Sitemaps: Above and Beyond the Crawl of Duty

Uri Schonfeld
UCLA Computer Science Department
4732 Boelter Hall
Los Angeles , CA 90095
shuri@shuri.org

Narayanan Shivakumar
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA
shiva@google.com

ABSTRACT

Comprehensive coverage of the public web is crucial to web search engines. Search engines use crawlers to retrieve pages and then discover new ones by extracting the pages' outgoing links. However, the set of pages reachable from the publicly linked web is estimated to be significantly smaller than the invisible web [5], the set of documents that have no incoming links and can only be retrieved through web applications and web forms. The Sitemaps protocol is a fast-growing web protocol supported jointly by major search engines to help content creators and search engines unlock this hidden data by making it available to search engines. In this paper, we perform a detailed study of how "classic" discovery crawling compares with Sitemaps, in key measures such as coverage and freshness over key representative websites as well as over billions of URLs seen at Google. We observe that Sitemaps and discovery crawling complement each other very well, and offer different tradeoffs.

Categories and Subject Descriptors: H.3.3: Information Search and Retrieval.

General Terms: Experimentation, Algorithms.

Keywords: search engines, crawling, sitemaps, metrics, quality.

1. INTRODUCTION

How can search engines keep up with the application rich, constantly changing, trillion URL scale web [17]? Search engines utilize massive computing and networking resources to run Web crawlers in order to build and maintain a frequently updated, quality snapshot of the web. However, even with such massive resources, crawlers still face huge challenges in this task. In this paper we investigate the Sitemaps protocol, how it is being used, and how it can be harnessed to better keep up with the web.

A Web crawlers starts by fetching a "seed" set of popular URLs such as aol.com and yahoo.com that link to many Web pages. It then proceeds by extracting their outgoing links, adding them to a list of known URLs, and finally selecting a set of URLs to retrieve next. The crawler repeats the above steps until it detects it has a sufficiently good set of Web pages to index and serve. The crawler then continues to recrawl some of these pages at different frequencies in order to maintain the freshness of the document collection [1].

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.
WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

Crawlers still face at least two prominent challenges:

- *Coverage*: One of the problems crawlers face is exposing the content "trapped" inside databases, behind forms and behind other browser technologies. For example, the growing use of rich AJAX applications where pages are dynamically constructed based on user actions, make link extraction more difficult. New advances in plug-in technologies (e.g., Google Gears, Silverlight) that create custom interactions between the website and the client browser make the problem even worse. In general, data that is only available through web forms, flash applications, or downloaded javascript programs together constitute the *deep web* [5]. How can we help direct crawlers towards picking up these islands of content?
- *Freshness*: Users today use search engines to search for world events, viral videos or a new product announcement minutes after the event transpired (e.g. http://www.google.com/trends). Considerable research has gone into modeling and predicting the rate at which the content of a website changes. [11]. But it is increasingly challengingly expensive for even sophisticated crawlers to keep a pulse on the rapidly changing and growing web [17]. How can we help crawlers discover all new and modified valid URLs of a site in a timely manner?

Sitemaps are an easy way for webmasters to inform search engines about pages on their sites that are available for crawling. In its simplest form, a Sitemaps file is an XML file that lists a site's URLs, along with additional metadata detailing: when was the page last updated, how frequently does the page change on average and how important it is relative to the other pages in the site. The purpose of Sitemaps is to enable search engines to crawl the site more intelligently. Using the Sitemaps protocol does not guarantee that web pages will be included in the search engine's index, but it does provide them with hints that help Web crawlers do a better job of crawling the site.

The value of such protocols lays primarily in their widespread adoption by key web-players – search engines, web sites and web site tools. In 2006, Google, Yahoo and Microsoft co-announced support for the Sitemaps protocol in an industry first [15]. Since then, billions of URLs have been published on the web via Sitemaps and millions of websites support Sitemaps. These include the US federal government, the US state governments [28], and many popular websites such as amazon.com, hp.com, cnn.com, wikipedia.org. A variety of website tools providers including Go Daddy, Yahoo! Small

Business, Google Search Appliance, IBM Portal, as well as open-source projects like Plone CMS, Cold Fusion all support auto-generating Sitemaps.

To the best of our knowledge this paper provides the first analysis of a large data set collected from the Sitemaps protocol. Our main contributions are: (a) We offer a case study of how three different websites with different characteristics organize and update Sitemaps; (b) We introduce metrics to evaluate the quality of the Sitemaps provided by a website; (c) We study how the web appears through Sitemaps crawl's and Discovery crawl's perspectives in terms of freshness and comprehensiveness; (d) We examine how Web crawlers can use Sitemaps and we introduce crawling and refreshing algorithms that make use of Sitemaps data.

The rest of this paper is structured as follows. Section 2 covers related work. Section 3 gives a detailed overview of the Sitemaps protocol, high level statistics on how it is being used by users, and how it fits in Google's crawling architecture. Section 4 details a case study of three prominent sites having very different characteristics. Section 5 introduces metrics to evaluate the quality of Sitemaps data and uses these metrics to evaluate a large data set of real world web data. Finally Section 6 examines different ways in which Sitemaps data can be incorporated inside Web crawlers.

2. RELATED WORK

Brandman et al [7] discussed how to make web servers more friendly to crawlers, including the idea of exporting a list of URLs along with additional metadata to offer an efficient communication mechanism between the crawler and the web servers. In 2006, Google, Yahoo and Microsoft announced support for Sitemaps as a common standard for websites based on a simple list of URLs [15]. A good overview of the protocol itself was described in [27] by Ranjan et. al.

Other XML-based protocols similar to Sitemaps include RSS, Atom, OAI-PMH. RSS and Atom are popular XML-based web protocols used by websites and blogs typically used to give recent updates and surfaced in RSS readers such as google.com/reader. The Sitemaps protocol was designed to support large sites and is flexible in supporting differential updates. [18]. OAI is an older sophisticated protocol used by digital libraries [20]. Google supported each of the above protocols in submissions. However as we see later, the fraction of URLs coming in through RSS/Atom were smaller and virtually none come from OAI [16]. We believe RSS/Atom will continue to be popular for consumer-friendly updates and Sitemaps will become increasingly popular as a simple and crawler-friendly update mechanism. OAI will continue to be popular in the library community as a mechanism to submit rich metadata.

The problem of exposing the deep web has been studied extensively. Recently, in [21] Madhavan et al. discuss the problem of increasing the visibility of web sites that have content hidden behind forms and inside data bases. They introduced techniques to automatically fill web forms in order to expose some of the URLs hidden inside the deep web of the site. While these techniques are complementary to Sitemaps, such techniques do not utilize a web servers knowledge, nor offer provable guarantees on coverage, freshness or efficiency.

