LAZY PRESERVATION: RECONSTRUCTING WEBSITES FROM THE WEB INFRASTRUCTURE

by

Frank McCown
B.S. 1996, Harding University
M.S. 2002, University of Arkansas at Little Rock

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

COMPUTER SCIENCE

OLD DOMINION UNIVERSITY
December 2007

ABSTRACT

LAZY PRESERVATION: RECONSTRUCTING WEBSITES FROM THE WEB INFRASTRUCTURE

Frank McCown
Old Dominion University, 2007
Director: Dr. Michael L. Nelson

Backup or preservation of websites is often not considered until after a catastrophic event has occurred. In the face of complete website loss, webmasters or concerned third parties have attempted to recover some of their websites from the Internet Archive. Still others have sought to retrieve missing resources from the caches of commercial search engines. Inspired by these post hoc reconstruction attempts, this dissertation introduces the concept of *lazy preservation*– digital preservation performed as a result of the normal operations of the Web Infrastructure (web archives, search engines and caches). First, the Web Infrastructure (WI) is characterized by its preservation capacity and behavior. Methods for reconstructing websites from the WI are then investigated, and a new type of crawler is introduced: the *web-repository crawler*. Several experiments are used to measure and evaluate the effectiveness of lazy preservation for a variety of websites, and various web-repository crawler strategies are introduced and evaluated. The implementation of the web-repository crawler Warrick is presented, and real usage data from the public is analyzed. Finally, a novel technique for recovering the generative functionality (i.e., CGI programs, databases, etc.) of websites is presented, and its effectiveness is demonstrated by recovering an entire Eprints digital library from the WI.

To my wife, Becky.

ACKNOWLEDGMENTS

There are a number of people who I would like to acknowledge for their support during my doctoral work. I would especially like to thank my advisor, Michael L. Nelson, for the time and effort he put into mentoring me these past several years. Our many discussions sparked a number of great ideas and helped turn several dead-ends into possibilities. I am also grateful to my doctoral committee and for the input they have provided.

Much of this dissertation is the product of collaboration with a number of excellent researchers. Michael Nelson and Johan Bollen (LANL) provided many of the initial ideas on lazy preservation and the Web Infrastructure. Other ideas about lazy preservation developed from collaboration with Cathy Marshall (Microsoft). Joan Smith worked with me on the decaying website experiment from Chapter IV and provided helpful ideas and advice throughout my time at ODU. Giridhar Nandigam helped with the search engine sampling experiment in Chapter IV, Amine Benjelloun developed most of the Brass system from Chapter VI, and Norou Diawara helped perform the statistical analysis in Chapter VII.

I would like to thank Janet Brunelle, Hussein Abdel-Wahab and other ODU faculty for their friendship and encouragement these past several years. I enjoyed attending a number of conferences with Michael, Johan, Joan, Martin Klein, Marko Rodriguez (LANL) and Terry Harrison (CACI) and learned a lot from our many discussions. Thanks also to members of the Systems Group who kept our infrastructure running smoothly. My friends and colleagues at Harding University were very supportive of me while I worked on my Ph.D., and I am thankful to them for providing me the time off to pursue my doctorate degree.

This dissertation is dedicated to my wife Becky who cheered me on through the good times, encouraged me when I was down and wanted to quit, and made me laugh every day. I could not have had a more supportive spouse. I especially enjoyed becoming a father in the final year of my doctoral work– much of this dissertation was written only a few feet away from Ethan as he slept and played. I thank my parents Andy and Genia and my siblings John and Sara who have encouraged me in all my pursuits, and I thank my family at the Bayside Church of Christ who provided friendship and spiritual guidance while we lived in Virginia. Finally and primarily, I thank God for giving me the strength to finish what I started, and, in the spirit of 1 Corinthians 10:31, I hope that this dissertation glorifies Him.

“ ‘My son,’ the father said, ‘you are always with me,
and everything I have is yours.
But we had to celebrate and be glad,
because this brother of yours was dead and is alive again;
he was lost and is found.’ ”
- Luke 15:31

TABLE OF CONTENTS

Page

LIST OF TABLES ix

LIST OF FIGURES x

Chapter

I INTRODUCTION 1
- 1 MOTIVATION 1
- 2 OBJECTIVE 3
- 3 APPROACH 3
- 4 ORGANIZATION 4

II PRESERVING THE WEB 6
- 1 LINK ROT 6
- 2 PRESERVING THE WEB 8
- 3 WEB CRAWLING 11
- 4 PRESERVING WEBSITES 13
- 5 CONCLUSIONS 14

III LAZY PRESERVATION AND THE WEB INFRASTRUCTURE 15
- 1 LAZY PRESERVATION 15
- 2 LIMITATIONS 17
- 3 WEB REPOSITORIES 18
- 4 CONCLUSIONS 27

IV CHARACTERIZING THE WEB INFRASTRUCTURE 29
- 1 A MODEL FOR RESOURCE AVAILABILITY 29
- 2 WEB INFRASTRUCTURE PRESERVATION CAPABILITY 30
- 3 WEB INFRASTRUCTURE CONTENTS 38
- 4 DISCUSSION 44
- 5 CONCLUSIONS 45

V WEB-REPOSITORY CRAWLING 46
- 1 CRAWLER ARCHITECTURE 46
- 2 LISTER QUERIES AND CRAWLING POLICIES 47
- 3 URL CANONICALIZATION 48
- 4 CONCLUSIONS 53

VI WARRICK, A WEB-REPOSITORY CRAWLER 54
- 1 BRIEF HISTORY 54
- 2 IMPLEMENTATION 55
- 3 OPERATION 56
- 4 RUNNING 58
- 5 BRASS 59
- 6 USAGE STATISTICS 64
- 7 CONCLUSIONS 65

