# Evaluation of Crawling Policies for a Web-Repository Crawler

Frank McCown
Old Dominion University
Computer Science Department
Norfolk, Virginia, USA 23529
fmccown@cs.odu.edu

Michael L. Nelson
Old Dominion University
Computer Science Department
Norfolk, Virginia, USA 23529
mln@cs.odu.edu

## ABSTRACT

We have developed a web-repository crawler that is used for reconstructing websites when backups are unavailable. Our crawler retrieves web resources from the Internet Archive, Google, Yahoo and MSN. We examine the challenges of crawling web repositories, and we discuss strategies for overcoming some of these obstacles. We propose three crawling policies which can be used to reconstruct websites. We evaluate the effectiveness of the policies by reconstructing 24 websites and comparing the results with live versions of the websites. We conclude with our experiences reconstructing lost websites on behalf of others and discuss plans for improving our web-repository crawler.

## Categories and Subject Descriptors

H.3.5 [**Information Storage and Retrieval**]: Online Information Services—*Web-based services*

## General Terms

Measurement, Experimentation, Design

## Keywords

digital preservation, website reconstruction, crawler policy, search engine

## 1. INTRODUCTION

When a website is lost due to a hard drive crash, file system failure, virus or hacking, a webmaster may breathe a sigh of relief if she has been diligent in backing-up her site. Unfortunately, backups are often considered only *after* a catastrophic loss has occurred. In cases such as negligence, fire or death of the website owner, backups are often unavailable for recovering a website. When a website is lost and no backups exist, webmasters and third parties often turn to the Internet Archive (IA) "Wayback Machine" for help. Although IA may be of assistance, their index is at least six months out-of-date [26], and their best-effort approach may miss numerous websites and web resources. More recent copies of missing web content may be found in the caches of commercial search engines like Google, MSN and Yahoo.

We have built a new type of crawler, a *web-repository crawler*, that is used for reconstructing lost websites when backups are unavailable [34, 49]. Unlike a traditional web crawler, a web-repository crawler retrieves web resources from web repositories. Web repositories are web archives (e.g., the Internet Archive) and search engine caches (e.g., Google, Yahoo, MSN) that contain web resources that are accessible using a direct URL. Like a traditional web crawler, a web-repository crawler must download resources systematically and find links to other related resources.

There are many challenges for a web-repository crawler: efficiently requesting web resources from web repositories, finding links to missing resources, and evaluating which resource to keep when several versions of the same resource have been recovered. The first two issues are explored in more detail in this paper.

We describe an experiment evaluating three different crawling policies for our web-repository crawler named Warrick. In early 2006 we downloaded 24 websites that varied in size, subject matter, MIME types and top-level domains. We then reconstructed three different versions of each website using three different crawling policies. We compared the downloaded versions with the reconstructed versions and found the policies to significantly affect the number and type of resources recovered and the number of repository requests that were issued. We show the results of our analysis and discuss our experiences reconstructing three websites which had been lost. We conclude with a discussion of future improvements for our web-repository crawler and research goals in reconstructing websites.

## 2. BACKGROUND AND RELATED WORK

A crawler is a program that repeatedly downloads and stores web resources, typically in batch mode. One or more seed URLs are initially placed in the *crawl frontier*. The crawler retrieves a URL from the frontier, downloads the web resource, extracts URLs from the downloaded resource and adds the new URLs to the frontier. The crawler continues in this manner until the frontier is empty or some other condition causes it to stop. The crawler may extract URLs from the frontier based on some prioritization scheme, and the crawler typically adds URLs to the frontier that meet some predefined criteria (e.g., URLs less than $n$ levels deep that point to resources in the $x$ domain).


*HT'06,* August 22–25, 2006, Odense, Denmark.

Web crawling is useful for a variety of purposes. For example, search engines use crawling to build their indexes, researchers use crawling to measure properties of the Web, businesses use crawling to mine data about their competitors, and archivists use crawling to populate their web archives.

Web crawling has mainly been studied in the context of search engines, web characterization, web archiving and the deep web. Search engines are interested in using crawlers that keep their index fresh [1, 11, 18, 52] and populated with the most popular pages [3, 10, 38]. Due to the enormous size of the Web, researchers have increasingly focused on developing crawlers that can produce large crawls quickly through distributed and parallel crawling approaches [12, 24, 47]. Work has also focused on identifying duplicate web content [13, 20] and web spam [19] during crawling to avoid populating search engine indexes with low quality pages.

Web characterization is focused on measuring properties of the Web [41] or subsets of the Web [2, 22]. Here the focus is on the crawling policies or strategies used to produce an accurate representation of the Web [14, 46].

Web archivists want to preserve the Web for posterity and research, and they employ crawlers that are skilled at producing deep crawls of websites [36, 48]. Focused or topical crawlers may be used by archivists and others that seek out only Web resources that pertain to a specific topic [9, 17, 37] or language [27].

Crawling the deep or hidden web is appealing on multiple fronts because of the high-quality resources that cannot be found through traditional crawling [6]. Efforts to crawl the deep web have focused on performing queries on Web search interfaces using human-assisted techniques [29, 30, 45] and automated methods [40]. Other mechanisms like Google Sitemaps [23] and mod_oai [39] attempt to discover a website's resources without traditional web crawling.

The literature is scant in reference to web-repository crawling. As far as we know, our work in [34, 49] was the first to address the possibility of reconstructing websites by crawling web archives and search engine caches. Others have compared crawling policies or strategies for focused crawlers [35], general content crawlers [14] and crawlers maintaining fresh or popular pages [10, 11, 38]. This work compares crawling policies used to enhance web-repository crawling.

Investigation of web-repository crawling is part of a larger goal of evaluating the Web Infrastructure (WI) as a preservation strategy. The WI is composed of commercial web search engines (e.g., Google, Yahoo, MSN), personal web archives (e.g., Spurl.net and Hanzo:web), web archives operated by non-profit companies (e.g., the Internet Archive) and research projects (e.g., CiteSeer and NSDL). Most preservation strategies involve some form of refreshing and migrating [50]. The WI actively engages in refreshing and migrating web resources, often as a natural by-product of their everyday operations. Many of the issues involved in preserving web pages are addressed by Marshall and Golovchinsky [32] when preserving literary hypertext. We have recently harnessed the WI for automatically locating missing web pages [25].

# 3. RECONSTRUCTING WEBSITES

## 3.1 Definition

We define a **reconstructed website** to be the collection of recovered resources that share the same URIs as the resources from a lost website or from some previous version of the lost website. The recovered resources may be equivalent to, or very different from, the lost resources. For websites that are composed of static resources, recovered resources would be equivalent to the files that were lost. For sites produced dynamically using CGI, PHP, etc., the recovered resources would be equivalent to the client's view of the resources and would be useful to the webmaster in rebuilding the server-side components. The server-side components are currently not recoverable from web repositories (see Section 6).

