

	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: Popular User-Agents in `robots.txt` Files

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.

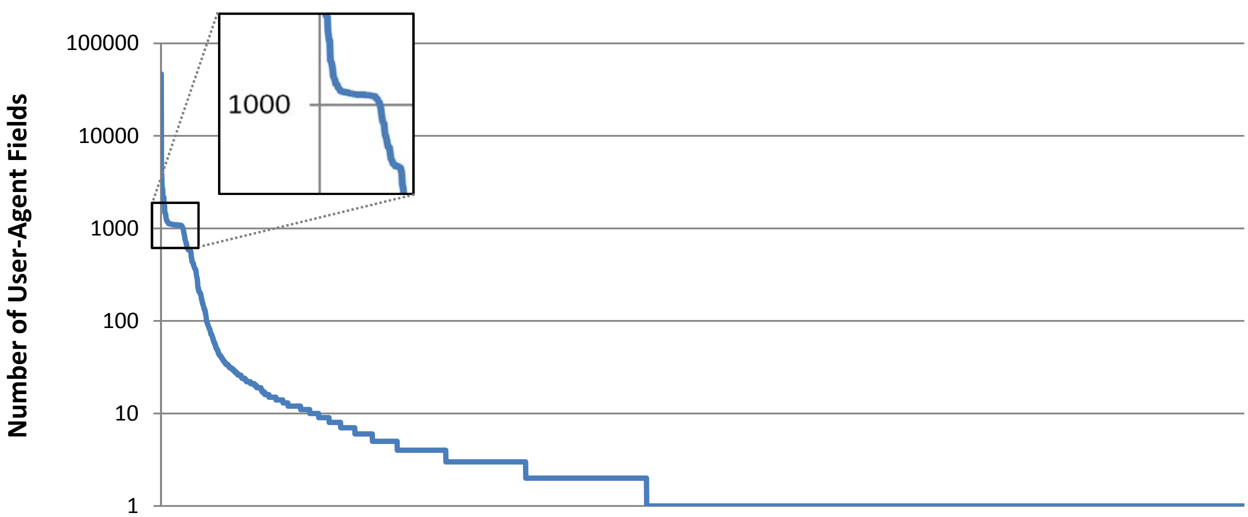

Figure 7: Distribution of `User-Agent` Field Values

Figure 7 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.

The fact that many crawlers are only mentioned once or a small number of times can be explained by the fact that a particular crawler might have had problems with the setup of a certain Web site, which then blocked access to that crawler by adding it to its `robots.txt` file. However, this depends on the assumption that this would be a badly behaving crawler which is just poorly implemented, putting undue strain on a server, but still well-behaving in the sense that it respects `robots.txt`. In addition, there are badly behaving crawlers which do not even respect `robots.txt`, in which case they need to be blocked at the HTTP or TCP level, implementing blocking based on HTTP `User-Agent` headers or even IP addresses. Limitations sites enforce for this latter kind of crawler do not show up in `robots.txt` files.

These considerations about crawler behavior and how they might be affected by `robots.txt` files are based on our static `robots.txt` analysis. Section 7 discusses a more dynamic approach chosen by the *BotSeer* system, which observes dynamic behavior of crawlers by setting up honeypots.

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. As shown in Figure 3, the likelihood of a Web site providing sitemaps is much 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.