VII EVALUATING LAZY PRESERVATION 67
- 1 WEBSITE DEFINITIONS 67

2 RECONSTRUCTION MEASUREMENTS 70
3 INITIAL RECONSTRUCTION EXPERIMENT 73
4 CRAWLING POLICIES EXPERIMENT 78
5 FACTORS AFFECTING WEBSITE RECONSTRUCTION 83
6 CONCLUSIONS 96

VIII RECOVERING A WEBSITE'S SERVER COMPONENTS 97
1 GENERATING DYNAMIC WEB CONTENT 97
2 WHAT TO PROTECT 98
3 INJECTION MECHANICS 98
4 EXPERIMENTS 103
5 DISCUSSION 113
6 CONCLUSIONS 113

IX CONCLUSIONS AND FUTURE WORK 114
1 CONCLUSIONS 114
2 CONTRIBUTIONS 114
3 FUTURE WORK 115

BIBLIOGRAPHY 117

APPENDICES

A WARRICK COMMAND-LINE SWITCHES 129

B RECONSTRUCTED WEBSITES 130

VITA 134

LIST OF TABLES

Table Page

1 Systems for preserving and recovering web resources. 14
2 Sample of reconstructed websites. 17
3 Web repository-supported data types as of July 10, 2007. 22
4 Implementation summary of web-repository interfaces. 24
5 Resource availability states. 30
6 Caching of HTML resources from four web collections. 36
7 Web and cache accessibility. 39
8 Indexed and cached content by type. 40
9 Staleness of search engine caches (in days). 42
10 Search engine overlap with the Internet Archive. 44
11 Repository request methods and limits. 55
12 Brass usage statistics from 2007. 64
13 Brass recovery summary. 65
14 Repository use, contributions and requests. 66
15 General levels of reconstruction success. 73
16 Results of initial website reconstructions. 76
17 Results of crawling-policy website reconstructions (Part 1). 79
18 Results of crawling-policy website reconstructions (Part 2). 80
19 Statistics for crawling-policy website reconstructions. 80
20 Descriptive statistics for reconstruction success levels. 86
21 Reconstruction performance of web repositories. 91
22 Regression parameter estimates. 95
23 Various r values (bold). 102
24 Ten reconstructed digital libraries. 104
25 Composition of Monarch DL. 105
26 Updates and changes made throughout the experiment. 108
27 Recovered server files and repository contributions each week. 112
28 Warrick command-line switches. 129
29 Listing of 309 websites. 130
30 Listing of 309 websites continued. 131
31 Listing of 309 websites continued. 132
32 Listing of 309 websites continued. 133

LIST OF FIGURES

Figure Page

1 Refreshing and migrating occurring in the living Web. 2
2 Publisher's cost and Web coverage of preservation systems/mechanisms. 16
3 Google provides "Cache" and "View as HTML" links to cached resources. 19
4 Original PDF and the HTML cached versions. 21
5 Google's cached thumbnail of http://www.cs.odu.edu/files/ecsbdg.jpg. 21
6 Google error screen returned when automated queries are detected. 23
7 Google query for `site:www.cs.odu.edu` returning "about 34,600" results. 25
8 Google query for `cache:http://www.cs.odu.edu/` returning the cached page as it was crawled on July 7, 2007. 26
9 Google Image query for `site:www.cs.odu.edu` returning "about 1,000" results. . . . 26
10 IA response to http://web.archive.org/web/*sr_0nr_20/http://www.cs.odu.edu/*. . . 27
11 IA response to http://web.archive.org/web/*/http://www.cs.odu.edu/. 28
12 Timeline of search engine resource acquisition and release. 29
13 Number of resources in web collection. 32
14 Example PDF resource from synthetic collection. 34
15 Crawling (top) and caching (bottom) of HTML resources from the mln web collection. 37
16 Distribution of TLDs from sample. 40
17 Distribution of Web file sizes (top) and cached file sizes (bottom) on log-log scale. Web file size means: Ask = 88 KB, Google = 244 KB, MSN = 204 KB, Yahoo = 61 KB. Cached file size means: Ask = 74 KB, Google = 104 KB, MSN = 79 KB, Yahoo = 34 KB. 41
18 Distribution of staleness on log-log scale. 43
19 Scatterplots of similarity vs. staleness (x-axis is on a log scale). 43
20 Venn diagram showing overlap of search engine caches with IA. 44
21 Distribution and hit rate of sampled URLs with IA. 45
22 Architecture of a traditional web crawler (left) and web-repository crawler (right). . 46
23 Website W (left) and reconstructed website W' (right) that is missing resource F. . . 48
24 Architecture of a web-repository crawler using lister queries. 48
25 Yahoo search results for `site:otago.settlers.museum`. 50
26 Warrick's algorithm for reconstructing a website. 57
27 Reconstruction summary file. 58
28 Screenshot of Brass's submit job interface. 60
29 User checking status of current job. 61
30 Screenshot of Brass's administrative interface. 62
31 Job information stored in XML. 63
32 Distribution of websites by percentage of recovered resources. 65
33 Brass recovered websites by TLD. 66
34 IA versions of http://www.dl00.org/. 69
35 The website www.techlocker.com remains accessible, but the previous content is lost. 69
36 The domain name markfoley.com is awaiting re-purchase. 70
37 Lost website (left), reconstructed website (center), and reconstruction diagram (right). 70
38 Comparing a crawled website (left) to its reconstruction (right). 74
39 Recovery success by MIME type. 75
40 Web repositories contributing to each website reconstruction. 77
41 Number of queries performed and time taken to reconstruct websites. 78
42 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. 82