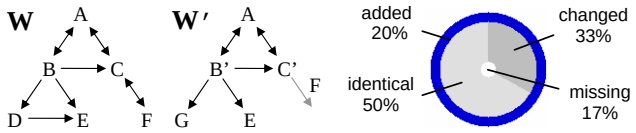


**Figure 1: Lost website (left), reconstructed website (center) and reconstruction diagram (right).**

Our web-repository crawler is only able to recover content that can be identified uniquely by a URI. Content that is dynamically produced but is not uniquely identified by a URI presents difficulty for web repositories that rely on URIs for identifying distinct resources. Other mechanisms like the robots exclusion protocol (robots.txt), NOARCHIVE meta tags, session IDs, password-protection, Flash pages and links generated by JavaScript may prevent or hinder web repositories from storing many web resources. In light of these difficulties, search engines like Google offer tips for webmasters to make their websites easier to crawl [51], and an entire industry (search engine optimization) has even developed to help commercial websites improve their footprint in search engine indexes.

## 3.2 Reconstruction Measurements

To quantify the difference between a reconstructed website and a lost website, we use the graphs produced by the websites to classify the recovered resources as first introduced in [34]. A website can be represented as a graph $G = (V, E)$ where each resource $r_i$ (HTML, PDF, image, etc.), identified by a URI, is a node $v_i$, and there exists a directed edge from $v_i$ to $v_j$ when there is a hyperlink or reference from $r_i$ to $r_j$. This graph may be constructed for any website by downloading the host page (e.g., `http://foo.com/`) and looking for links or references to other web resources, a method employed by most web crawlers.

The left side of Figure 1 shows a web graph representing some website $W$ if we began to crawl it beginning at A. Suppose $W$ was lost and reconstructed forming the website $W'$ represented in the center of Figure 1.

For each resource $r_i$ in $W$ we may examine its corresponding resource $r'_i$ in $W'$ that shares the same URI and categorize $r'_i$ as

1. *identical* – $r'_i$ is byte-for-byte identical to $r_i$
2. *changed* – $r'_i$ is not identical to $r_i$
3. *missing* – $r'_i$ could not be found in any web repository and does not exist in $W'$

We would categorize those resources in $W'$ that did not share a URI with any resource in $W$ as

4. *added* – $r'_i$ was not a part of the current website but was recovered due to a reference from $r'_j$

For those resources that are 'changed', we can use any number of metrics to get a feel for how different the two resources are. In [34] we used shingling [8] to measure the difference between text-based resources. We do not apply any change metric in this paper.

Figure 1 shows that resources A, G and E were recovered and are identical to their lost counterparts. An older version of B was found (B') that pointed to G, a resource that does not currently exist in $W$. Since B' does not reference D, we did not know to recover it. It is possible that G is actually D renamed, but we do not test for this. An older version of C was found, and although it still references F, F could not be found in any web repository.

A measure of change between the original website $W$ and the reconstructed website $W'$ can be described using the following **difference vector**:

$$\text{difference}(W, W') = \left( \frac{R_{changed}}{|W|}, \frac{R_{missing}}{|W|}, \frac{R_{added}}{|W'|} \right) \quad (1)$$

For Figure 1, the difference vector is (2/6, 1/6, 1/5) = (0.333, 0.167, 0.2). The best case scenario would be (0,0,0), the complete reconstruction of a website. A completely unrecoverable website would have a difference vector of (0,1,0).

The difference vector for a reconstructed website can be illustrated as a **reconstruction diagram** as shown on the right side of Figure 1. The changed, identical and missing resources form the core of the reconstructed website. The dark gray portion of the core grows as the percentage of changed resource increases. The hole in the center of the core grows as the percentage of missing resources increases. The added resources appear as crust around the core. This representation will be used later in Table 1 when we report on the websites we reconstructed in our experiments.

## 3.3 Web Repository Requests

In order for Warrick to reconstruct a website, it must make numerous requests for resources from web repositories. The minimum requirement for a web repository is that it has the functionality to return a stored resource when given the complete URL of where the resource exists (or did exist) on the Web. The resource returned may be the canonical resource or a modified version of the resource. IA always returns canonical versions, but search engines usually return HTML versions of resources (e.g., for PDF, PS, DOC, etc.) or thumbnail images. Warrick will choose canonical versions of resources over non-canonical versions, and if all competing versions are canonical, it will choose the most recent version. It is important that a website be reconstructed soon after it disappears since the search engines are often quick to purge cached resources that are no longer accessible on the Web.

Warrick 'respects' web repositories by issuing a limited number of requests per IP address in a 24 hour period. Warrick uses the APIs provided by all three search engines which each allow a limited number of daily queries (IA does not have an API). Warrick makes a maximum of 1000, 1000, 10,000 and 5000 daily requests from a single IP address to IA, Google, MSN and Yahoo, respectively. Reconstruction halts for 24 hours once any of the daily requests are exhausted. The limited number of daily requests causes the largest bottleneck to website reconstruction; the response times from the individual web repositories is inconsequential.

The first version of Warrick used the public web interface for issuing requests, but we are now using the search engine APIs because they are unlikely to change (or will likely change at a slower rate) than the HTML-formatted search results produced by the public web interface. In the case of Google we had no choice: they have recently begun to deny requests from an IP address for hours at a time if they suspect automated requests are being made through their public interface [33].

Web repositories have different access mechanisms to extract their resources. Google may be queried for a cached resource using `cache:http://foo.org/` or directly through an API function. Yahoo and MSN may be queried using `url:http://foo.org/`, and the cached URL can be extracted from the returned page. Yahoo and MSN also provide the cached URL directly through an API function call. IA can be queried using a URL of the form `http://web.archive.org/web/*/http://foo.org/`, and the returned page can be scraped for links to stored versions.

Google and Yahoo have different access mechanisms for extracting images. Google does not support image searching through their API, but Yahoo does. Google Images can be searched with a URL like so: `foo.org/img.gif`. The returned page can be scraped for a link pointing to the thumbnail image. Yahoo does not support direct URL queries, but it can be queried using `site:foo.org` with the image name (minus the file extension), and the results can be matched against the `foo.org/img.gif` to see if it is found. MSN does not support image searching, and MSN Images cannot be reliably queried to produce a missing image. IA has the same interface for extracting all resources.

## 3.4 Lister Queries

Warrick reconstructs websites by starting at a seed URL. It fetches the resource (identified by its URL) from each web repository, stores the canonical or most recent version of the resource and then examines the resource (if HTML) for links to additional resources. New links are added to the crawl frontier (a queue), and reconstruction continues until the frontier is empty. Warrick recovers resources in breadth-first order, the order in which it encounters links; breadth-first crawling of the Web results in finding high-quality pages early in the crawl [38]. A more detailed discussion of the algorithm can be found in [34].