Extensive research has been done on the problem of approximating the rate of change of web content [10, 11, 9, 26], and developing crawling algorithms that improve the freshness of documents. In [24] it was shown that approximating the rate of change is not always possible. On the other hand, if the rate of change is known, optimal crawling schedules are known for various metrics [9, 29]. The Sitemaps protocol uses the web server's knowledge to offer an alternative to approximating change rates and thus is complementary to the optimal scheduling techniques that are based on change rates. In this paper, we examine the quality of the data supplied through the Sitemaps protocol, and how it differs from that available through Discovery crawl.

Much has gone into examining the web through crawl and through random walks [10, 13, 3]. Our paper offers an alternative view of the web through Sitemaps data.

3. SITEMAPS PROTOCOL

Sitemaps files are XML files with a list of URLs with additional metadata, as shown in the example below.

```
<?xml version="1.0" encoding="UTF-8"?>
<urlset xmlns=
  "http://www.sitemaps.org/schemas/sitemap/0.9">
   <url>
      <loc>http://www.example.com/</loc>
      <lastmod>2005-01-01</lastmod>
      <changefreq>monthly</changefreq>
      <priority>0.8</priority>
   </url>
   <url>
       ...
   <url>
</urlset>
```

Within each <*url*> record:

- *loc* is a required field, representing the URL of the web page.
- *lastmod* is an optional field in the W3C Datetime format (e.g., YYYY-MM-DD and HH-MM-SS if necessary) representing the last time the URL was known to change.
- *changefreq* is an optional field representing how frequently the page is likely to change. Valid values include always, hourly, daily, weekly, monthly, never.
- *priority* is an optional field representing the relative importance of this URL in the web site.

The full specification of the protocol is available at `http://www.sitemaps.org` and the protocol is extensible to support new types of metadata. Examples of Google's extensions are available at `http://www.google.com/webmasters`. For large sites, the protocol supports SitemapIndex files. Conceptually, these index files list Sitemaps files. This allows websites to break large Sitemaps into smaller Sitemaps (under 10MBs compressed) that are more convenient to download over HTTP. This technique also allows *incremental Sitemaps*, Sitemaps files that only include URLs that are either new or modified in a given time period, allowing sites to provide updates without the need to rewrite large files. More details about the specification and examples are available at `http://www.sitemaps.org`.

In order to inform search engines of new or updated Sitemaps files, along with where these files can be found, websites can use one of the following techniques:

- *Robots.txt file*: Sitemaps can be published in the robots.txt file of a website using the "Sitemaps:" directive. For ex-

Format	Percentage
XML Sitemap	76.76
Unknown	17.51
Url List	3.42
Atom	1.61
RSS	0.11

Table 1: Breakdown of URLs Published by Submission Format

ample, at `http://cnn.com/robots.txt` you see the following line: "Sitemaps: `http://www.cnn.com/sitemaps.xml`." Crawlers can then use this URL to download the corresponding Sitemaps file that contains CNN's URLs.

- *HTTP request*: Search engines may provide *ping URLs* that can be used to notify them of new or updated Sitemaps. In general, a ping URL can take the following form: `http://searchengineURL/ping?sitemap=sitemapUrl`
- *SearchEngine Submission Tools*: Search engines may also provide tools and APIs to submit Sitemaps. Google Webmaster Tools are an example of such tools and APIs (`http://www.google.com/webmasters`).

3.1 Sitemaps Usage

We now discuss some high level statistics on usage of Sitemaps published through websites' robots.txt, at the time of writing this paper (October 2008). Sitemaps are distributed across many countries and top-level-domains:

- Approximately 35M websites publish Sitemaps, and give us metadata for several billions of URLs. The top ten TLDs for these websites are .com, .net, .cn, .org, .jp, .de, .cz, .ru, .uk, .nl. And a long tail of TLDs comprising the last 5% of websites.
- In Table 1 we list the percentage of URLs published, broken down by submission format. Notice that XML Sitemaps are most of the URLs. Websites also list a simple text file with URLs as a UrlList, and submit RSS/Atom feeds if they already support one. The Unknown format includes Sitemaps that are misformatted.
- 58% of URLs include a lastmodification date, 7% include a change frequency field, and 61% include a priority field.

3.2 Sitemaps At Google

In Figure 1 we show how Sitemaps are incorporated into Google's crawling pipeline.

- *Discovery*: Discovery starts with a seed set of URLs, and maintains a prioritized queue of URLs to be crawled. Periodically, it dequeues the next set of URLs and passes this candidate set of to-be-crawled URLs to the SpamFilter component.
- *Sitemaps*: The Sitemaps component takes the set of URLs submitted to Google using ping or published on websites' robots.txt, and passes it on to the SpamFilter component.
- *SpamFilter*: The Spam filter detects and removes the spam links from the set of links it receives [14] before passing the clean set of links to the web crawler.

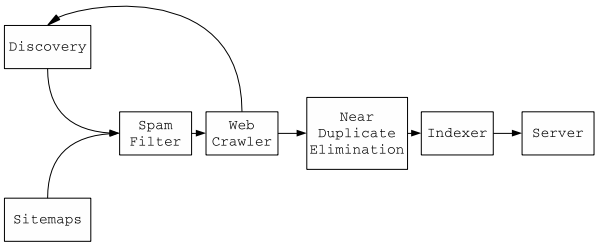


Figure 1: Google Search Main Pipeline Architecture

- *WebCrawler*: The web crawler receives the next set of URLs to be crawled, issues HTTP requests to websites and retrieve the associated page content. URLs are then extracted from the content and passed to the Discovery component. The web crawler also handles the periodic refreshing of pages. [11, 8].
- *Indexer/Server*: The Indexer selects a subset of the crawled pages according to various quality metrics and builds an index of these pages. The specific metrics used will not be discussed in this paper, nor are they pertinent to the discussions in the rest of the paper.
- *Server*: This component serves uses the index to produce responses to users' search queries.

4. SITEMAPS CASE STUDY

In this section we study a few specific sites to give the reader a flavor for how Sitemaps are used under different scenarios and to motivate some of the metrics we introduce in a later section. These sites publish their Sitemaps URL through the special "Sitemaps:" directive in the robots.txt file.

The sites have different characteristics, including: (a) How often their pages change; (b) How many web pages they have; (c) If they have URL canonicalization (or duplication) issues, e.g., how many distinct URLs render the same content; (d) If they use Sitemaps to be exhaustive or to focus on specific content (e.g., recent content). In Section 5, we study these properties across an aggregate set of sites.