43 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. 82
44 Success of reconstructions by week. 85
45 Distribution of s2 to s5 success. 86
46 Distribution and recovery of websites based on ratio of textual resources. 87
47 Distribution and recovery by TLD. 88
48 Fraction of resources from the first crawl still existing after *n* weeks (dark bars) and new resources (light bars). 89
49 Recovery category of HTML resources grouped by change rate. 89
50 Percentage of resources recovered by age. 90
51 Encoding of a file in HTML comments. 99
52 Injecting and recovering a server file into and from the WI. 100
53 Approaches to injecting server components into web pages. 101
54 Google Scholar maintains 11 versions of the same paper, some of which are cached. . 106
55 Monarch DL screen shot. 106
56 Encoding of a server file in HTML comments from the Monarch DL. 107
57 Recovered DL resources each week. 109
58 Percent of recovered DL resources each week. 110
59 Images from Monarch DL cached by Google. 110
60 Availability of Monarch DL resources. 111

CHAPTER I

INTRODUCTION

1 MOTIVATION

"My old web hosting company lost my site in its entirety (duh!) when a hard drive died on them. Needless to say that I was peeved, but I do notice that it is available to browse on the wayback machine... Does anyone have any ideas if I can download my full site?"
- A request for help at archive.org [147]

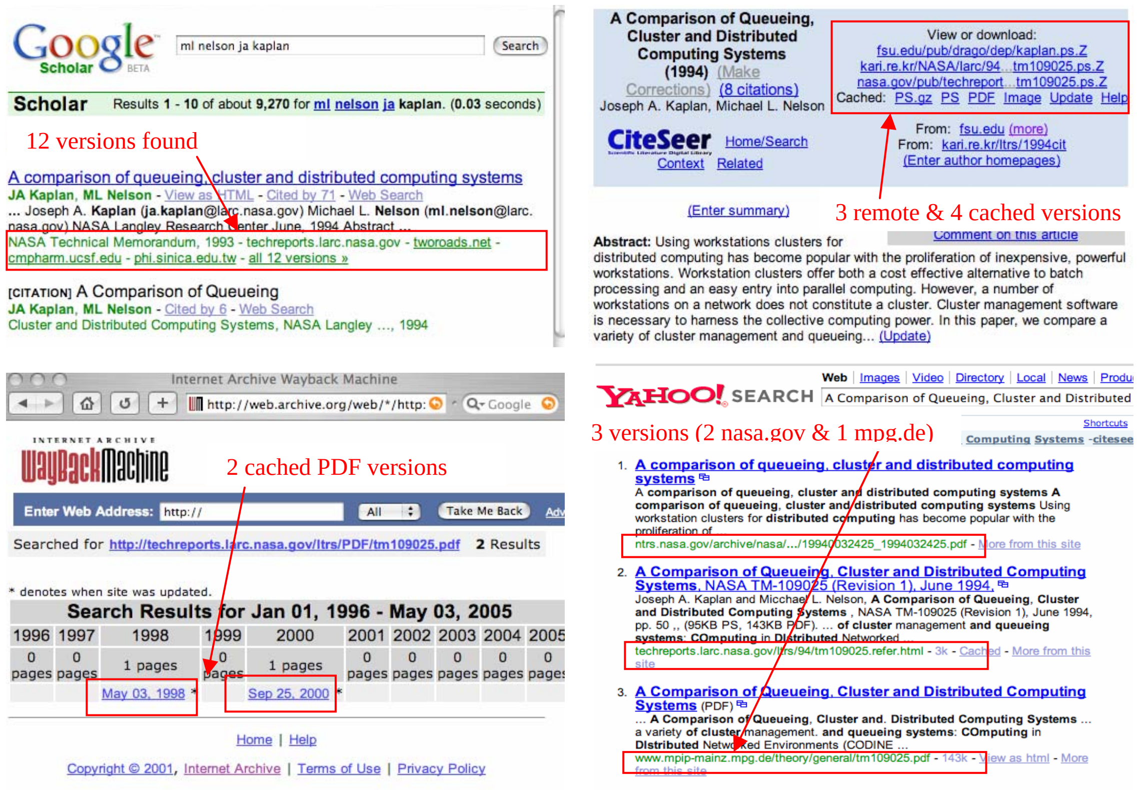
As the most popular publishing medium today, the Web has unleashed exponential growth of mankind's creative output, and along with it, an abundance of curatorial and preservation challenges[1]. Preserving this "global mess of previously unimagined proportions" [33] is a considerable challenge for the current generation, much less for future generations. Individuals do not recognize their personal responsibility for preserving personal data like websites: a 2006 survey indicates only 57% of individuals who store personal data on their computers ever backup their data [31], and even experts who work on backup storage techniques admit they do not backup their personal files [44]. Webmasters who rely on ISPs and web hosting companies to preserve their websites are often surprised and disappointed to discover that these organizations are not immune to viruses, hackers, bankruptcy and run-ins with the law [73, 105, 114]. Just one year ago, members of the Internet Archive's web team recovered over 200 websites for individuals who had lost their websites and did not have a functional backup [119].

Even if a great deal of personal care and energy are expended to properly ensure the availability of the individual's or organization's website, the website may still become defunct once it is no longer needed, financial pressures dictate ending the website or the maintainer dies [105]. In this case, interested third parties may have few means available to resurrect the website since they typically do not have access to backups.

To combat the ephemeral nature of the Web, preservationists may employ refreshing (copying of data to different systems), migration (transferring of data to newer system environments [171]) and emulation (replicating the functionality of an obsolete system [149]) to protect collections of known importance. Such strategies often involve a large institutional investment of time and money. For example, the Library of Congress recently funded a $13.9M project to preserve "digital content relating to important people, events and movements that have had a major impact on the nation's history [92]." Not only do these types of *in vitro* preservation projects have a limited scope and require significant effort within a controlled environment to be successful, it is often difficult to determine in advance what might be important in the future.