The first version of Warrick did not know in advance if a repository contained a resource or not. It would first issue a request to see if the resource was stored, and if it was stored then another request would be issued to retrieve the resource. We could say that Warrick was naïve in that it made requests for resources assuming the repository had at least some portion of them stored. If a repository did not have any relevant resources stored, Warrick naïvely made numerous wasted requests to the repository anyway. For example, if a website to be reconstructed had 1000 resources, Warrick might make 1000 requests to a single repository that did not have any of the website's resources. This is problematic considering the limited number of repository requests that can be made per 24 hours and the fact that reconstruction is a race against the clock to extract cached search engine resources before they are removed. If Warrick could know in advance that a repository did not have any

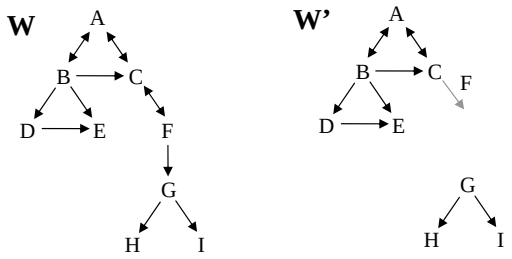


**Figure 2: Website W (left) and reconstructed website W' (right) that is missing resource F.**

resources stored, it could avoid making numerous wasted requests. Additionally, if it knew the resources that a web repository did have stored in advance, it could make requests only for stored items.

Fortunately, all four web repositories allow queries which list all the URLs that are stored for a particular website. For clarity, we will call these *lister queries*.

All three search engines support lister queries by using the 'site:' parameter. Further granularity for subsites can be obtained by using the 'allinurl:' parameter for Google and 'inurl:' for MSN and Yahoo. For example, the query '`site:foo.org inurl:bar`' will list all the cached web pages in MSN for `foo.org/~bar/*` (although some cached resources like `foo.org/~barabc` may also appear in the listing). IA also supports lister queries using specially-formatted URLs like this: `http://web.archive.org/web/*sr_0nr_20/http://foo.org/~bar/`.

There are two primary advantages of using lister queries:

1. discovery of URLs that would not be found by just examining links in recovered pages, and

2. avoiding unnecessary queries to a web repository that is known not to have a particular resource.

The first advantage is especially useful when a page that links to many more pages cannot be recovered. For example, if the website W in Figure 2 were to be reconstructed without lister queries, and resource F could not be recovered, it would not be possible to recover G, H and I since no link would exist to those resources from any recovered resources. But if a lister query revealed resources G, H and I, we would be able to recover them as shown on the right of Figure 2, even though they are not directly connected to W'.

The second advantage, saved queries, will allow us to reconstruct websites at a faster pace. This is beneficial when we need to recover resources from a search engine cache that may soon be purged.

Lister queries may produce links to web pages that are no longer part of the website. For example, occasionally a webmaster may remove a link to an old resource without also removing the old resource from the web server. If the resource has been indexed by a search engine, it will likely keep the resource indexed until it is removed from the server (much to the chagrin of unsuspecting college professors who merely remove links to old exam keys on their web site but leave the document on their web server). IA will keep the resource indefinitely, even if it is removed from the web server. Although the resources may no longer be part of the latest

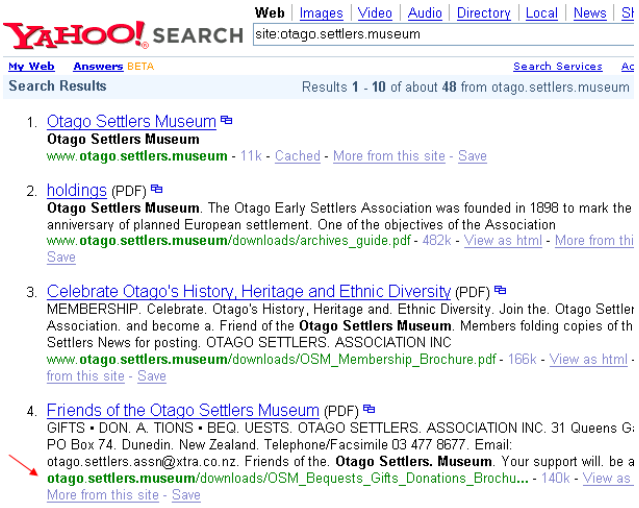


**Figure 3: Yahoo 'site:otago.settlers.museum' search results.**

version of the website, they could prove useful in rebuilding missing resources.

## 3.5 URL Canonicalization

Using direct URL queries presents a number of difficulties to a web-repository crawler because each of the web repositories may perform URL canonicalization (or URL normalization [7, 28, 42]) in different ways. Lister queries are useful for resolving some of the problems created by repositories using different canonicalization policies.

### 3.5.1 'www' Prefix

Some websites provide two URL variations to access their websites, one with a 'www' prefix and one without. For example, the website `otago.settlers.museum` may be accessed as `http://otago.settlers.museum/` or `http://www.otago.settlers.museum/`. Many websites will redirect users and web crawlers from one version to the other using an http 301 (Moved Permanently) status code. Google and other search engines request this behavior to simplify web crawling [15]. Google, MSN and IA may be queried with either version of the URL successfully, but Yahoo will fail to recognize a request for `http://foo.org/` if they crawled `http://www.foo.org/`.

Lister queries will reveal if a web repository stores a resource using the 'www' prefix or not as the Yahoo query for `site:otago.settlers.museum` illustrates in Figure 3. Here we see Yahoo using `www.otago.settlers.museum` for results 1-3 and `otago.settlers.museum` for result 4. A web-repository crawler may normalize all the URLs under one common host.

### 3.5.2 Case Insensitivity

Web servers housed on a case-insensitive filesystem like Windows will allow URLs to be accessed case-insensitively. Therefore the URLs `http://foo.org/bar.html` and `http://www.foo.org/BAR.html` are accessing the same resource on a case-insensitive web server. Google, Yahoo and IA index all URLs they crawl by the case of the URL and do not take into account the underlying filesystem. Therefore if

they are asked if they have the URL `http://www.foo.org/BAR.html` stored, they will reply 'no' when they actually have the URL indexed as `http://foo.org/bar.html`. MSN does not care about case sensitivity of URLs when queried.

Lister queries reveal the case of the URLs a repository has stored. If a web-repository crawler knows in advance that a website was housed on a case-insensitive web server, it can convert all URLs found by lister queries and by page scraping into lowercase so case is no longer an issue.

#### 3.5.3 Missing the Terminating Slash

When scraping a page for URLs to recover, some URLs that point to a directory may lack the proper terminating slash. For example, a URL that points to directory `abc` should end with a slash like so: `http://foo.org/abc/`, but the URL may appear instead as `http://foo.org/abc`. This does not present a problem when accessing the document live on the Web since the web server will respond with a 301 (Moved Permanently), and the browser will transparently redirect the user to the correct URL. When reconstructing a missing website, the resource is not accessible on the Web, and there is no way to automatically know in advance if the URL refers to a directory or not. And although Google, MSN and IA all properly report URLs that end with a slash, Yahoo does not.

Lister queries are useful for reducing the missing slash problem. If a web repository like Google reports that it has a URL stored as `http://foo.org/bar/` and Yahoo reports the URL stored as `http://foo.org/bar`, we may infer that the URL is pointing to a directory since we know Google's canonicalization policy dictates proper terminal slashes for directories. But if Yahoo is the only repository storing the resource `http://foo.org/bar`, we must arbitrarily decide to either treat the URL as a directory or not.

#### 3.5.4 Root Level URL Ambiguity

When a web server receives a request for a URL pointing to the root level of a directory, it will respond with any number of resources depending on its configuration. For example, `http://foo.org/bar/` may be accessing index.html, default.htm, index.cgi or any number of resources. When recovering a URL pointing to the root level of a directory, we cannot automatically discover the name of the file the URL is pointing to. When we recover a link pointing to `http://foo.org/bar/index.html` directly, we may infer that `http://foo.org/bar/` is referring to index.html, but we cannot be certain.