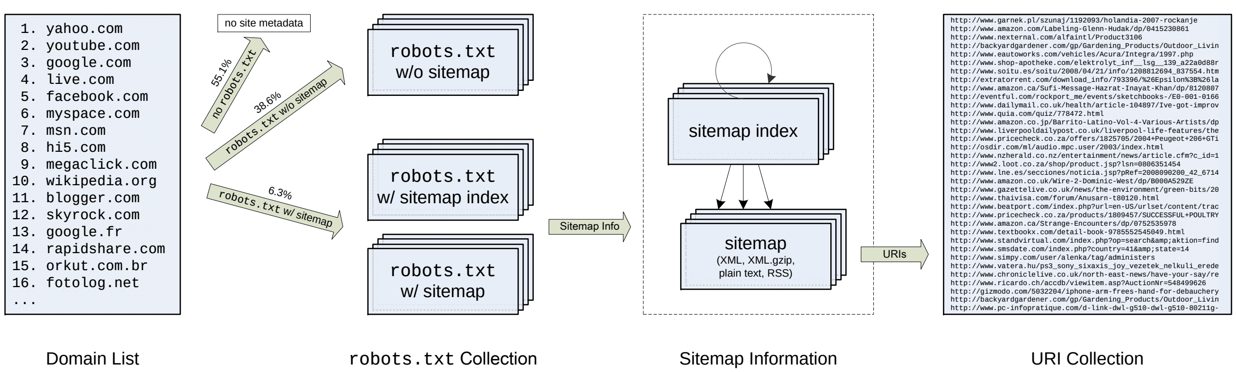

Figure 8: Overview of the Crawling Process

Figure 8 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).

Figure 9 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 sitemap format specifies two kinds of files: *Index files* and *Sitemaps*. Index files do not contain Web page URIs, they simply point to other sitemap files (they may point to index files, allowing hierarchies of index files); they always use an XML syntax. Table 3 shows the levels of indirections found when crawling for

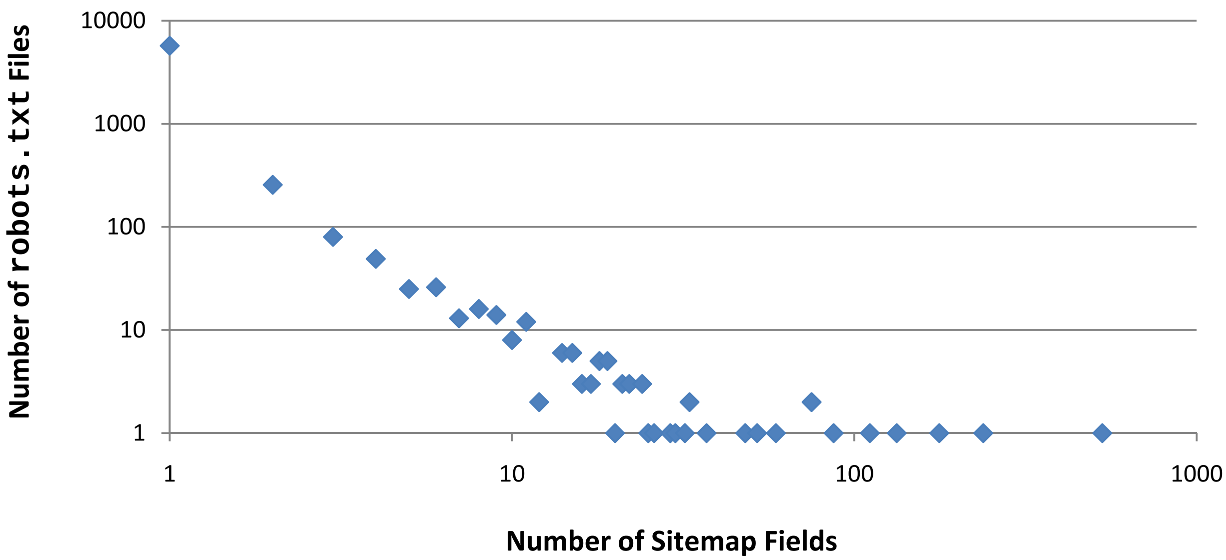

Figure 9: `Sitemap` Fields per `robots.txt` File

Level	0	1	2	3	4
#Files	9'081	90'512	43'044	533	510

Table 3: Indirection Level of Sitemap Information

sitemap files, where the indirection level indicates how many index files had to be traversed to the ultimate sitemap file containing a site's page URIs.

Sitemaps can use XML, plain text, or a feed format (RSS 2.0 or Atom) as their syntax. Both kinds of sitemap files may be gzip-compressed. 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.

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. Here is a collection of the most frequent error conditions when requesting or processing sitemap files:

- *Sitemap Variants:* The sitemap format does not explicitly support different variants of a sitemap to be made available. Some sites contain links to `.xml` and `.xml.gz` files with the same content, which results in duplicates and the necessity to deal with these.
- *Syntax Issues:* A number of sites (slightly more than 1%) assumed that the URI for the sitemaps file(s) in the `robots.txt` file should be enclosed in angle brackets, probably because the format description for the sitemaps file format does show angle brackets and is not sufficiently explicit that these are not part of the actual syntax.[6] For our sitemaps data crawl, we removed the angle brackets and treated those URIs as if they had been correctly specified.