Other collections of unknown value may also be preserved although in a somewhat random and haphazard manner by the "living web." For example, consider a 1994 NASA report (Figure 1) that has been refreshed and migrated to other formats (PDF, PNG, HTML) from its original

[1]This dissertation follows the style of the *International Journal on Digital Libraries.*


12 versions found
3 remote & 4 cached versions
2 cached PDF versions
3 versions (2 nasa.gov & 1 mpg.de)

Google Scholar BETA
ml nelson ja kaplan
Search

Scholar Results **1** - **10** of about **9,270** for **ml nelson ja kaplan**. (**0.03** seconds)

A comparison of queueing, cluster and distributed computing systems
JA Kaplan, **ML Nelson** - View as HTML - Cited by 71 - Web Search
... Joseph A. **Kaplan** (**ja.kaplan**@larc.nasa.gov) Michael L. **Nelson** (**ml.nelson**@larc.nasa.gov) NASA Langley Research Center June, 1994 Abstract ...
NASA Technical Memorandum, 1993 - techreports.larc.nasa.gov - tworoads.net - cmpharm.ucsf.edu - phi.sinica.edu.tw - all 12 versions »

[CITATION] A Comparison of Queueing
JA Kaplan, **ML Nelson** - Cited by 6 - Web Search
Cluster and Distributed Computing Systems, NASA Langley ..., 1994

A Comparison of Queueing, Cluster and Distributed Computing Systems (1994) (Make Corrections) (8 citations)
Joseph A. Kaplan, Michael L. Nelson

View or download:
fsu.edu/pub/drago/dep/kaplan.ps.Z
kari.re.kr/NASA/larc/94...tm109025.ps.Z
nasa.gov/pub/techreport...tm109025.ps.Z
Cached: PS.gz PS PDF Image Update Help

From: fsu.edu (more)
From: kari.re.kr/ltrs/1994cit
(Enter author homepages)

CiteSeer Home/Search Context Related

(Enter summary)

Comment on this article

Abstract: Using workstations clusters for distributed computing has become popular with the proliferation of inexpensive, powerful workstations. Workstation clusters offer both a cost effective alternative to batch processing and an easy entry into parallel computing. However, a number of workstations on a network does not constitute a cluster. Cluster management software is necessary to harness the collective computing power. In this paper, we compare a variety of cluster management and queueing... (Update)

Internet Archive Wayback Machine
http://web.archive.org/web/*/http:
Google

INTERNET ARCHIVE WaybackMachine

Enter Web Address: http:// All Take Me Back Adv

Searched for http://techreports.larc.nasa.gov/ltrs/PDF/tm109025.pdf **2** Results

* denotes when site was updated.

Search Results for Jan 01, 1996 - May 03, 2005									
1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
0 pages	0 pages	1 pages	0 pages	1 pages	0 pages	0 pages	0 pages	0 pages	0 pages
		May 03, 1998 *		Sep 25, 2000 *					

Home | Help

Copyright © 2001, Internet Archive | Terms of Use | Privacy Policy

Web | Images | Video | Directory | Local | News | Produ
YAHOO! SEARCH A Comparison of Queueing, Cluster and Distributed
Shortcuts
Computing Systems -citesee

1. A comparison of queueing, cluster and distributed computing systems
A **comparison of queueing**, **cluster and distributed computing systems A comparison of queueing**, **cluster and distributed computing systems** Using workstation clusters for **distributed computing** has become popular with the proliferation of ...
ntrs.nasa.gov/archive/nasa/.../19940032425_1994032425.pdf - More from this site

2. A Comparison of Queueing, Cluster and Distributed Computing Systems, NASA TM-109025 (Revision 1), June 1994,
Joseph A. Kaplan and Micchael L. Nelson, **A Comparison of Queueing**, **Cluster and Distributed Computing Systems** , NASA TM-109025 (Revision 1), June 1994, pp. 50 ,, (95KB PS, 143KB PDF). ... **of cluster** management **and queueing systems**: **COmputing** in **DIstributed** Networked ...
techreports.larc.nasa.gov/ltrs/94/tm109025.refer.html - 3k - Cached - More from this site

3. A Comparison of Queueing, Cluster and Distributed Computing Systems (PDF)
... **A Comparison of Queueing**, **Cluster and. Distributed Computing Systems** ... a variety **of cluster** management. **and queueing systems**: **COmputing** in **DIstributed** Networked Environments (CODINE ...
www.mpip-mainz.mpg.de/theory/general/tm109025.pdf - 143k - View as html - More from this site

FIG. 1: Refreshing and migrating occurring in the living Web.

compressed PostScript. Only one of the links points to the original 1994 location. This type of *in vivo* preservation is not guaranteed by any single institution or archive; it is the result of the distributed efforts of users, web administrators and commercial services.

A major contributor to *in vivo* preservation is the Web Infrastructure (WI), the collection of commercial web search engines (e.g., Google, Yahoo, Live Search, etc.), personal web archives (e.g., Furl.net and Hanzo:Web), web archives operated by non-profit companies (e.g., the Internet Archive's Wayback Machine), and research projects (e.g., CiteSeer and NSDL). The WI supports refreshing and migrating of web resources, often as side-effects of their intended purposes. Some members of the WI are in competition with each other (such as Google, Live Search and Yahoo) to increase their holdings and will continue to improve the utility of the WI. Although members of the WI may come and go, the combined efforts of the WI can ensure the wide-spread and long-term preservation of many web resources, some of which may not have recognized value today but may in the future.