Web repositories canonicalize root level URLs differently as well. If Google and Yahoo are queried with index.html appended to a root level URL, they will reply with a 'found' response when such a URL does not really exist. MSN takes a different approach; if `http://foo.org/bar/index.html` is crawled and they do not encounter the URL `http://foo.org/bar/` in their crawl, a request for the later URL will result in a 'not found' response.

Lister queries can reduce this problem. If MSN reports that it has a URL stored as `http://foo.org/bar/index.html`, we may assume that this URL refers to `http://foo.org/bar/` (although this is not *always* true).

### 3.6 Crawling Policies

The ability to perform lister queries allows us to reconstruct websites using one of three crawling policies:

1. **Naïve Policy** - Do not issue lister queries and only recover links that are found in recovered pages.
2. **Knowledgeable Policy** - Issue lister queries but only recover links that are found in recovered pages.
3. **Exhaustive Policy** - Issue lister queries and recover all resources found in all repositories.

The naïve policy was used exclusively by the first version of Warrick. The knowledgeable and exhaustive policies have since been implemented. The crawling policy may be specified before a reconstruction begins.

## 4. CRAWLING POLICIES EXPERIMENT

### 4.1 Experiment Methodology

To better understand how the naïve, knowledgeable and exhaustive crawling policies affect our website reconstructions, we downloaded the 24 websites from our previous paper on website reconstruction [34]. The downloads served as baseline comparisons with our reconstructions. Although some of the websites are actually subsites, we will refer to them as websites for simplicity. We manually selected the 24 sites because they varied in size, subject, MIME types and top-level domains, and they did not use robots.txt files to keep search engines from crawling their pages.

We began the crawl of each site at the base URL shown in Table 1 and downloaded all images, style sheets, JavaScripts, etc. that a web crawler could find by following links. We did not limit the path depth of the crawls. For simplicity, we restricted the download to port 80 and to only resources that were in and beneath the starting directory. We did not follow links to other hosts within the same domain name. So if the base URL for the website was `http://www.foo.edu/abc/`, only URLs matching `http://www.foo.edu/abc/*` were downloaded. This is consistent with the default settings used by Warrick for reconstructing websites.

After downloading each website, we immediately started three concurrent reconstructions using each of the crawling policies. We began our downloads and reconstructions in late February 2006 and used six servers running Solaris, each with their own IP address.

### 4.2 Reconstruction Results

The downloads and reconstructions took 14 days to complete. Much of the delay was due to running out of daily requests to IA and Google, the two web repositories with the lowest quota of daily requests.

The complete results are shown in Table 1 ordered by total URIs (number of resources downloaded). The makeup of each downloaded website (percentage of HTML, images and other MIME types) are listed in the 3rd, 4th and 5th columns. The difference vector and reconstruction diagram are given for each reconstruction using the three crawling policies.

Website number 1 (`www.techlocker.com`) went out of business several months before we reconstructed it. The root page no longer had links to other portions of the website, so downloading the website resulted in only 1 file. We will discuss this reconstruction in more detail in Section 5.1.

Table 2 contains the descriptive statistics of several important factors of the website reconstructions. The percentage of recovered resources are for those resources that share

**Table 1: Results of Website Reconstructions**

| Website | URIs | Html | Images | Other | Naïve | Knowledgeable | Exhaustive |
|---|---|---|---|---|---|---|---|
| 1. www.techlocker.com | 1 | 100.0% | 0.0% | 0.0% | (0.000, 0.000, 0.000)  | (0.000, 0.000, 0.000)  | (0.000, 0.000, 1.000)  |
| 2. www.harding.edu/hr | 50 | 30.0% | 10.0% | 60.0% | (0.720, 0.220, 0.328)  | (0.640, 0.280, 0.163)  | (0.620, 0.240, 0.591)  |
| 3. www.smoky.ccsd.k12.co.us | 57 | 21.1% | 50.9% | 28.1% | (0.298, 0.509, 0.000)  | (0.298, 0.509, 0.000)  | (0.316, 0.491, 0.970)  |
| 4. www.genesis427.com | 65 | 15.4% | 76.9% | 7.7% | (0.508, 0.077, 0.016)  | (0.523, 0.077, 0.016)  | (0.538, 0.062, 0.500)  |
| 5. englewood.k12.co.us/schools/clayton | 77 | 39.0% | 57.1% | 3.9% | (0.247, 0.286, 0.000)  | (0.247, 0.286, 0.000)  | (0.247, 0.286, 0.304)  |
| 6. www.raitinvestmenttrust.com | 79 | 30.4% | 57.0% | 12.7% | (0.228, 0.127, 0.014)  | (0.278, 0.253, 0.033)  | (0.253, 0.127, 0.859)  |
| 7. otago.settlers.museum | 120 | 25.0% | 70.8% | 4.2% | (0.208, 0.525, 0.017)  | (0.208, 0.542, 0.286)  | (0.208, 0.533, 0.341)  |
| 8. www.usamriid.army.mil | 121 | 36.4% | 60.3% | 3.3% | (0.397, 0.413, 0.220)  | (0.364, 0.512, 0.253)  | (0.364, 0.471, 0.880)  |
| 9. www.mie2005.net | 136 | 6.6% | 62.5% | 30.9% | (0.699, 0.199, 0.009)  | (0.801, 0.096, 0.000)  | (0.801, 0.096, 0.335)  |
| 10. searcy.dina.org | 164 | 59.1% | 38.4% | 2.4% | (0.128, 0.049, 0.071)  | (0.128, 0.049, 0.077)  | (0.134, 0.037, 0.325)  |
| 11. www.cookinclub.com | 216 | 31.9% | 67.6% | 0.5% | (0.565, 0.037, 0.140)  | (0.556, 0.037, 0.148)  | (0.560, 0.032, 0.790)  |
| 12. www.gltron.org | 306 | 6.9% | 71.2% | 21.9% | (0.284, 0.180, 0.004)  | (0.301, 0.183, 0.035)  | (0.301, 0.183, 0.260)  |
| 13. privacy.getnetwise.org | 326 | 66.0% | 14.7% | 19.3% | (0.021, 0.479, 0.306)  | (0.092, 0.475, 0.296)  | (0.261, 0.307, 0.321)  |
| 14. www.americancaribbean.com | 329 | 17.6% | 80.9% | 1.5% | (0.380, 0.450, 0.005)  | (0.368, 0.450, 0.005)  | (0.368, 0.450, 0.689)  |
| 15. www.eskimo.com/~scs | 357 | 94.4% | 5.6% | 0.0% | (0.008, 0.006, 0.508)  | (0.008, 0.008, 0.509)  | (0.014, 0.006, 0.834)  |
| 16. www.digitalpreservation.gov | 389 | 85.1% | 2.8% | 12.1% | (0.015, 0.946, 0.000)  | (0.653, 0.321, 0.612)  | (0.643, 0.308, 0.898)  |
| 17. www.aboutfamouspeople.com | 396 | 58.1% | 41.9% | 0.0% | (0.705, 0.005, 0.088)  | (0.434, 0.005, 0.086)  | (0.419, 0.005, 0.844)  |
| 18. home.alltel.net/bsprowl | 474 | 34.6% | 65.4% | 0.0% | (0.004, 0.654, 0.006)  | (0.013, 0.808, 0.000)  | (0.055, 0.665, 0.006)  |
| 19. www.dpconline.org | 580 | 40.7% | 34.0% | 25.3% | (0.543, 0.209, 0.000)  | (0.450, 0.217, 0.032)  | (0.452, 0.214, 0.282)  |
| 20. www.cs.odu.edu/~pothen | 610 | 25.2% | 42.3% | 32.5% | (0.549, 0.067, 0.044)  | (0.485, 0.146, 0.048)  | (0.480, 0.152, 0.178)  |
| 21. www.mypyramid.gov | 646 | 47.8% | 39.2% | 13.0% | (0.367, 0.327, 0.011)  | (0.291, 0.345, 0.002)  | (0.291, 0.344, 0.102)  |
| 22. www.financeprofessor.com | 673 | 48.9% | 44.6% | 6.5% | (0.184, 0.165, 0.147)  | (0.189, 0.080, 0.069)  | (0.215, 0.120, 0.511)  |
| 23. www.fishingcairns.com.au | 1181 | 22.4% | 77.2% | 0.4% | (0.439, 0.025, 0.000)  | (0.434, 0.040, 0.000)  | (0.411, 0.036, 0.197)  |
| 24. www.kruderdorfmeister.com | 2503 | 90.7% | 9.3% | 0.0% | (0.068, 0.916, 0.000)  | (0.068, 0.916, 0.000)  | (0.069, 0.914, 0.243)  |