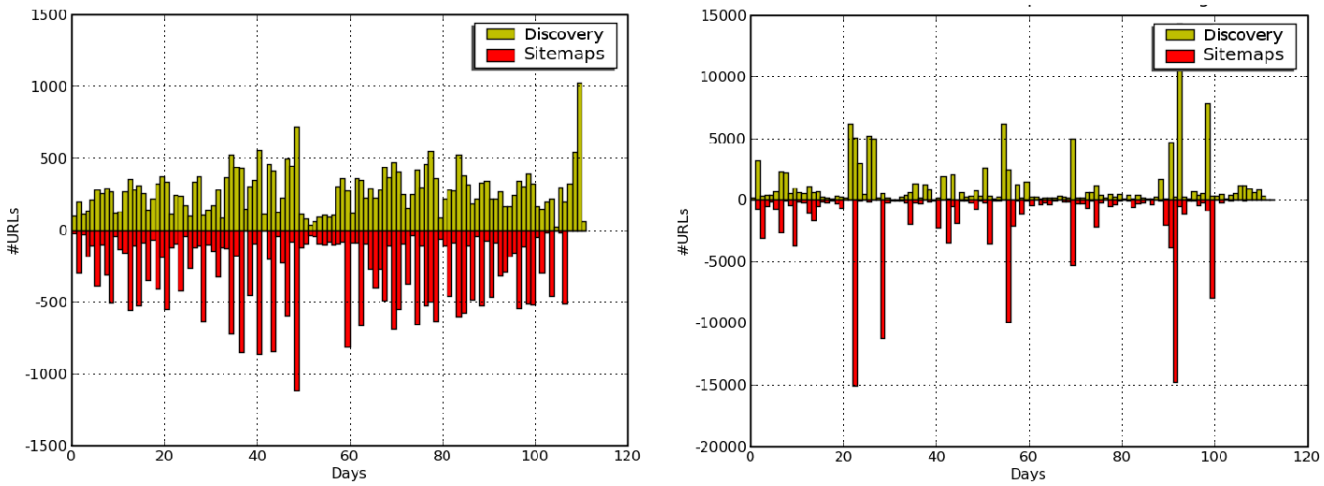
4.1 Amazon.com

Amazon is a popular commercial website with several tens of millions of URLs. The site is built on a service-oriented architecture, where each page is constructed on the fly by calling potentially hundreds of services [19]. Each of these services renders one component in the page such as product information, customer reviews or the cover photo of a book. With such a large and dynamic site, finding new content and keeping the content fresh is clearly a challenge.

Amazon.com also suffers from URL canonicalization issues, multiple URLs reference identical or similar content. For example, our Discovery crawl crawls both

- `http://.../B000FEFEFW?showViewpoints=1`, and
- `http://.../B000FEFEFW?filterBy=addFiveStar`.

The two URLs return identical content and offer little value, since these pages offer two "different" views of an empty customer review list. Simple crawlers cannot detect these type of duplicate URLs without downloading all duplicate URLs first, processing their content and wasting resources in the process. Sophisticated crawlers may use predictive



(a) cnn.com (b) pubmedcentral.nih.gov

Figure 2: Histogram of new URLs over time

techniques [4] to avoid crawling some of these kinds of pages.

Sitemaps usage. At the time of writing this paper, Amazon publishes 20M URLs listed in Sitemaps using the "Sitemaps" directive in amazon.com/robots.txt. They use a SitemapIndex file that lists about 10K Sitemaps files each file listing between 20K and 50K URLs. The SitemapIndex file contains the last modification time for each underlying Sitemaps file, based on when the Sitemaps file was created. When we examined the last-modification dates of each Sitemaps file over the past year, we noticed that new Sitemaps files are added every day and do not change after they are added. Each new Sitemaps file is composed of recently added or recently modified URLs. When we examined the Sitemaps, we saw that Amazon is primarily listing single product pages. Also, Amazon Canada and Amazon UK maintain Sitemaps in a similar manner on their own domains.

Conceptually, Amazon is using a "log structured" approach by appending new Sitemaps each day to a SitemapIndex. This aids different crawls to update their index at different frequencies and yet read a small number of files containing only the new URLs. For example, crawlerA and crawlerB could choose to crawl content from Amazon once a day and once a week respectively, and would only have to examine the last day or week's worth of Sitemaps. Furthermore, providing a set of canonical URLs that provide good coverage of Amazon's products, helps the crawlers prioritize what URLs to crawl as well as pick what URL is the canonical URL that should be used in the search results.

In Section 5 we will introduce several metrics for evaluating the content. For now, consider a simple metric of *efficiency* in terms of URLs crawled vs unique pages crawled. The Discovery crawl of amazon.com is 63% efficient, meaning for every 100 URLs that we crawl, we collect 63 unique pages. The Sitemaps crawl was 86% efficient. To reiterate, this is a simplification and we'll study this in more detail in Section 5.

4.2 Cnn.com

CNN.com is a much smaller site than amazon.com. The site covers breaking news in addition to including other content such as weekly polls and columns. When we examined some of the URLs in CNN, we observed that they have slightly different canonicalization issues. For example, we see the same content in web viewable format and in printable format, and the crawlers may not automatically know which of these URLs to crawl and index.

In figure 2(a) we see the counts of new URLs over time as seen by Sitemaps crawl and Discovery crawl in `www.cnn.com`. This graph shows that the website is very active with many new URLs being added daily. While the number of URLs found by Discovery and Sitemaps are similar, it seems that Sitemaps finds these URLs somewhat faster since the number of URLs appears front-loaded. However, it is clear that Google adjusts the rate at which Discovery crawls the site according to the site's high change rate.

Sitemaps organization. CNN adopts a different approach from Amazon in creating Sitemaps. CNN publishes multiple Sitemaps, each with a small numbers of URLs. The Sitemaps file labeled "news", includes the URLs of pages most recently changed or added. The news-specific Sitemaps changes multiple times each day and has 200–400 URLs. In addition, the weekly Sitemaps file has 2500–3000 URLs and the "monthly" has 5000–10000 URLs. These lists do not overlap but rather complete each other. We also noticed that CNN publishes unique and canonical URLs for the articles. In addition, CNN publishes hub pages on various topics, and these are covered in a SitemapIndex file consisting of additional Sitemaps where the URLs appear organized alphabetically rather chronologically. We expect CNN organized their Sitemaps in this fashion, because they are focused on helping web crawlers find the latest news and articles of that month. Additionally, it is pretty cheap for them to reproduce new Sitemaps periodically, since they have a small number of URLs.

4.3 Pubmedcentral.nih.gov

The pubmedcentral.nih.gov site offers a huge archive of medical journal articles, the earliest journal appearing on the site was published in 1809! The underlying content changes infrequently once published, and new articles are added periodically. The site has some URL duplication issues. For example, two URLs found through our Discovery crawl were

- `http://.../articlerender.fcgi?artid=1456828` and
- `http://.../articlerender.fcgi?&pubmedid=16085696`

are actually the exact same content. However, the URLs look different and no crawler could automatically "guess" these URLs were the same content without downloading them first.