The WI can be utilized as a *lazy preservation* service for capturing entire websites; lost websites can be reconstructed from the WI on-demand by the public. Like RAID (Redundant Arrays of Inexpensive Disks) systems, where reliable storage is built on top of individually unreliable disks, the individual WI elements are also unreliable, at least with respect to any given resource. However, unlike RAIDs, the WI elements cannot be directly controlled. When a website is lost and backups are unavailable, lazy preservation may offer the only means for recovery. For example, when a website is hacked or an ISP goes out of business, an owner without any backups may reconstruct their website from the WI and make it available once again. Or when a popular website is no longer available because the owner has passed away, an interested third party can reconstruct the website and use the recovered information for their own use, or they may make the resource available once again to the community. In the face of complete loss, the WI may provide hope and relief, even if only a portion of what was lost can be found.

2 OBJECTIVE

The objective of this dissertation is to *demonstrate the feasibility of using the WI as a preservation service* and *to evaluate how effectively this previously unexplored service can be utilized for reconstructing lost websites.* To meet this objective, this dissertation focuses on answering these research questions:

- What interfaces are necessary for a member of the WI to be used in website reconstruction? (Chapter III)
- What types of resources are typically stored in the WI search engine caches, and how up-to-date are the caches? (Chapter IV)
- How successful is the WI at preserving short-lived web content? (Chapter IV)
- How much overlap is there with what is found in search engine caches and the Internet Archive? (Chapter IV)
- How does a web-repository crawler work, and how can it reconstruct a lost website from the WI? (Chapters V and VI)
- What types of websites do people lose, and how successful have they been recovering them from the WI? (Chapter VI)
- How completely can websites be reconstructed from the WI? (Chapter VII)
- What website attributes contribute to the success of website reconstruction? (Chapter VII)
- Which members of the WI are the most helpful for website reconstruction? (Chapters VI and VII)
- What methods can be used to recover the server-side components of websites from the WI? (Chapter VIII)

3 APPROACH

A number of experiments have been designed to answer the posed research questions.

Characterizing the WI: To evaluate the WI as a preservation resource, an experiment was designed to sample from members of the WI (search engine caches) and analyze the content found cached and missing from cache. An overlap analysis was performed examining sampled content with the holdings of the Internet Archive. A second experiment observed WI caching behavior by analyzing crawl logs and performing targeted searches on three search engines as they discovered and cached four decaying web collections.

Web-repository crawling: The concept of crawling the WI is introduced, and the architecture for a web-repository crawler is presented. An implementation of a web-repository crawler called Warrick is discussed, and the algorithm used by Warrick to reconstruct websites from four web repositories (Internet Archive, Google, Live Search and Yahoo) is provided.

Reconstruction from the WI: The effectiveness of reconstructing websites from the WI is evaluated using three controlled experiments on a variety of websites. The first experiment used Warrick to reconstruct 24 websites using single repositories by themselves and all four repositories together. In the second experiment, the same 24 websites were reconstructed using three different crawling policies which were evaluated for effectiveness. The final experiment reconstructed 300 randomly selected websites over a period of three months, and website characteristics were analyzed to discover which attributes influenced reconstruction success.

Reconstructing website server components: Several methods are proposed for recovering the server components of a website from the WI. The feasibility of the approaches were demonstrated by creating a digital library (DL) using Eprints software [53]; the pages of the DL were modified to store the server components using erasure codes. The DL was exposed to the WI over a period of four months and then removed from the Web to measure how long it could be reconstructed after it was lost.

4 ORGANIZATION

This dissertation is organized as follows:

Chapter 2: Preserving the Web – The Web preservation problem is introduced (digital preservation concepts, link rot on the Web and in academic literature), and various strategies for combating the problem are discussed (web archiving, backup mechanisms, search engine caches, etc.).

Chapter 3: Lazy Preservation and the Web Infrastructure – The lazy preservation paradigm is introduced along with various members of the Web Infrastructure. Limitations of lazy preservation are discussed, various web repositories (individual members of the WI) are introduced, and methods to mine content from the various web repositories are presented.

Chapter 4: Characterizing the Web Infrastructure – The Web Infrastructure is characterized by observing its behavior to discover and preserve four decaying web collections. Its holdings are measured in a second experiment.

Chapter 5: Web-Repository Crawling – The concept of a web-repository crawler is introduced and its architecture is discussed in detail. Lister queries, crawling policies and URL canonicalization issues are discussed.

Chapter 6: Warrick, a Web-Repository Crawler – Warrick, the first web-repository crawler, is introduced and its algorithm presented. A queueing system for Warrick called Brass is also discussed, and usage data from the public is provided.

Chapter 7: Evaluating Lazy Preservation - The findings of three reconstruction experiments using Warrick are presented. The first examines the aggregate performance of the WI versus using individual members, and the second experiment examines the efficiency of three crawling policies. The final experiment gauges what might be recovered from the WI if the "typical" website were lost today.

Chapter 8: Recovering a Website's Server Components – Several methods for recovering the generative components of a website are discussed. One promising method for injecting the server components into the WI using erasure codes is implemented, and experimental results are shared.

Chapter 9: Conclusions and Future Work – The findings are summarized, the contributions of this dissertation are listed and directions for future work are presented.

CHAPTER II

PRESERVING THE WEB

As Jeff Rothenberg aptly put it, "Digital information lasts forever or five years, whichever comes first" [148]. In regards to the Web, it might be said, "A web page lasts forever or five *months*, whichever comes first." Web pages that are here today may not be here tomorrow, or at least not at the same location. This phenomena is known as *link rot* or web decay [90], and although a number of solutions have been offered to combat link rot, today's Web user is still very familiar with the "404 Not Found" message.

In this chapter, the groundwork is laid for defining the scope of link rot and the various solutions (and limitations) for combatting it. Digital preservation concepts are defined, and a number of systems and mechanisms are presented which could be used to recover missing web content. Web crawling is also discussed as it pertains to web archiving and search engine caching.