**Table 2: Statistics for Website Reconstructions**

| Category | Pol | Mean | Median | Std | Min/Max |
|---|---|---|---|---|---|
| Recovered (%) | N | 71.4 | 79.6 | 27.5 | 5.4/100.0 |
| | K | 72.4 | 76.5 | 25.3 | 8.4/100.0 |
| | E | 74.7 | 80.2 | 23.6 | 8.6/100.0 |
| Added (%) | N | 8.1 | 1.3 | 13.3 | 0.0/50.8 |
| | K | 11.1 | 3.4 | 16.7 | 0.0/61.2 |
| | E | 51.1 | 42.1 | 30.8 | 0.6/100.0 |
| Total requests | N | 1711.7 | 1131 | 1580.5 | 6/4880 |
| | K | 710.9 | 368.5 | 714.1 | 48/2412 |
| | E | 1587.5 | 941.5 | 1481.1 | 180/5220 |
| Efficiency ratio (all resources) | N | 0.16 | 0.15 | 0.05 | 0.07/0.27 |
| | K | 0.41 | 0.39 | 0.14 | 0.21/0.67 |
| | E | 0.49 | 0.48 | 0.10 | 0.29/0.65 |
| Efficiency ratio (excluding added) | N | 0.15 | 0.15 | 0.06 | 0.04/0.27 |
| | K | 0.37 | 0.35 | 0.15 | 0.02/0.64 |
| | E | 0.22 | 0.23 | 0.14 | 0.00/0.50 |

the same URI as resources in the downloaded website; this does not include the number of 'added' resources. Although the exhaustive policy performed moderately better, all three crawling policies generally recovered the same number of resources. The naïve policy performed significantly worse than the other policies only once: when reconstructing site 16 (`www.digitalpreservation.gov`). This was due to a recent website redesign which had not yet been fully captured in the search engine caches.

The percentage of resources categorized as 'added' for the exhaustive policy (51.1%) averaged 40% more than the knowledgeable policy and 43% more than the naïve policy. As we would expect, the exhaustive policy recovers significantly more added resources because every resource stored in every web repository is recovered regardless if a link is found to the resource or not.

Table 2 also shows the total number of repository requests issued for each website reconstruction. The number of requests per reconstruction varied widely, but the knowledgeable policy (710.9) averaged less than half the number of requests as the exhaustive (1587.5) and naïve policies (1711.7).

A better gauge for comparing these policies is to examine each website reconstruction's **efficiency ratio**, the total number of recovered resources divided by the total number of issued repository requests. The most efficient reconstruction would result in one request per recovered resource, a 1.0 efficiency ratio.

The efficiency ratio for each website reconstruction is plotted in the top graph of Figure 4, and the distribution is plotted at the bottom. It is also useful to examine the efficiency ratios when added resources are not considered. Figure 5 shows the efficiency ratio (top) and distribution (bottom) for each crawling policy when added resources are not considered in the total recovered resources. The efficiency ratio descriptive statistics are shown at the bottom of Table 2.

We grouped each of the efficiency ratios into pairs and ran a Wilcoxon Matched-Pairs Signed Ranks test on each of the pairs. The tests revealed statistically significant differences ($p < 0.001$) between the crawling policies when all resources are included in the efficiency ratio. As we would expect, the naïve policy is least efficient at recovering resources because the crawler does not know in advance which resources a web repository has stored. The exhaustive policy was shown to be slightly more efficient than the knowledgeable policy, likely because lister queries produce 'false positives' for the

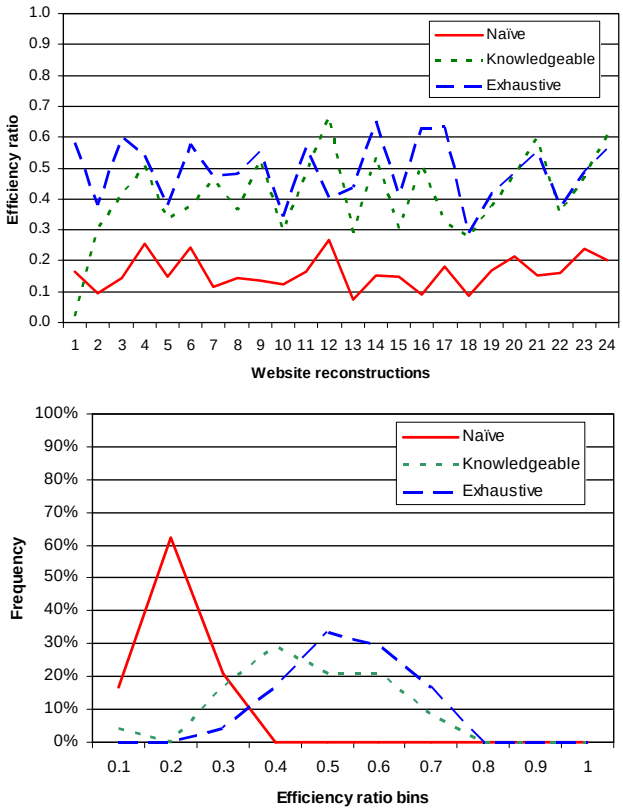


**Figure 4: Top: Efficiency ratio = total number of recovered resources divided by total number of repository requests. Bottom: Distribution of efficiency ratios where bin 0.1 corresponds to ratios with values between 0 and 0.1, etc.**

knowledgeable policy that are not recovered.