[6] The syntax to be used is described as "`Sitemap: <sitemap_location>`" in the definition of how to specify a sitemap location in a `robots.txt` file.

- *Attempted Sitemap Sharing:* Many domains, especially those belonging to the same entity (for example, `google.com` and `google.co.vn`) attempt to share sitemap files. The sitemaps formats does not allow this kind of cross-site references, so it is up to the discretion of the crawler to ignore or use these shared sitemaps.
- *Connection Problems:* A number of servers did not properly close the connection, so that connections would remain open for a long time. Another problem were dropped connections. However, in the vast majority of cases, connections were handled properly by the crawled servers.

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 4 shows the number of sitemaps/sitemap indices for various `amazon` domains. It also shows the total number of URIs contained in these sitemaps.

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.de`	3'108	226[7]

Table 4: Sitemap Information about `amazon` Domains

Amazon is a good example for the *Deep Web* motivation described in Section 1. Amazon has a large number of products available through it's 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.

5.1 Access Controlled Sitemaps

Starting from the 10'029 references to sitemap files found in `robots.txt` files, the crawling process produced 70'984 successful file downloads of 60'321 distinct files; there were also a number of errors, summarized in Figure 10. Most common errors were timeout errors, which is expected because of our strict 30s timeout policy. There were also a significant number of `404 (Unavailable)` errors.

`robots.txt` files are usually openly available, and in principle, the same can be said about sitemaps. However, it seems that some of them are not only unavailable (as signaled by a `404` error), but access controlled. For 111 sitemap files, our crawler received `403 (Forbidden)` responses. One interesting question is why sites might access control sitemap files, because we ran into various combinations of HTTP status codes and redirects which indicated that access control might be in place.

Sitemaps might expose more of a site's structure than a site would like to make public, for example a complete set of accounts if a social networking site exposes all accounts as URIs. One the one hand, the site might be interested to make these pages available to search engines; on the other hand, it might want to make harvesting of all the accounts a little less easy than just reading sitemaps. In this case, access control could be based on implicit authentication such as through well-know IP addresses of authorized crawlers. We don't think that access controlled sitemaps will use HTTP authentication, and we received only very few HTTP-level authentication responses in the form of `401 (Unauthorized)`.

[7] Many download errors were encountered.

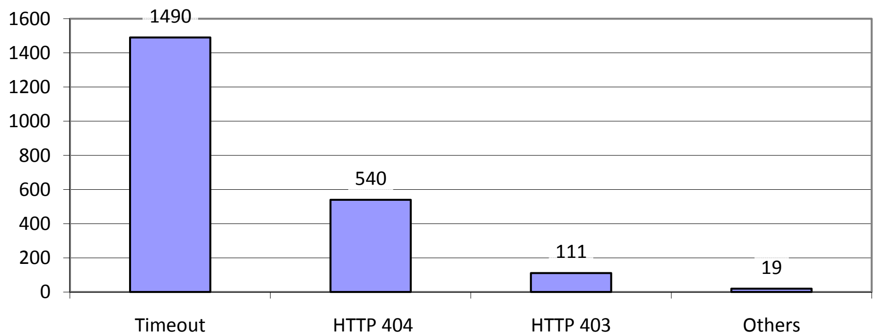

Figure 10: Sitemap Download Errors Encountered

We believe that for brick and mortar businesses, there is probably little incentive to access control sitemap information, because the Web site is only providing representations for goods or services that extend beyond the Web. On the other hand, typical Web 2.0 businesses often do not provide any physical goods or services, so for them, the Web representations often are close to the essence of what they are and do. For these businesses, exposing this information in a machine-readable way is a more critical decision, and therefore they might make the decision to only disclose it to trusted clients, such as crawlers of major search engines.

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.