1 LINK ROT

Link rot continues to be a problem for several reasons:

- The Web relies on each web resource to be accessible from a URL, yet URLs are usually tied directly to the website's domain name, directory structure and sometimes even the operating system (`http://foo.edu/~joe/` on a Linux server may become `http://foo.edu/joe/` if moved to a Windows server). Any changes to the website's infrastructure could result in link rot.
- Almost anyone with Web access can easily create a web page, yet there is no requirement that the creator or any other entity maintain access to the page.
- Although a URL may continually be accessible over time, the content it points to may change. There is no mechanism in place to alert a user what has changed about a web page or when the change occurred.
- There is usually no timestamp associated with a link or URL. Although a link to a URL may be valid when it is initially created, it may be difficult or impossible to later verify when the URL was created and therefore what content was actually being pointed to.
- Domain names expire, institutions disband, website owners pass away, and pages are abandoned out of disinterest. Content may be removed from the Web because the information is no longer relevant or correct, it is embarrassing to the owner or other third parties or it violates the law.
- Mechanisms to redirect web users to the new location of a web page are often difficult to setup or maintain and are usually not automated.

Research in the area of link rot has mainly focused on measuring the disappearance of specific resources, especially in reference to academic citations, and finding methods to combat link rot. Each of these will be discussed in turn.

1.1 Web Decay

The mortality rates for general web pages is high. Koehler [88, 89] estimates the half-life of a website and web page to be 2.9 and 1.6 years, respectively. Brewster Kahle, founder of the Internet Archive, estimates that web pages have an average life expectancy of only 100 days [174], and a 2003 study found that two percent of the Web disappeared from its current location every week [57].

Link rot has had a significantly negative impact on scholarly literature which regularly cites web resources. In one of the earliest URL persistence studies of scholarly literature [77], one-third of 47 URLs from scholarly e-journals (published from 1993 to 1995) were inaccessible in 1995. Another study [103] monitored 515 URLs that referenced scientific content or education from 2000-2001 and found 16.5% of the URLs became inaccessible or had their content changed. Other studies have shown pervasive link rot in law review articles [150], MEDLINE abstracts [175], digital library articles [111] and World Wide Web training and education articles [151]. Computer science literature is not immune: a study examining computer science articles obtained from the CiteSeer ResearchIndex database [63] found the percentage of invalid referenced URLs increased from 23% from articles written in 1999 to 53% from 1994 articles [96]. Another study of articles from the ACM and IEEE Computer Society on-line digital libraries showed that referenced URLs had a half-life of approximately four years from the publication date [156].

Even resources protected within the confines of a digital library are not always accessible. A study that tested 1000 digital objects (using URLs) from a collection of digital libraries found a 3% loss (after manual searching) in URL accessibility during 2000-2001 [125].

1.2 Preventing Link Rot

Because link rot is such a pervasive problem, a number of solutions have been offered. The HTTP protocol [58] provides built-in mechanisms for relocating pages that move by manually configuring a web server to return a 3xx status code and a pointer to the new location of the page. This functionality is frequently not available to those who do not have access to web server configuration files or who no longer own the domain name of the old URLs. It also does not help locate pages that are no longer accessible on the Web.

A proactive solution to the problem is to avoid creating URLs that will likely become inaccessible later. Berners-Lee [22] has developed popular guidelines for creating durable URLs. Most blogging software today encapsulates some of these best-practices when creating durable URLs (permalinks) to blog posts, and in fact much of the popularity and success of the Blogosphere can be attributed to the wide-spread use of permalinks [162].

A number of indirection mechanisms like Persistent URLs (PURLs) [152], handles [93], and Digital Object Identifiers (DOIs) [137] have also been developed to allow creation of permanent URLs. Unfortunately, adoption of these mechanisms has largely been limited to digital libraries and archives. This is likely due to the fact that extra software may be required on the client or server, and that most webmasters do not have the motivation or perceive the personal benefit of using such services.

Phelps and Wilensky [138] proposed the use of *robust hyperlinks* and search engines to find resources after they have gone missing. Robust hyperlinks are URLs with a lexical signature (LS) of 5 words appended onto the end that attempt to capture the uniqueness of the resource. When the resource goes missing, the LS can be submitted to search engines to find the new location of the resource. The use of lexical signatures has not been widely adopted for several reasons: the extra work required of the webmaster to generate them for every link, the problem of having to re-generate the signatures when the documents change, and the lack of sufficient motivation for the webmaster to produce the signatures in the first place. In many cases, a broken link is a problem for someone else, not the link's owner.

Another system for automatically discovering the new location of a missing URL is called Opal [75, 76]. Opal uses lexical signatures, search engines and web archives to discover similar documents at other locations and leverages the wisdom of crowds when the new location cannot be automatically located. Opal is in an early state of development and has not been publicly deployed. A less advanced service called Pagefactor [135] maintains a user-contributed database of inaccessible URLs and their new locations; a bookmarklet can be added to a browser to quickly check if a URL has been registered in the Pagefactor database.

All of these mechanisms require individuals to do some amount of work to prevent link rot. For example, a web administrator could create a redirection rule to ensure a request for `http://foo.edu/~joe/` redirects to `http://foo.edu/joe/`, but unless the administrator is properly motivated to create the rule (a directive from a boss or financial incentive), it will likely not be created. Additionally, the administrator may lack the knowledge to create redirection rules or simply be unaware of the consequences of changing a website's URLs.

2 PRESERVING THE WEB

All the mechanisms presented in Section 1.2 are an attempt to preserve mainly the current web graph. But none of the mechanisms guarantee long-term access to web content, nor do they account for changes in content over time. In the face of potential loss, archivists may take a variety of steps to ensure items deemed 'important' are not lost over time.