In Figure 2(b) we plot the counts of new URLs over time as seen by Sitemaps crawl and Discovery crawl for this domain. Notice here that URLs seen by Sitemaps occur in large bursts, likely due to when pubmed's Sitemaps generator runs and when our crawler finds the new Sitemaps.

Sitemaps organization. Pubmed organizes its Sitemaps differently from CNN and Amazon. A SitemapIndex points at 50+ Sitemaps files, each with around 30K URLs. The SitemapIndex file is modified each day. One of the Sitemaps consists of about 500 entries for the Table of Contents of journals. All Sitemaps files indicate that change-rate of URLs is monthly. However, we noticed in general that some of the last modified metadata was inaccurate unlike CNN and Amazon. For example, some Sitemaps were listed as recently updated but the underlying content had not changed. In other cases, the publication date of the last volume was newer than the last modification date in the Sitemaps file.

5. EFFICACY OF SITEMAPS

In this section we explore the possible benefits of Sitemaps to its users, meaning its benefit both to websites and to web crawlers. While we present our techniques in the context of domains, they are equally applicable to hosts or smaller units of a website.

5.1 Coverage

The first aspect we explore is the notion of coverage, including (a) how much of a domain is captured in Discovery and Sitemaps, and (b) how much of the "useful" part of a domain is captured. We introduce some metrics for this evaluation and then discuss our experimental results.

5.1.1 Metrics

We can define many possible coverage metrics based on different parameters, including page content and the link structure. In [12] Cho et al. present a coverage metric based on PageRank – how much of a domain's overall PageRank is captured in the crawled document collection. This class of metrics builds on the acceptance of PageRank as a proxy for quality. We adapt this metric to allow us to compare the coverage achieved by Discovery and Sitemaps. We further introduce additional coverage metrics specifically suited in the context of search engines.

For this paper, we adopt a simple definition of uniqueness based on *Shingles* [22]. Consider all k-grams in a document. The similarity of two documents is then defined as the percent of k-grams shared between the documents. Two documents are considered near-duplicates if the percent of shared k-grams is above some threshold value. The Shingles technique allows identifying such near-duplicates efficiently. The Shingles are produced by hashing each of the k-grams and choosing some small subset of these hashes. The Shingles can then be compared directly rather than comparing the documents themselves. For a study of some of Shingling techniques and alternate techniques, please read [22].

Let us first define a few key measures to motivate some coverage metrics. Consider how URLs propagate through the search engine pipeline we discussed in Figure 1. For a domain, the URLs can be classified into the following states. Conceptually, a URL is more "useful" as it progresses through each of the following sequence of states:

1. *Seen*: The set of URLs seen by the web crawler.
2. *Crawled*: The set of URLs the web crawler fetched.
3. *Unique*: The set of URLs after eliminating duplicates.
4. *Indexed*: The set of URLs the Indexer included in the index.
5. *Results*: The set of URLs the Server showed to the user in response to a query.
6. *Clicked*: The set of URLs the user clicked on in the results page.

For domain D, we define the following coverage metrics:

$$Coverage(D) = \frac{Crawled_{Sitemaps}(D)}{Crawled(D)} \quad (1)$$

$$UniqueCoverage(D) = \frac{|Unique_{Sitemaps}(D)|}{|Unique(D)|} \quad (2)$$

$$IndexCoverage(D) = \frac{|Indexed_{Sitemaps}(D)|}{|Indexed(D)|} \quad (3)$$

$$PageRankCoverage(D) = \frac{\sum PageRank_{Sitemaps}(D)}{\sum PageRank(D)} \quad (4)$$

Where $Unique_{Sitemaps}(D)$ refers to the unique URLs in Sitemaps in Domain D, and *Unique* refers to unique URLs known to either Discovery or Sitemaps in Domain D. We use similar notation for Indexed URLs and PageRank for URLs.

One additional metric is not directly related to coverage but is useful in evaluation the quality of either crawl:

$$SignalToNoise(D) = \frac{|Unique(D)|}{|Seen(D)|} \quad (5)$$

These metrics combined together give us insights into the quality of URLs in Sitemaps. SignalToNoise evaluates the fraction of URLs that are worth further consideration (for Discovery and Sitemaps), while UniqueCoverage, IndexCoverage and PageRankCoverage compare the URLs from Sitemaps compared to what we know globally about the domain.

5.1.2 Coverage on pubmedcentral.nih.gov

Conceptually, archival domains such as `pubmedcentral.nih.gov` can be exhaustively covered using Sitemaps because it is easy to generate a list of the important URLs of this domain and the content of these pages changes at a low rate. We evaluate this domain carefully to consider what types of URLs are left out of Sitemaps.

- *Seen and Crawled*: Discovery and Sitemaps together crawled approximately 3 Million URLs from this domain. Sitemaps only contained 1.7 million URLs.

- *Duplicates*: One million of above missed URLs were found to be near-duplicates when applying the Shingling techniques discussed earlier. By contrast, Sitemaps had only 100 duplicate URLs.

- *Missing content*: We manually examined a random sample of 3000 URLs of the $300K$ URLs that Sitemaps "missed". Table 2 outlines the examination results .

Type of missed content	%age
Errors - bad certificates and "404s"	8%
Redirects	10%
Dup content – extra CGI parameter	3%
Dup content – unmatched URLs	8%
Table of contents	4%
Image dominated page	30%
Subset of another page	17%
Missing content	20%

Table 2: Classifying missing content from pubmedcentral.nih.gov

- First 8% of the URLs generate errors and report bad certificates but do not return the right HTTP error codes.
- The next three lines of the table account for redirect URLs and duplicate URLs that were not identified as duplicates, and add up to 21% of the missed URLs. We include redirect URLs in this category, because a crawler has to spend work units to crawl the redirecting URL first. Also the duplicate URLs that are not identified as duplicates are due to false negatives from the Shingling algorithm [22].
- 51% of the URLs have content where a judgement call needs to be made about the usefulness of the URLs – will the domain or search engines want some of these pages indexed?
- The other 20% of content appear to be missing content that Discovery found and was not in Sitemaps.

The main observation from this study is that for an archival domain, Discovery is 63% "efficient" and Sitemaps is 99% efficient in crawling the domain at the cost of missing a small fraction of the content. In the next sections, we evaluate the quality based coverage metrics as an aggregate over a set of domains.

5.1.3 *Experimental Results*

In this section we will first introduce our results and follow with our observations. The dataset we chose to examine in this section consists of all the known URLs starting with the prefix "am" and belonging to the top level domain ".com", and were crawled sometime in 2008. This dataset covers about 500 million URLs. In Figure 3 we plot UniqueCoverage vs DomainSize (# URLs crawled) on a *log-log* plot of URLs seen both by Sitemaps and Discovery. The size (and color) of the dot represents the number of URLs in the domain. A color version of the paper is available online in [30]. We use color primarily to make it easier to differentiate the domains that are close to each other on the graph.