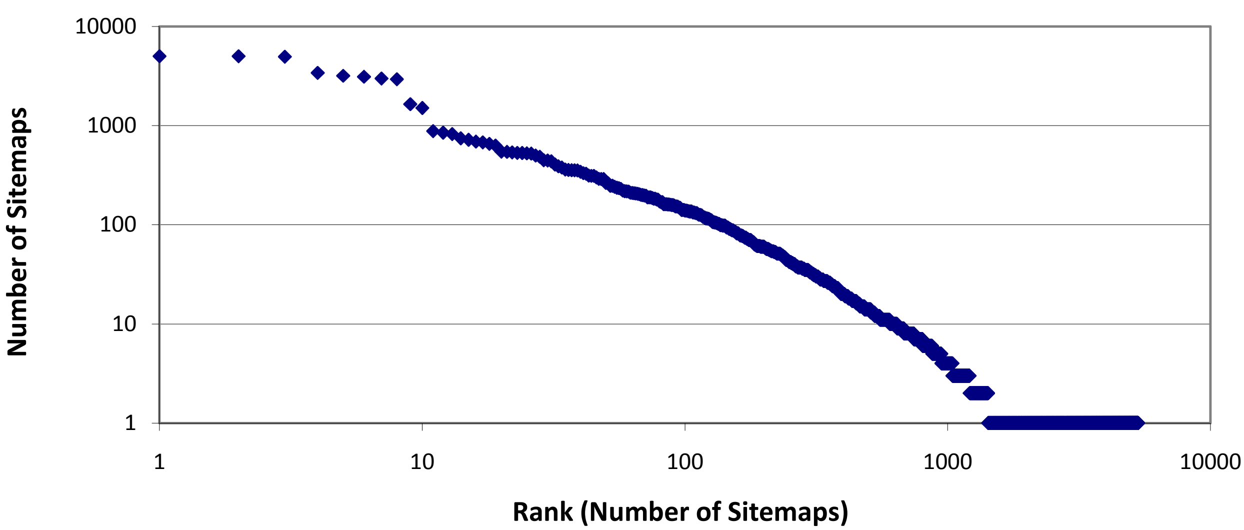

Figure 11: Distribution of Sitemaps Across Domains

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

	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 5: Top 10 Domains for #Sitemaps/Domain

Table 5 shows the top ten domains in terms of number of sitemaps.[8] 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. We discuss the distribution of URIs per sitemap in detail below.

Continuing from the question of sitemap files per domain (as shown in Figure 11), the next question then is how many URIs are eventually listed in these files? Figure 12 shows the distribution of domains based on how many URIs are specified for them in sitemap files (vertical axis), and how many domains with this many URIs exist. The horizontal axis then ranks the domains according to the number of URIs for them. As shown in Table 4, `amazon.com` is the highest ranked domain listing 119'346'271 URIs in its sitemap files (in fact, the top three ranked domains in this figure are the top three domains from Table 4). Another way to look at the same dataset and distribution would be to ask how much of the entire set of URIs contained in sitemaps (836'260'857 URIs) is covered by which share of sites publishing large sitemaps.

Figure 13 shows this coverage of the complete URI dataset by ranking how much domains contribute to covering that dataset. It starts with `amazon.com`'s 119'346'271 URIs, and continues to grow logarithmically until it gets to the point where domains only specify increasingly smaller URI sets and thus contribute almost nothing to the overall coverage. Since the plot is a straight line, it shows that the distribution of URIs across domains is neither exponential $y = e^{\lambda x}$ (in which case, this distribution would have been skewed with more weight toward the origin) nor a power law $y = x^{-\lambda}$ (in which case, this distribution would have been skewed away from the origin).

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 14 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

[8]The top ten are the easily recognizable outliers visible in Figure 11.