When we consider the efficiency ratio without added resources, we still see strong, statistically significant differences ($p < 0.001$) between the crawling policies with the exception of the (exhaustive, naïve) pair which still maintains a low p value ($p < 0.05$). When we exclude added resources, the knowledgeable policy performs the best. The exhaustive policy showed a 45% loss in the mean when compared to the efficiency ratio with all resources included. The naïve policy showed no appreciable difference.

## 4.3 Summary of Findings

Our experiments reveal several important characteristics about the three crawling policies. The naïve policy will recover nearly as many non-added resources as the knowledgeable and exhaustive policies, but at a huge expense in increased repository requests. This policy should be avoided if all the web repositories support lister queries.

The exhaustive policy will regularly recover significantly more added resources than the other two policies with a relatively high efficiency ratio. This may be desirable when reconstructing a website since the added resources may aid the human operator in manually re-creating missing resources. On the other hand, the added resources may contain outdated or useless information that is not useful for recon-

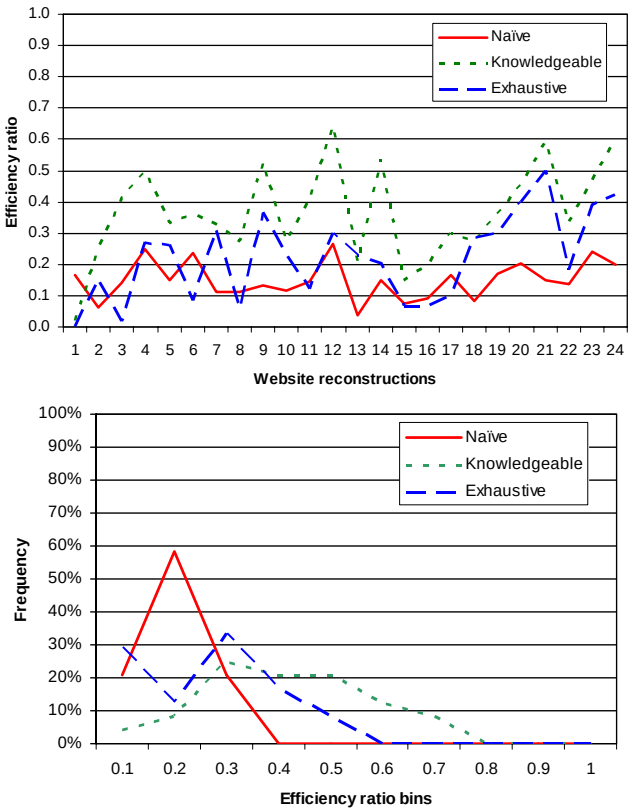


**Figure 5: Top: Efficiency ratio excluding 'added' resources. Bottom: Distribution of efficiency ratios excluding 'added' resources where bin 0.1 corresponds to ratios with values between 0 and 0.1, etc.**

structing the website, and the extra time spent recovering added resources may be significant for large websites.

The knowledgeable policy issues the fewest number of repository requests per reconstruction (less than half the number of requests of the other policies) and has the highest efficiency ratio when wanting to recover only non-added resources. Because fewer requests are issued with the knowledgeable policy, a website can be reconstructed much more quickly than if the other policies are used.

# 5. RECONSTRUCTING LOST WEBSITES

The experiments we have performed previously involved websites that had not yet been lost (with the exception of the TechLocker website). In this section we discuss various issues we have encountered while reconstructing three real websites that have been lost and discuss their implications for future work.

## 5.1 TechLocker Website

We were not aware that `www.techlocker.com` had disappeared until months afterwards when we ran our experiments. Using the naïve and knowledgeable policies, we were only able to recover a single HTML resource. The exhaustive policy recovered 2244 resources with 1952 of the resources coming from IA and 292 coming from Google. MSN and Yahoo had nothing cached. The Google resources were composed of 161 images and 131 HTML resources. Although we are not able to determine the cached date of images in Google, we were able to discover the HTML resources ranged from January to September 2005.

We used the original URLs of the recovered resources to see if they still resided on the TechLocker web server. Each request consistently resulted in a http 404 (not found) response. It may be surprising to some that resources over one year in age remain in the Google cache, but as our previous experiments have demonstrated [34], Google may make resources available in their cache long after the resources have been removed from a website. We have not observed this behavior from MSN or Yahoo.

## 5.2 WWW 2006 Conference Website

The 15th International World Wide Web 2006 Conference website (`www2006.org`) was lost after a fire destroyed the Mountbatten building at the University of Southampton (UK) in October 2005 [21]. The fire was approximately one week before long papers were to be submitted. We ran Warrick with the naïve policy (lister queries had not yet been implemented) and were able to recover 77 resources, 36 from Google, 3 from Yahoo and 38 from MSN. After contacting the conference organizers, we learned that they had used a script to recover some pages from Google's cache, but they had not thought to look for pages in MSN or Yahoo. Leslie Carr, one of the conference chairs, stated in an email correspondence: "Our problem was not with data recovery but with service sustainability. We *knew* that we would be able to reconstruct the data (at some point); the challenge was reconstructing enough data *now*." The web server was later recovered intact, and the website was restored a week later.

Several of the pages we recovered were produced dynamically by PHP. Web repositories are currently only storing the client's view of web pages, not the server-side logic used to generate the client view. In Section 6 we address how we might inject the server-side logic (scripts, data files, databases, etc.) into web pages that are stored by web repositories so the server-side files of a websites can also be reconstructed.

## 5.3 Internet Christian Library Website

In March 2006, we were approached by an individual who wanted to reconstruct a non-profit academic archival website (`www.iclnet.org`) which had been lost a month earlier when the private ISP which had been hosting the site ended their support. The website had been removed by the new ISP, and the same "domain renewal" page was returned whenever any of the website's original URLs were accessed. These soft 404 responses [5] caused the search engines to replace their cached pages with the domain-renewal pages as illustrated in Figure 6.

Although MSN's cache contained many soft 404 pages, it appeared that Google and Yahoo had only replaced the cache of the root page, `http://www.iclnet.org/`, with the soft 404 page; the remainder of the original pages remained cached. It is possible that the Google and Yahoo crawlers can detect soft 404 pages and will not cache more than one of them. Or they were just slow at re-crawling these pages. IA was the only web repository that did not have the soft 404 pages stored (it takes 6-12 months for their repository to be updated with the most recent Alexa crawls [26]).