2.1 Digital Preservation Strategies

A number of strategies have been suggested to ensure the long-term preservation of digital documents. In practice, the following three methods have been widely deployed:

- **Refreshing** – Copying data to different media or systems [171]. This strategy preserves the bit stream composing the object and is necessary when the underlying media or system is soon to become unreadable or obsolete.

- **Migration** – Transferring data to newer system environments [171]. This may include converting the resource from one format to another, from one operating system to another or from one programming language to another so the resource remains fully accessible and functional.

- **Emulation** – Replicating the functionality of an obsolete system [149]. Emulating an older system which created or used the object ensures the object's behavior and "look-and-feel" are preserved.

All three of these methods are labor-intensive and expensive, requiring a commitment of institutional resources to successfully implement on a large scale. Furthermore, a repository must ensure the trustworthiness of its archival process [84] and maintain the provenance of its curated objects [64]. As data becomes increasingly "born digital" and individuals continue to amass communications, photos, music and video in digital form, considerable effort must also be employed individually to refresh and migrate personal data [104].

As will be discussed next, refreshing is the primary strategy employed to ensure the Web is preserved– both institutions and individuals make copies of web pages and websites in case they disappear. As the Web ages and older formats become obsolete, the need for migrating and emulating older content is slowly emerging [146, 159].

2.2 Archiving the Web

Many initiatives have been undertaken to preserve snapshots of the Web for historians, researchers and posterity. Due to its immense size, archiving the Web is very challenging from a technical standpoint. Due to the diversity of content and variety of laws protecting work published online, web archiving is also challenging from a legal standpoint. Day [48, 49] has outlined a number of challenges to archiving the Web and lists several current initiatives.

The Internet Archive (IA), founded by Brewster Kahle, is the first large-scale attempt to create an archive of the publicly accessible Web [85]. IA started archiving the Web in October 1996 [30]; they rely primarily on Alexa Internet for providing crawl data, supplemented with focused crawls from their in-house web crawler Heretrix [120]. Because Alexa is a private company focusing on popular websites, and because they do not disclose the inner workings of their crawler or crawling policies, IA's holdings are not complete, nor are they necessarily representative of the Web at large [166]. The IA is mirrored at the Bibliotheca Alexandrina [25] which aids in protecting their large archive from loss.

National libraries, national archives and a variety of institutions have also recognized the value of preserving portions of the Web and have launched a number of programs to archive culturally important websites (e.g., Sweden's Kulturarw3 [10], France's BnF [1], the UK's UKWAC Archive [14] and Australia's PANDORA Archive [32]). Like the Internet Archive, most of these rely on web crawling to fill their archives. Some may also require their citizens to deposit their works in their archives (although enforcing such laws is problematic) [1, 72].

Social bookmarking and archiving services like Furl [61], Spurl.net [157] and Hanzo:Web [74] have emerged that allow users to archive selected web resources which can be accessed from any location at a later time. Services like WebCite [54] and StayBoyStay [158] allow users to archive web pages so they can be cited without danger of being lost. And archive-on-demand subscription services like Archive-It [7] have made it increasingly easy for individuals and institutions to archive web material.

Some systems have been built to preserve a much smaller yet significant portion of the Web. Having recognized that digital content from publishers' websites periodically disappears from the Web, LOCKSS [144] was created to archive selected publishers' websites for library use. LOCKSS is a peer-to-peer (P2P) system which performs focused crawls of a publisher's website and archives the web content in a closed environment. When a user tries to access content that is no longer available on the publisher's website, LOCKSS will return the archived version instead. LOCKSS is a dark archive– its holdings are not publicly accessible due to constraints that publishers place on their content.

CiteSeer also preserves academic output; it is a project developed at the NEC Research Institute to discover and index scientific and academic papers on the Web [63]. It migrates discovered documents to several file formats including PDF, PostScript and PNG and makes these files accessible from their website. If the original document disappears from the Web, it will remain accessible from CiteSeer.

Other systems have been developed that reside on a web server to aid in archiving web resources. TTApache [52] is a transaction-time web server [51], implemented as an Apache module, which performs automatic archiving of web pages on a per-request basis. Web pages must be requested (by a user, robot or other mechanism) in order to be archived. TTApache maintains a document version history for all requested documents and provides a URL query interface to view the archived versions. iPROXY [142] is similar to TTApache in that it serves as an archiving service for web pages, but it is not limited to a specific web server. iPROXY is a proxy server that archives web pages as they are visited by a user, and it can also archive selected websites through crawling. A URL query interface is used to access archived web pages. A similar approach using a proxy server with a content management system for storing and accessing Web resources was proposed by [55], but they did not provide many details of their prototype. The Apache module mod_oai can produce archival-ready resources for a website, but it relies on web archives to actually store the resources [126].

2.3 Web Server Archiving

While web archives and web server add-ons focus on archiving the client-side representation of web pages, none of them are capable of archiving the server components that generate dynamically-produced websites. In the advent of a website's loss, the generative functionality would be completely lost unless a backup of the components were available. This is especially problematic for sites whose pages cannot be crawled and are therefore unlikely to be archived by any of the systems mentioned in Section 2.2.

To aid in the recovery of a website in the advent of a disk failure, the InfoMonitor archives the files and server-side components of a website [42, 43]. It runs on a separate machine that periodically scans web server's filesystem and propagates any detected changes (like modified or deleted files) to the archive.

Many backup systems provide similar protection for a web server although access to older versions of files may not be present in all systems. Most operating systems come with backup software that will automate backups to local disk drives [26], and there are numerous off-site backup solutions available for a subscription fee. For example, Backup.com offers an automated backup solution of 250 MB for $5 a month [11], and Amazon S3 offers $0.15 per GB a month with a $0.20 per GB transfer rate [5]. Researchers have also proposed a number of methods (usually involving P2P systems) to make backup simple and affordable for the masses (e.g., [44, 121]), but such systems are not yet in widespread use.