Figure 4 similarly shows UniqueCoverage vs Coverage. For example, the big outlying (yellow) dot in the center of the graph represents amitymama.com with 6.4 million URLs, with Coverage = 0.55 and UniqueCoverage = 0.64.

Finally, Figure 5(a) and Figure 5(b) plot PageRankCoverage vs DomainSize and IndexCoverage vs UniqueCoverage, which together give us insights into the quality of the Sitemaps URLs.

Putting all of this together, the above graphs allow us to identify and make the following patterns and observations:

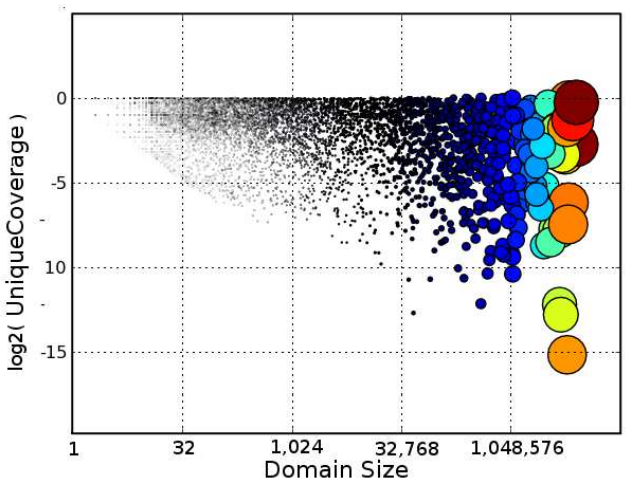


Figure 3: Unique Coverage vs Domain Size (URLs seen)

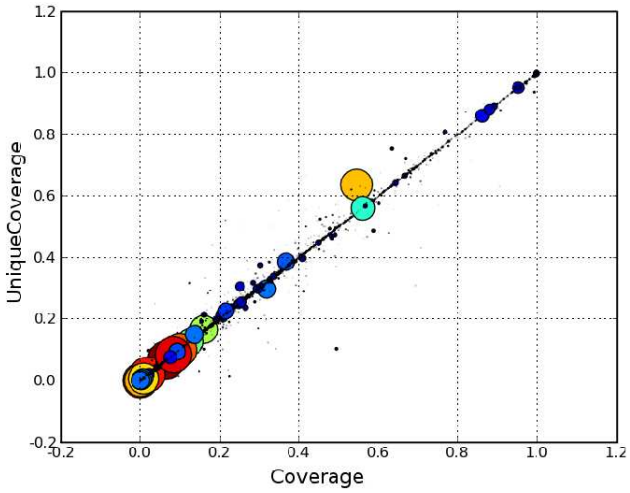
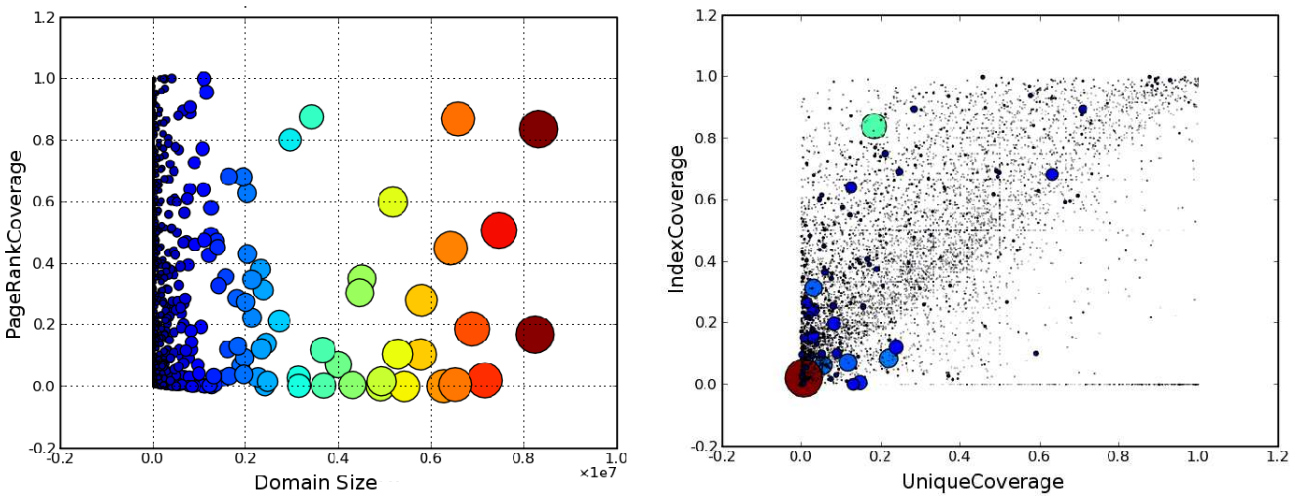


Figure 4: Sitemaps UniqueCoverage vs Coverage

- *Duplicate URLs*: Figure 4 indicates that the percent of duplicates inside Sitemaps is mostly similar to the overall percent of duplicates.
- *Coverage*: Figure 3 shows how the coverage of domains are distributed by their size. At least 46% of the domains have above 50% UniqueCoverage and above 12% have above 90% UniqueCoverage. There is no apparent correlation between the size of the domain and UniqueCoverage.
- *Quality*: Figures 5(a) and 5(b), show us that according to both IndexCoverage and PageRankCoverage some domains have very high quality Sitemaps coverage while others are spammier. For example, in Figure 5(b) we see that most domains are above the diagonal. This indicates that for most domains, Sitemaps achieves a higher percent of URLs in the index with less unique pages. That is, Sitemaps crawl attains a higher *utility*.



(a) PageRankCoverage vs Domain Size

(b) IndexCoverage vs Unique Coverage

Figure 5: Quality of URLs seen through Discovery and Sitemaps

5.2 Freshness

The second metric we wish to examine is that of freshness. Are sites using Sitemaps more likely to be fresh? We introduce some metrics we use and discuss our experimental results.

5.2.1 Metrics

Freshness can be defined in many ways. One useful measure is the difference between the content of a page saved on the search engine side and the content currently being served by the web server. This value can then be aggregated in different manners to produce a single freshness score [9, 25].

In this paper we mainly focus on a simpler definition – what is the time difference between the first time Discovery sees a URL and the first time Sitemaps sees it. The alternative freshness measures mentioned above are complementary to this definition.