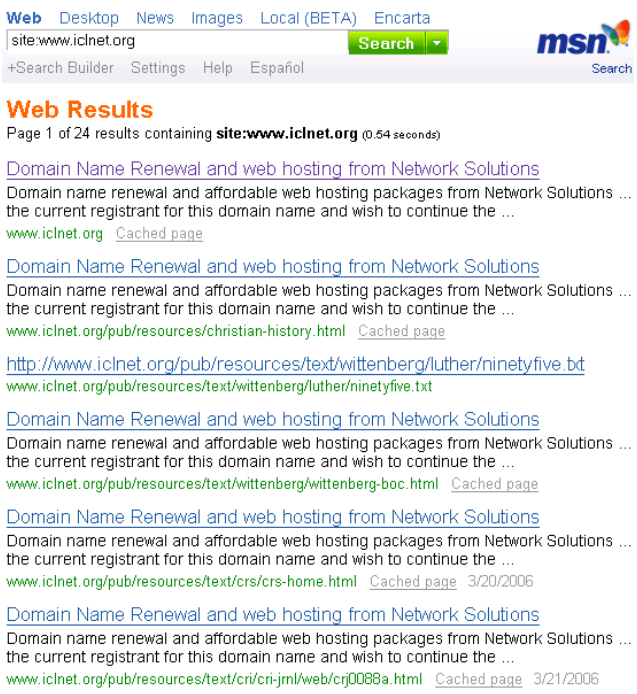


**Figure 6: MSN 'site:www.iclnet.org' search results contain "domain renewal" soft 404 pages.**

The algorithm used by Warrick to reconstruct websites looks for the most recent HTML page to save. When we reconstructed the ICL website using the knowledgeable crawling policy, Warrick chose the domain-renewal pages from MSN because they were the most recent. Because there were no links to follow to other resources, none of the original pages were restored with this policy. We then ran two reconstructions using the exhaustive policy: one using IA only and one using the three search engines. By separating IA pages from the search engine pages, we ensured that at least one reconstruction would be free of domain-renewal pages.

The IA-only reconstruction yielded 3744 recovered resources for the ICL website with 966 missing resources. The search-engine-only reconstruction yielded 1337 resources, mostly from Google; 5363 resources could not be recovered.

In the next version of Warrick we would like to incorporate automatic detection of soft 404s, possibly using the algorithm proposed in [5]. We would also like to give users the option to choose which of the resource versions for a single URL they would like to save.

## 6. FUTURE WORK

As mentioned in Section 5.2, we are interested in developing methods for injecting the server-side functionality of a website into the WI so it can be reconstructed. Not only do we want to recover the hypertext, we want to recover the generative functionality of the hypertext. We are investigating how we might use erasure codes [44] for encoding server-side components (programs, databases, etc.) and injecting them into HTML comments in indexable web pages. Erasure codes are frequently used in RAID systems where

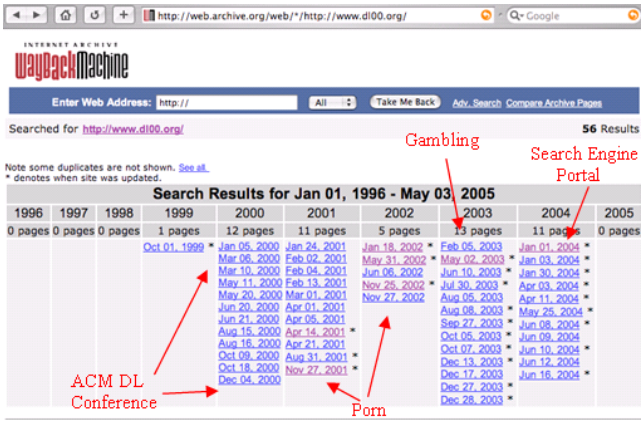


**Figure 7: IA versions of www.dl00.org web page.**

portions of files are distributed among multiple devices to prevent data loss [43]. If we were able to recover some subset of encoded pages from the WI, we could reconstruct the server files.

Another area for future work involves reconstructing a website when repository copies of a website are tainted. An interesting example is shown in Figure 7. The web page for the 2000 ACM Digital Library Conference (`www.dl00.org`) no longer points to the conference materials. The URL is still valid, but the website has been lost. The conference did not renew the domain registration, and in late 2001 the domain was "hijacked". In order to reconstruct this website, we would need to distinguish between the correct and tainted versions of a page. We are interested in applying trend analysis to detect significant shifts in the page "aboutness", similar to the techniques applied in the Walden's Path Project [16]. The Museum of E-Failure [4] is an excellent source of lost websites for testing purposes.

Reconstructing websites can be problematic for a highly dynamic environment where navigating or crawling the site changes the link structure (e.g., [31]). For these types of websites, links are always changing to exhibit navigational patterns, and the link URLs do not constitute as unique identifers for resources. Recovered resources with dynamic links will never be byte-for-byte identical to any original resources. We need to develop methods for more accurately assessing success when reconstructing such websites.

Finally, we are wanting to run a large-scale experiment reconstructing a larger sample of randomly chosen websites in order to discover those website properties that assist in WI preservation. In our previous study [34] we did not find a statistical correlation between website size, Google's PageRank and reconstruction success. We would like to develop guidelines that webmasters can use to ensure their websites are more successfully reconstructed from the WI.

## 7. CONCLUSIONS

We have discussed several of the challenges of using direct URL queries with web repositories. We have shown how lister queries can be used to negate some of the repository URL canonicalization problems.

We have presented three crawling policies for our web-repository crawler and evaluated the policies by download-

ing and reconstructing 24 websites. We found that the naïve policy was nearly as effective at recovering non-added resources as the knowledgeable and exhaustive policies but at a huge expense in increased repository requests. The exhaustive policy recovered significantly more added resources than the other two policies with a relatively high efficiency ratio. The knowledgeable policy was shown to be the most efficient when recovering only non-added resources, and it caused the crawler to issue the fewest number of repository requests per reconstruction.

We also shared our experiences reconstructing lost websites on behalf of others and examined some of the weaknesses of our current crawler. We are planning on expanding our work in website reconstruction from the WI to further evaluate its effectiveness as a preservation strategy. We hope our work will provide a useful public service for those who lose their websites when no backups are available.

## 8. ACKNOWLEDGEMENTS

We thank our anonymous reviewers for their valuable suggestions that were used to improve our paper.

## 9. REFERENCES

[1] A. Arasu, J. Cho, H. Garcia-Molina, A. Paepcke, and S. Raghavan. Searching the Web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, 2001.

[2] R. Baeza-Yates and C. Castillo. Characterization of national web domains. Technical report, Universitat Pompeu Fabra, 2005.

[3] R. Baeza-Yates, C. Castillo, M. Marin, and A. Rodriguez. Crawling a country: better strategies than breadth-first for web page ordering. In *Proceedings of WWW '05*, pages 864–872, 2005.

[4] S. Baldwin. Museum of e-failure, 2006. `http://disobey.com/ghostsites/mef.shtml`.

[5] Z. Bar-Yossef, A. Z. Broder, R. Kumar, and A. Tomkins. Sic transit gloria telae: towards an understanding of the web's decay. In *Proceedings of WWW '04*, pages 328–337, 2004.

[6] M. K. Bergman. The deep web: Surfacing hidden value. *The Journal of Electronic Publishing*, August 2001. `http://www.press.umich.edu/jep/07-01/bergman.html`.

[7] T. Berners-Lee, R. Fielding, and L. Masinter. Uniform Resource Identifier (URI): Generic syntax. RFC 3986, Jan. 2005.

[8] A. Z. Broder, S. C. Glassman, M. S. Manasse, and G. Zweig. Syntactic clustering of the Web. *Computer Networks and ISDN Systems*, 29(8-13):1157–1166, 1997.

[9] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: A new approach to topic-specific resource discovery. In *Proceedings of WWW '04*, 1999.

[10] J. Cho and H. Garcia-Molina. The evolution of the web and implications for an incremental crawler. In *Proceedings of VLDB '00*, pages 200–209, 2000.