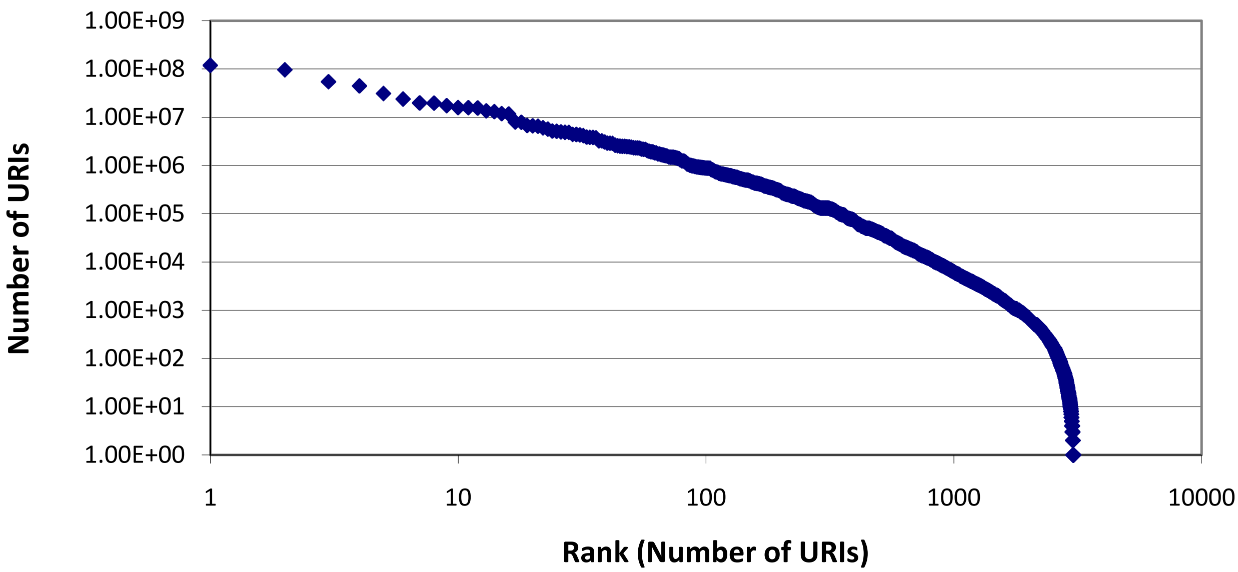

Figure 12: Distribution of URIs Across Domains

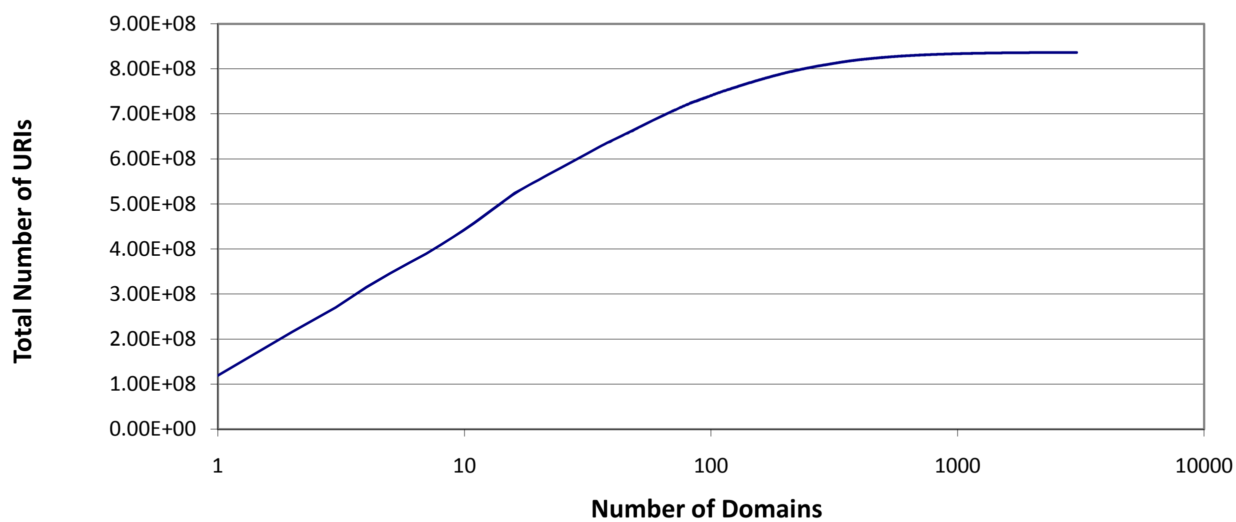

Figure 13: Cumulative View of Figure 12

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.[9] 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