For this purpose we define the *SeenDelta* measure, for a period of time T and a set of URLs D:

$$SeenDelta(D_T) = \sum_{u \in D_T} FirstSeen_{Discovery}(u) - FirstSeen_{Sitemaps}(u) \quad (6)$$

Where D_T is the set of URLs in domain D first seen in T by both Sitemaps and Discovery, $FirstSeen_{Sitemaps}(u)$ is the time Sitemaps first saw URL u and $FirstSeen_{Discovery}(u)$ is the time Discovery first saw URL u. In the next section we use this metric to evaluate the freshness of URLs seen through Discovery and Sitemaps.

5.2.2 Experimental Results

We tracked URLs found through Discovery crawl and Sitemaps over a period of 6 months. In Figure 6(a), for the URLs in the domain `pubmedcentral.nih.gov` which were seen both by Sitemaps and Discovery, we plot the day Sitemaps first saw the URLs compared to the day Discovery first saw the URLs. Each dot represents a cluster of URLs that happen to fall on the same point in the graph, and the size of the dot represents the size of the cluster. We see that in the domain `pubmedcentral.nih.gov` Discovery and Sitemaps perform equally well for many URLs. However, there are some URLs seen significantly later by the Discovery crawl than by the Sitemaps crawl. In an archival site such as Pubmed, a less frequent Discovery crawl makes sense because Sitemaps can achieve better freshness at a potentially lower price. Similarly, in Figure 6(b) we see the same graph for the `cnn.com` domain. Since CNN's site is an important and dynamic one, Discovery likely crawls at a high frequency and the difference is less obvious.

First Seen by Sitemaps	Source ping?	URL Count
yes	yes	12.7%
	no	80.3%
no	yes	1.5%
	no	5.5%

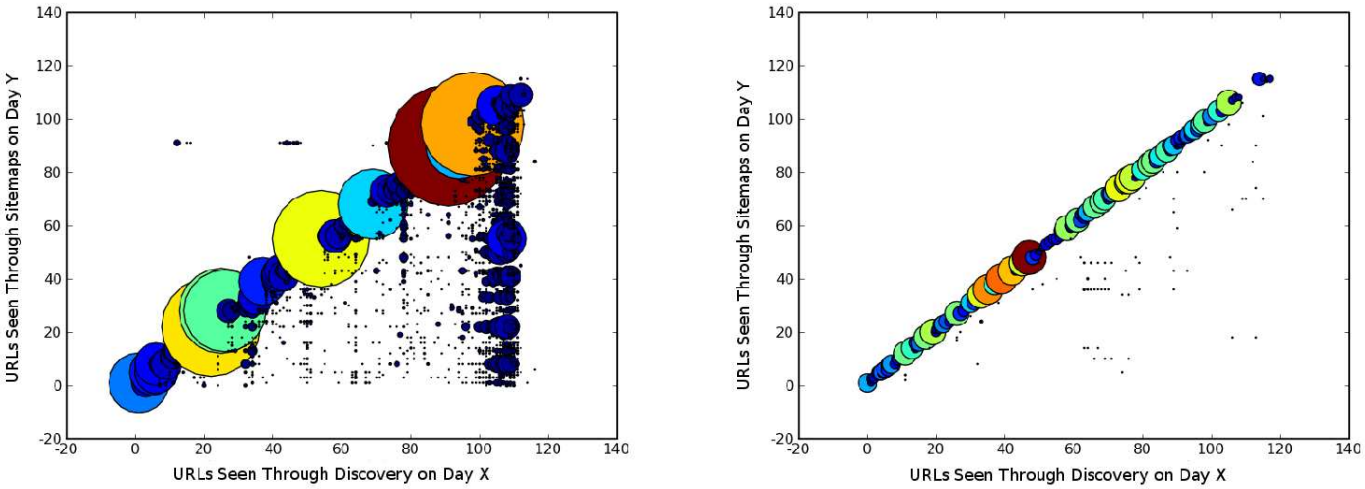
Figure 7: First Seen and Ping URL Count

Next, we study which of the two crawl systems, Sitemaps and Discovery, sees URLs first. We conduct this test over a dataset consisting of over five billion URLs that were seen by both systems. According to the most recent statistics at the time of the writing, 78% of these URLs were seen by Sitemaps first, compared to 22% that were seen through Discovery first.

Finally, we study how usage of ping affects Sitemaps and Discovery. The use of ping requires more effort to setup than the use of robots.txt. Thus it is not surprising that only 14.2% of URLs are submitted through ping. But as can be seen in Figure 7, the probability of seeing a URL through Sitemaps before seeing it through discovery is independent of whether the Sitemaps was submitted using pings or using robots.txt.

6. SEARCH ENGINE USE OF SITEMAPS

In previous sections, we got a flavor for the tradeoffs and metrics involving the (a) quality of URLs and some of the metadata, (b) freshness impact of different crawls. In this section, we consolidate some of our learnings and propose



(a) pubmedcentral.nih.gov (b) cnn.com

Figure 6: Firstseen by Discovery vs Sitemaps

how to integrate Sitemaps into existing web crawlers to complement Discovery and gain the best of both worlds.

6.1 Crawling Sitemaps Files

We now discuss techniques to fetch Sitemaps and periodically refresh them.

- Finding Sitemaps: As discussed earlier, crawlers support several methods that webmasters can use to indicate that a Sitemaps file is available (e.g., through robots.txt). This step is cheap, since crawlers already need to crawl robots.txt periodically in order to keep up with the changes webmasters make and processing requests made through the ping URL interface is cheap as well.

- Downloading Sitemaps: This step is inexpensive because the number of Sitemaps corresponds roughly to the number of hosts (in the tens of millions), which is small compared to the number of URLs a crawler processes. Many Sitemaps are of course larger than most web pages (up to 10 MBs), even after compression is used. However, in our experience, websites and crawlers alike would rather avoid the overhead of additional HTTP connections even at the price of greater transfer sizes (up to a few MBs).

- Refreshing Sitemaps: As discussed earlier, the protocol supports web servers supplying metadata as to when Sitemaps change. For example, when a SitemapIndex file is provided, it may supply additional metadata about the last date in which a Sitemaps file has changed. If the web servers were accurate and honest, crawlers could rely solely on this information to keep track of the Sitemaps files. But as our experiments indicate, the metadata is not yet consistently trustworthy enough. Hence we should apply the same techniques Discovery crawl uses to keep a web archive fresh to refresh Sitemaps. For an example consider [9].

6.2 Canonical URL Selection

As we saw in the case studies in Section 4 and our discussions earlier, near duplicates and different URLs with very similar content are a well known reality on the web. Crawlers deal with this problem by (a) clustering pages that have similar content using techniques described in Section 5.1, and (b) choosing a canonical URL for each of these clusters. This URL is used to track the cluster and is returned in search results. Choosing the canonical URL for each cluster is a non-trivial problem that has not been extensively discussed in the research community.