[11] J. Cho and H. Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of SIGMOD '00*, pages 117–128, 2000.

[12] J. Cho and H. Garcia-Molina. Parallel crawlers. In *Proceedings of WWW '02*, pages 124–135, 2002.

[13] J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding replicated web collections. In *Proceedings of SIGMOD '00*, pages 355–366, 2000.

[14] V. Cothey. Web-crawling reliability. *Journal of the American Society for Information Science and Technology*, 55(14):1228–1238, 2004.

[15] M. Cutts. SEO advice: URL canonicalization. Jan 2006. `http://www.mattcutts.com/blog/seo-advice-url-canonicalization/`.

[16] Z. Dalal, S. Dash, P. Dave, L. Francisco-Revilla, R. Furuta, U. Karadkar, and F. Shipman. Managing distributed collections: evaluating web page changes, movement, and replacement. In *Proceedings of JCDL '04*, pages 160–168, 2004.

[17] M. Diligenti, F. Coetzee, S. Lawrence, C. L. Giles, and M. Gori. Focused crawling using context graphs. In *Proceedings of VLDB '00*, pages 527–534, 2000.

[18] J. Edwards, K. McCurley, and J. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of WWW '01*, pages 106–113, 2001.

[19] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In *Proceedings of WebDB '04*, pages 1–6, 2004.

[20] D. Fetterly, M. Manasse, and M. Najork. Detecting phrase-level duplication on the World Wide Web. In *Proceedings of ACM SIGIR '05*, pages 170–177, 2005.

[21] Fire destroys top research centre. Oct 31, 2005. `http://news.bbc.co.uk/2/hi/uk_news/england/hampshire/4390048.stm`.

[22] D. Gomes and M. J. Silva. Characterizing a national community web. *ACM Transactions on Internet Technology (TOIT)*, 5(3):508–531, 2005.

[23] Google Sitemap Protocol, 2005. `http://www.google.com/webmasters/sitemaps/docs/en/protocol.html`.

[24] Y. Hafri and C. Djeraba. High performance crawling system. In *Proceedings of MIR '04*, pages 299–306, 2004.

[25] T. L. Harrison and M. L. Nelson. Just-in-time recovery of missing web pages. In *Proceedings of HYPERTEXT '06*, Aug 2006.

[26] Internet Archive FAQ: How can I get my site included in the Archive?, 2006. `http://www.archive.org/about/faqs.php`.

[27] C. Lampos, M. Eirinaki, D. Jevtuchova, and M. Vazirgiannis. Archiving the Greek Web. In *Proceedings of the 4th International Web Archiving Workshop (IWAW'04)*, Sept 2004.

[28] S. H. Lee, S. J. Kim, and S. H. Hong. On URL normalization. In *ICCSA '05: Proceedings of the International Conference on Computational Science and Its Applications*, pages 1076–1085, June 2005.

[29] S. W. Liddle, D. W. Embley, D. T. Scott, and S. H. Yau. Extracting data behind web forms. In *Workshop on Conceptual Modeling Approaches for e-Business*, pages 402–413, Tampere, Finland, Oct 2002.

[30] S. W. Liddle, S. H. Yau, and D. W. Embley. On the automatic extraction of data from the hidden web. In *International Workshop on Data Semantics in Web Information Systems (DASWIS-2001)*, pages 212–226,

Yokohama, Japan, Nov 2001.

[31] T. Lutkenhouse, M. L. Nelson, and J. Bollen. Distributed, real-time computation of community preferences. In *Proceedings of HYPERTEXT '05*, pages 88–97, 2005.

[32] C. C. Marshall and G. Golovchinsky. Saving private hypertext: requirements and pragmatic dimensions for preservation. In *Proceedings of HYPERTEXT '04*, pages 130–138, 2004.

[33] F. McCown. Google is sorry. Jan 2006. `http://frankmccown.blogspot.com/2006/01/google-is-sorry.html`.

[34] F. McCown, J. A. Smith, M. L. Nelson, and J. Bollen. Reconstructing websites for the lazy webmaster. Technical report, Old Dominion University, 2005. `http://arxiv.org/abs/cs.IR/0512069`.

[35] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz. Evaluating topic-driven web crawlers. In *Proceedings of SIGIR '01*, pages 241–249, 2001.

[36] G. Mohr, M. Kimpton, M. Stack, and I. Ranitovic. Introduction to Heritrix, an archival quality web crawler. In *Proceedings of the 4th International Web Archiving Workshop (IWAW'04)*, Sept 2004.

[37] S. Mukherjea. Organizing topic-specific web information. In *Proceedings of HYPERTEXT '00*, pages 133–141, 2000.

[38] M. Najork and J. L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of WWW '01*, pages 114–118, 2001.

[39] M. L. Nelson, H. Van de Sompel, X. Liu, T. L. Harrison, and N. McFarland. mod_oai: An Apache module for metadata harvesting. In *Proceedings of ECDL 2005*, 2005.

[40] A. Ntoulas, P. Zerfos, and J. Cho. Downloading textual hidden web content through keyword queries. In *Proceedings of JCDL '05*, pages 100–109, 2005.

[41] E. T. O'Neill, B. F. Lavorie, and R. Bennett. Trends in the evolution of the public web. *D-Lib Magazine*, 3(4), April 2003.

[42] G. Pant, P. Srinivasan, and F. Menczer. "Crawling the Web". In *Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Edited by M. Levene and A. Poulovassilis*, pages 153–178. Springer-Verlag, 2004.

[43] J. S. Plank. A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. *Software: Practice and Experience*, 27(9):995–1012, 1997.

[44] M. O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, 36(2):335–348, 1989.

[45] S. Raghavan and H. Garcia-Molina. Crawling the hidden web. In *Proceedings of VLDB '01*, pages 129–138, 2001.

[46] M. A. Serrano, A. Maguitman, M. Boguna, S. Fortunato, and A. Vespignani. Decoding the structure of the WWW: facts versus sampling biases. Technical report, 2006. `http://www.arxiv.org/abs/cs.NI/0511035`.

[47] V. Shkapenyuk and T. Suel. Design and implementation of a high-performance distributed web crawler. In *Proceedings of the IEEE International Conference on Data Engineering (ICDE)*, volume 60, pages 357–368. IEEE Computer Society, 2002.

[48] K. Sigurosson. Incremental crawling with Heritrix. In *Proceedings of the 5th International Web Archiving Workshop (IWAW'05)*, Sept 2005.

[49] J. A. Smith, F. McCown, and M. L. Nelson. Observed web robot behavior on decaying web subsites. *D-Lib Magazine*, 12(2), Feb 2006.

[50] D. Waters and J. Garrett. Preserving digital information: Report of the task force on archiving of digital information. Technical report, 1996. `http://www.rlg.org/ArchTF/`.

[51] What are Google's design and technical guidelines? `http://www.google.com/support/webmasters/bin/answer.py?answer=35770`.

[52] J. L. Wolf, M. S. Squillante, P. S. Yu, J. Sethuraman, and L. Ozsen. Optimal crawling strategies for web search engines. In *Proceedings of WWW '02*, pages 136–147, 2002.