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

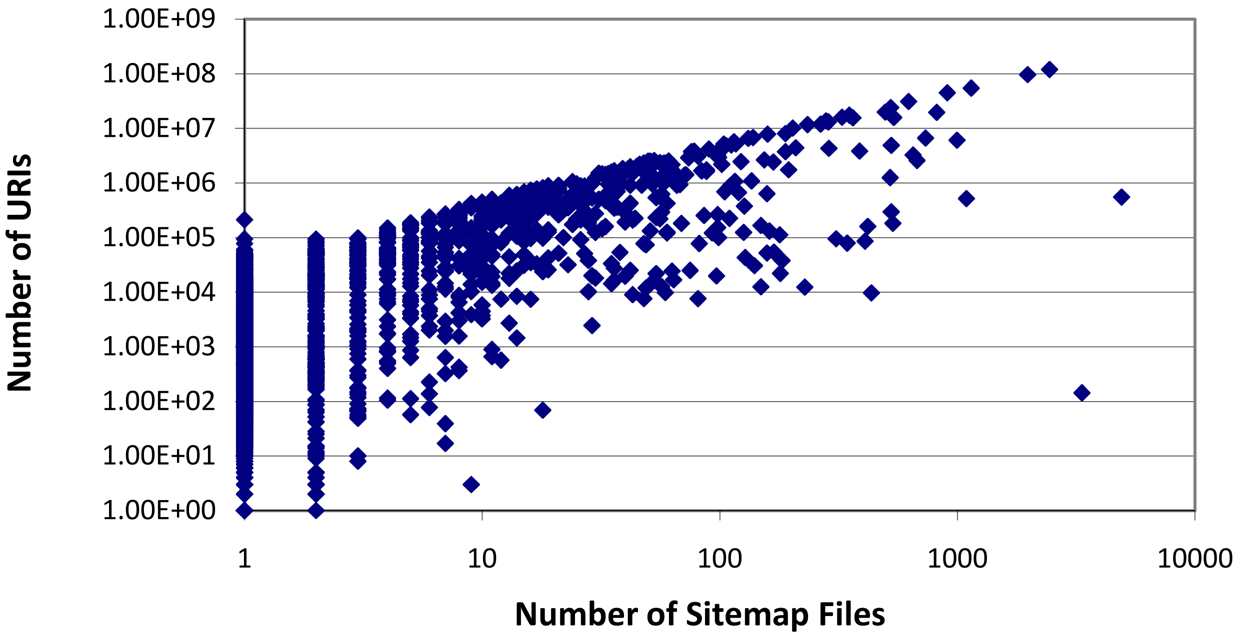

Figure 14: Number of URIs vs. Sitemap Files

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.

As the final analysis of the sitemap data, Figure 15 shows the distribution of the average number of URIs per sitemap file. Since the vertical axis is logarithmic and the horizontal axis is linear, it is clear that this distribution is mostly exponential, except at the tail, where it is super-exponential. Each data point in that figure is per domain, which explain data points with less than one URI per sitemap file; these are domains that use index files, but the count of actual page URIs was very low (which may be due to the sitemap access problems mentioned earlier).

7 Related Work

Regarding the analysis of `robots.txt` files, there is early work based on a rather small sample [3] (164 sites), and a specific analysis of corporate Web sites [5], 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 [12, 13] has been extended in the *BotSeer* project and now covers 13.2 million sites [11]. 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 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