A possible reason for this is that there is no clear cut answer. For example, the URL `http://borders.com` redirects to the URL `http://www.borders.com/online/store/Home`. Another example, the URL `http://en.wikipedia.org/w/index.php?title=Sitemaps&printable=yes` points at a printable version of this URL `http://en.wikipedia.org/wiki/Sitemaps`. As these examples may hint, a commonly used good heuristic is to use the shorter URL as the canonical URL, or the URL with the highest PageRank. An alternative is to use Sitemaps to give the power to webmasters, and choose the URL listed in a Sitemap file if one is available as the canonical URL.

6.3 Combining Sitemaps Feeds in Crawl Order

Both web sites and web crawlers have limited resources and crawlers need to prioritize which URLs to crawl. Typically, separate policies are used for the crawl of new URLs and for the crawl that maintains the freshness of the current document collection. In this section, we discuss how a search engine crawler can work in the URLs seen through Sitemaps with its existing crawling policies.

6.3.1 New Content

Significant research has gone into studying the order in which URLs should be crawled [12, 2, 23]. For example, in [12] Cho et al. propose the RankMass crawling algorithm.

ALGORITHM 6.1.
Function CrawlScheduleForDomain(N_{crawl},ϕ, $\mathcal{D}$)
1: $k_S = \frac{1}{2}N_{crawl}$
2: $k_D = \frac{1}{2}N_{crawl}$
3: $\Delta = \phi * \frac{1}{2}N_{crawl}$
4: $utility_{Sitemaps} = utility_{Discovery} = 0$
5: *for epoch in* $0..\infty$ *do*
: *Get Set of URLs Seen but Not Crawled in* $\mathcal{D}$
6: $CrawlQ_{Sitemaps}$ =
: $Seen_{Sitemaps}(\mathcal{D})$ / $Crawled_{Sitemaps}(\mathcal{D})$
7: $CrawlQ_{Discovery}$ =
: $Seen_{Discovery}(\mathcal{D})/Crawled_{Discovery}(\mathcal{D})$
8: *sort(*$CrawlQ_{Sitemaps}$*, SitemapScore)*
9: *sort(*$CrawlQ_{Discovery}$*, DiscoveryScore)*
10: *if (*$utility_{Sitemaps} < utility_{Discovery}$*):*
11: $k_D = k_D + \Delta$
12: $k_S = k_S - \Delta$
13: *else:*
14: $k_S = k_S + \Delta$
15: $k_D = k_D - \Delta$
16: $topK_S$ = *Crawl top* k_S *of* $CrawlQ_{Sitemaps}$
17: $topK_D$ = *Crawl top* k_D *of* $CrawlQ_{Discovery}$
18: *Crawl additional URLs by priority if Quota permits*
19: $utility_{Sitemaps} = utility(topK_S, \Delta)$
20: $utility_{Discovery} = utility(topK_D, \Delta)$
21: $\Delta = \phi * min(|k_S|, |k_D|)$
22: *Update Crawled and Seen sets.*

Figure 8: Combined Crawling Algorithm Sitemaps and Discovery

This algorithm produces a URL ordering for Discovery using an approximation of the Personalized PageRank of pages yet to be downloaded. In the case of Sitemaps URLs, however, some pages may not have any incoming links despite being "quality" pages. In Figure 5(b), we saw that the quality of pages seen through Discovery and Sitemaps varies from domain to domain. For domains above the diagonal, it would make sense to allocate more resources for Sitemaps URLs. In this section we discuss how Sitemaps URLs should be integrated into existing URL ordering policies, including (a) How to order Sitemaps URLs, and (b) How to integrate Sitemaps ordering with Discovery ordering?

Sitemaps Crawl Order. We wish to specify a priority function, *SitemapScore*, which assigns each URL in the domain's Sitemaps a score that indicates the importance of crawling it relative to the other URLs in the domain. Given a per-domain quota on the number of URLs we can crawl, we can then simply choose the URLs with the highest scores for crawl. We suggest several possible definitions for SitemapScore:

- *Priority*: Set SitemapScore equal to the priorities supplied by webmasters, and in the case priorities are not supplied, all URLs can be assumed to have equal priority. This ranking system operates under the assumption that the web master knows best.
- *PageRank*: Set SitemapScore equal to the PageRank of the page, or to zero if no incoming links exist. This system will give good results in domains that have good PageRankCoverage, as we saw in Figure 5(a). However, it would not enable leveraging Sitemaps to crawl new quality pages before the first incoming link is seen.
- *PriorityRank*: A hybrid approach which combines the two signals. The root of the domain is assumed to contain an implicit link to the Sitemaps file. In addition, the Sitemaps file is assumed to contain links to all of its URLs, either uniformly weighted or weighted by priority if this field is provided. Over this augmented graph, PageRank can be calculated for every URL in the Sitemaps page. We call this PageRank score *PriorityRank*. PriorityRank combines the two previous scoring methods, taking the web masters' preferences into account and boosting the score of pages that already have incoming links from high ranking pages. The idea of adding "virtual" links to allow PageRank to be calculated is not new [6]. For example, it is used to deal with dangling pages, pages with no outgoing links.

Next we will present how to use these scoring functions together with an existing crawling policy.

Prioritizing Between Discovery URLs and Sitemaps URLs. Given two scoring functions, *SitemapScore* (like above) and *DiscoveryScore*, we wish to choose some subset of the uncrawled URLs to crawl next. For this discussion, we assume that each domain is restricted by a quota, N_{crawl}, which sets the limit on the number of URLs that can be fetched in a given time period (e.g. day, week, etc.). We wish to use the scoring functions to choose the top k_D of the Discovery URLs and the top k_S of the Sitemaps URLs such that $k_S + k_D \leq N_{crawl}$. The question is, how do we balance between k_S and k_D, given that that the scoring functions are different. We draw from Figure 5(b) a *utility* metric to balance between these two queues. Specifically, we propose the value $\frac{IndexCoverage}{UniqueCoverage}$ which represents the percent of URLs that make it to the index out of the total number of URLs crawled.

Figure 8 presents an algorithm that uses this utility metric to iteratively converge on the right balance between the two URL sources for each domain. Conceptually, the utility of a sample of the lowest priority URLs crawled in the last time-period will determine how to adjust the balance for the next one. Specifically,

- In Figure 8, lines 1-4 set the initial values of the k_S, k_D, utility and Δ parameters. The Δ parameter controls the size of the sample of URLs the utility is measured over as well as the size of the adjustment to k_S and k_D. The value ϕ controls how fast the balance is adjusted and can be set through empirical evaluation (e.g., 0.1 is a slow converging default[1]). Lines 6-9 set and sort the crawl queues for the current time period (epoch). According to the previous epoch's utility statistics, the number of URLs to crawl from each queue are either incremented or decremented by Δ in lines 10-15. The top k_S and k_D URLs are crawled from the respective queues in lines 16-17 and additional URLs are crawled if quota permits (line 18). The utility statistics are updated in lines 19-20 using the utility function explained

[1]If a crawler retains additional histograms or has special knowledge about a domain, ϕ can be optimized based on the derivatives for functions SitemapScore and DiscoveryScore using classical numerical techniques.

shortly. Finally, the size of Δ, the Crawled set and the Seen set are all updated according to the latest crawl (lines 19-22).

