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].

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-