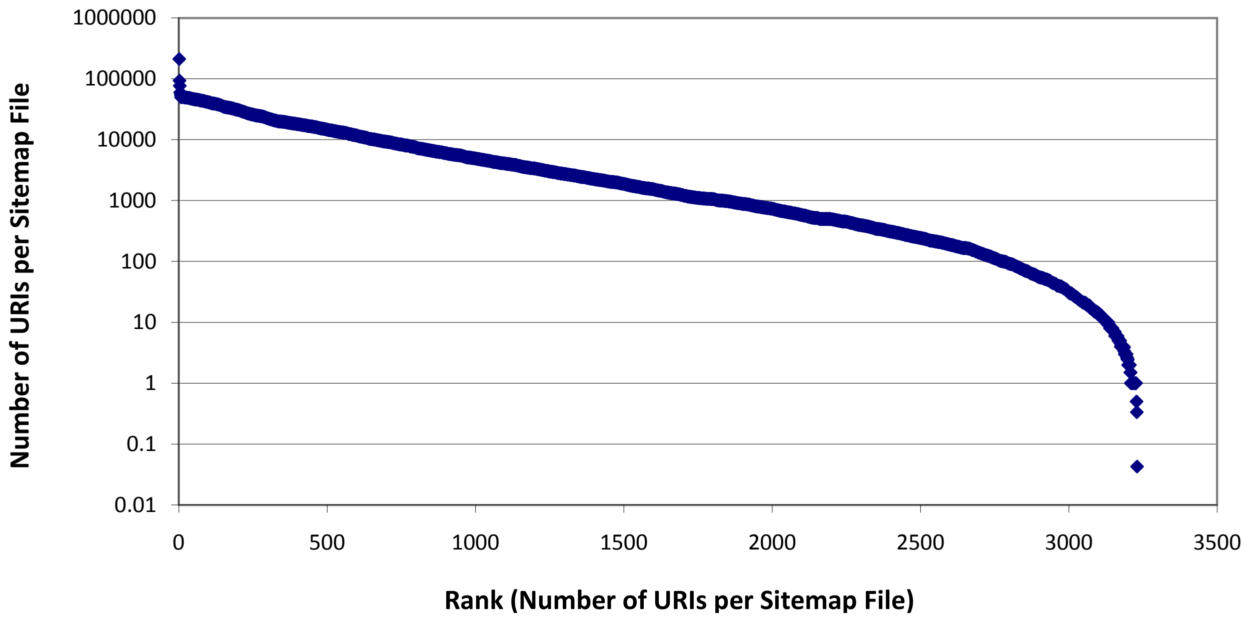

Figure 15: Average URIs per Sitemap File

`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)* [2] 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 [4] 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 [10] 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)* [1] 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. We believe that this metadata should be available so that it can be used by clients. Our approach [14] 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 Acknowledgements

We would like to thank Alexa for providing us with their dataset of the most popular 100'000 domains.

10 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).

References

[1] Ben Caldwell, Michael Cooper, Loretta Guarino Reid, and Gregg Vanderheiden. Web Content Accessibility Guidelines 2.0. World Wide Web Consortium, Recommendation REC-WCAG20-20081211, December 2008.

[2] Stefano Ceri, Piero Fraternali, and Maristella Matera. Conceptual Modeling of Data-Intensive Web Applications. *IEEE Internet Computing*, 6(4):20–30, 2002.

[3] Grégory Cobéna, Talel Abdessalem, and Yassine Hinnach. 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.

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

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

[6] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the Deep Web. *Communications of the ACM*, 50(5):94–101, May 2007.

[7] Martijn Koster. A Method for Web Robots Control. Internet Draft draft-koster-robots-00, December 1996.

[8] Jayant Madhavan, David Ko, Łucja Kot, Vignesh Ganapathy, Alex Rasmussen, and Alon Halevy. Google's Deep Web Crawl. In *Proceedings of the 34th International Conference on Very Large Data Bases*, pages 1241–1252, Auckland, New Zealand, August 2008. ACM Press.

[9] Gautam Pant, Padmini Srinivasan, and Filippo Menczer. Crawling the Web. In Mark Levene and Alexandra Poulovassilis, editors, *Web Dynamics: Adapting to Change in Content, Size, Topology and Use*, pages 153–178. Springer-Verlag, Berlin, Germany, November 2004.

[10] Sriram Raghavan and Hector Garcia-Molina. Representing Web Graphs. In Umeshwar Dayal, Krithi Ramamritham, and T. M. Vijayaraman, editors, *Proceedings of the 19th International Conference on Data Engineering*, pages 405–416, Bangalore, India, March 2003. IEEE Computer Society Press.

[11] Yang Sun, Isaac G. Councill, and C. Lee Giles. BotSeer: An Automated Information System for Analyzing Web Robots. In Daniel Schwabe, Francisco Curbera, and Paul Dantzig, editors, *Proceedings of the 8th International Conference on Web Engineering*, Yorktown Heights, NY, July 2008.

[12] Yang Sun, Ziming Zhuang, Isaac G. Councill, and C. Lee Giles. Determining Bias to Search Engines from Robots.txt. In *Proceedings of the 2007 IEEE/WIC/ACM International Conference on Web Intelligence*, pages 149–155, Silicon Valley, California, November 2007.

[13] Yang Sun, Ziming Zhuang, and C. Lee Giles. A Large-Scale Study of Robots.txt. In *Poster Proceedings of the 16th International World Wide Web Conference*, pages 1123–1124, Banff, Alberta, May 2007. ACM Press.

[14] Erik Wilde. Site Metadata on the Web. In *Proceedings of the Second Workshop on Human-Computer Interaction and Information Retrieval*, Redmond, Washington, October 2008.