- The *utility* function performs the following calculation. A subset of the list of URLs passed as the first parameter, $topK$, is chosen such that it includes the Δ URLs with the lowest priority. The function returns the number of URLs in this subset that were added to the search engine index, divided by the size of the subset, Δ. This function measures the expected incremental contribution of adding resources to each of the sources, that is, the derivative of the number of URLs indexed by URLs downloaded. In this manner, good balance can be found for domains depending on their specific characteristics (Figure 5(b)).

6.3.2 Refresh Crawl

The purpose of a typical Refresh Crawl process is simple, for a limited amount of resources, re-crawl the pages in your document collection to achieve maximal freshness. Part of this process typically requires, whether directly or indirectly, approximating the rate at which the content changes.

In the case of the Refresh Crawl policy, integrating the information provided by the Sitemaps feeds can simply be done by using the "last modified date" meta data instead of the approximate date. One caveat of this approach is that the accuracy of this field varies from site to site. In this case, using the historical data to approximate the accuracy of this metadata can be used to decide which source is more accurate, the history of crawling the URLs or the metadata supplied in the Sitemaps.

7. CONCLUSIONS

In this paper we provided insight on how users use Sitemaps today. We defined metrics for evaluating the quality of Sitemaps data and performed a large scale experimental evaluation over real web data. Finally, we presented several techniques of making use of Sitemaps data inside search engines including a crawling algorithm that finds the right balance between Discovery crawl and Sitemaps crawl.

8. REFERENCES

[1] H.T. Lee, D. Leonard, X. Wang, and D. Loguinov IRLBot: Scaling to 6 billion pages and beyond. In *Proc. 17th WWW*, 2008.

[2] R. Baeza-Yates, C. Castillo, M. Marin, and A. Rodriguez. Crawling a country: Better strategies than breadth-first for Web page ordering. In *Proc. 14th WWW*, pages 864–872, 2005.

[3] Z. Bar-Yossef and M. Gurevich. Random sampling from a search engine's index. In *Proc. 15th WWW*, pages 367–376, 2006.

[4] Z. Bar-Yossef, I. Keidar and U. Schonfeld Do not crawl in the DUST: different URLs with similar text. In *Proc.16th WWW*, pages 111–120, 2007.

[5] M.K. Bergman. The Deep Web: Surfacing hidden value. *Journal of Electronic Publishing*, 7(1):07–01, 2001.

[6] M. Bianchini, M. Gori, and F. Scarselli. Inside PageRank. *ACM Transactions on Internet Technology (TOIT)*, 5(1):92–128, 2005.

[7] O. Brandman, J. Cho, H. Garcia-Molina, and N. Shivakumar. Crawler-friendly Web servers. *ACM SIGMETRICS Performance Evaluation Review*, 28(2):9–14, 2000.

[8] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[9] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. *ACM SIGMOD Record*, 29(2):117–128, 2000.

[10] J. Cho and H. Garcia-Molina. The evolution of the Web and implications for an incremental crawler. In *Proc. 26th VLDB*, pages 200–209, 2000.

[11] J. Cho and H. Garcia-Molina. Effective page refresh policies for Web crawlers. *ACM Transactions on Database Systems (TODS)*, 28(4):390–426, 2003.

[12] J. Cho and U. Schonfeld. RankMass Crawler: A crawler with high PageRank coverage guarantee. In *Proc. 33rd VLDB*, volume 7, pages 23–28.

[13] D. Fetterly, M. Manasse, M. Najork, and J.L. Wiener. A large-scale study of the evolution of Web pages. *Software Practice and Experience*, 34(2):213–237, 2004.

[14] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam Web pages. In *Proc. 7th WebDB*, pages 1–6, 2004.

[15] Google. Joint support for the Sitemap protocol. Available online at: `http://googlewebmastercentral.blogspot.com/2006/11/joint-support-for-sitemap-protocol.html`, 2006.

[16] Google. Retiring support for OAI. Available online at: `http://googlewebmastercentral.blogspot.com/2008/04/retiring-support-for-oai-pmh-in.html`, 2008.

[17] Google. We knew the web was big... Available online at: `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`, 2008.

[18] Microsoft Google, Yahoo. Sitemaps.org. Available online at: `http://sitemaps.org`, 2008.

[19] J. Gray. A conversation with Werner Vogels. Available online at: `http://www.acmqueue.com/modules.php?name=Content&pa=showpage&pid=388`, 2006.

[20] Open Archive Initiative. Open archive. Available online at: `http://www.openarchives.org`, 2008.

[21] J. Madhavan, D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, and A. Halevy. Google's Deep Web crawl. In *Proc. 34th VLDB*, 2008.

[22] G.S. Manku, A. Jain, and A.D. Sarma. Detecting near-duplicates for Web crawling. In *Proc. 16th WWW*, pages 141–150, 2007.

[23] M. Najork and J.L. Wiener. Breadth-first crawling yields high-quality pages. In *Proc. 10th WWW*, pages 114–118, 2001.

[24] A. Ntoulas, J. Cho, and C. Olston. What's new on the Web?: The evolution of the Web from a search engine perspective. In *Proc. 13th WWW*, pages 1–12, 2004.

[25] C. Olston and S. Pandey. Recrawl scheduling based on information longevity. 2008.

[26] S. Pandey and C. Olston. User-centric Web crawling. In *Proc. 14th WWW*, pages 401–411, 2005.

[27] P. Ranjan and N. Shivakumar. Sitemaps: A content discovery protocol for the Web. In *Proc. 17th WWW*, 2008.

[28] Reuters. Google, 4 states partner on government info search. Available online at: `http://www.reuters.com/article/domesticNews/idUSN2946293620070430?sp=true`, 2007.

[29] J.L. Wolf, M.S. Squillante, P.S. Yu, J. Sethuraman, L. Ozsen, and L. Ozsen. Optimal crawling strategies for Web search engines. In *Proc. 11th WWW*, pages 136–147, 2002.

[30] U. Schonfeld and N. Shivakumar. Sitemaps: Above and beyond the crawl of duty. In *Proc. 18th WWW*, 2009. Available online at: `http://www.shuri.org/publications/www2009_sitemaps.pdf`, 2009.