2.4 Search Engine Caching

Google was the first search engine to allow users access to the pages they indexed [133]. They made these *cached* pages available for all HTML resources that they crawled. As other document formats have become available on the Web (e.g., PDFs, Word documents, PostScript, etc.), Google has converted these to HTML in order to index them. They have also made these HTML versions available through a "View as HTML" link next to each resource. Many commercial search engines now permit access to their cached resources.

Search engine caches are not intended necessarily to be used for preservation purposes; most search engines will purge a resource from their cache as soon as they discover it is no longer available on the Web [119]. Nevertheless, some individuals have used Google's cache to find web pages that have gone missing from the Web [24, 105, 114, 161].

Besides the work presented in this dissertation, little research has been performed examining the caching behavior of commercial search engines. A notable exception is an experiment by Lewandowski et al. [98] which measured the index freshness of Google, Yahoo and MSN by examining 38 cached web pages from German websites daily for six weeks. They found Google maintained the freshest index with cached pages averaging 3.1 days in age. MSN was second with 3.5 days, and Yahoo was last with 9.8 days. MSN was the most consistent at providing pages that were no older than 20 days, but Google and Yahoo both had pages cached that were almost 2 months old.

3 WEB CRAWLING

3.1 Background

Web crawling is important to website preservation because web archives and search engines (the thrust of this dissertation) primarily rely on web crawling to discover new resources and refresh their holdings. Crawler limitations are thus directly related to the holdings of web archives and search engines.

A web crawler is a tool which automates the collecting of websites [136]. A crawler is initially given a seed URL. It makes an HTTP request for the resource, downloads it and examines it for links to other resources it is interested in crawling. This process generally continues until there are no more URLs left to discover.

In order to avoid overloading a web server, a crawler typically delays some amount of time between requests and avoids crawling URLs that are marked off-limits using the robots exclusion protocol [91]. These factors limit the ability to precisely capture a website at a specific moment in time since some pages may be changing immediately before or after being crawled, and the crawler is prohibited from crawling some pages. Additionally, crawlers may want to limit the depth to which they crawl a website to avoid crawler traps, a set of URLs that cause a crawler to crawl indefinitely [78].

Using a web crawler and a large number of seeds, a large portion of the surface web can be discovered, but those websites not connected to larger sections of the Web will be missed [28].

Keeping a web repository fresh is another challenge of web crawling. Because the Web is too large to be crawled, a number of methods have been developed to crawl only those resources that are popular (e.g., [38, 124]) or to schedule re-crawls based on resource change rates (e.g., [35, 128]).

3.2 Crawling the Deep Web

The deep web [21] or hidden web [60, 95] are those web-accessible resources that cannot be found by crawling the surface web. It has been estimated that the deep web is several magnitudes larger than the surface web [21]. The deep web is composed of several parts:

- **Dynamic content** – Pages that are returned in response to a query (e.g., in response to a form submission) or in response to user interaction (e.g., AJAX-style web applications).
- **Unlinked content** – Pages that are not linked to by other pages or linked only through JavaScript, Flash or form interfaces.
- **Limited-access content** – Pages that are protected by passwords or CAPTCHAs [4] or accessible to only certain user agents, pages marked "off-limits" by the robots exclusion protocol [91] or special crawler-protection tags like `noindex`, `nofollow` and `noarchive`.
- **Contextual web** – Pages whose content varies depending on the user agent, IP address range, cookie contents or previous navigational sequences.
- **Non-textual content** – Resources which may not be crawled (especially by search engines) because they are not of a format the repository wants to crawl.

Efforts to crawl the deep web have focused on performing queries on Web search interfaces using human-assisted techniques [99, 100, 141] and automated methods [130]. Other methods like the Sitemap Protocol and mod_oai (discussed in the next section) can also be used to discover deep web resources on a website. OAI-PMH can used to locate deep web resources held in OAI-compliant repositories. In June 2005, McCown et al. [113] found 3.3M unique web resources that were discovered using OAI-PMH and found that Yahoo had indexed 65% of the resources. Google had indexed 44% and MSN only 7%, but almost a quarter of the resources (many deep web) were not indexed by any of the three search engines. Yahoo's success was primarily attributed to a deal they had made in March 2004 [169] to obtain content harvested by OAIster [71], a service that uses OAI-PMH to harvest academically oriented digital resources from various OAI repositories.

Other systems like DP9 and Errols have been developed to allow search engines to discover OAI repository holdings through traditional web crawling. The DP9 gateway service harvests records from OAI repositories in batches and converts them into web pages that search engines can then crawl and index [101]. The Extensible Repository Resource Locators (Errols) for OAI Identifiers project allows the creation of URLs that dynamically perform OAI-PMH queries against registered OAI repositories and generates HTML pages suitable for Web crawling [180].

3.3 Crawling Alternatives

Recognizing the inefficiencies of conventional web crawling and the difficulty of finding deep web resources, researchers have examined various alternative methods that would allow crawlers to more efficiently discover a web server's resources. Brandman et al. [27] proposed that web servers export metadata about their holdings, indicating which resources are available and when they have last changed. Having access to such metadata allows crawlers to only access those resources that they are interested in and those resources which have changed since the last visit. Nelson et al. [126, 127] extended this idea by building an Apache module (mod_oai) which exports web server metadata using OAI-PMH.

The Sitemap Protocol [155], based on the ideas from [27], allows a website to provide a text file listing the set of URLs available on the site, their change rates and their relative importance. The file can be generated by hand or by automated methods. The Sitemap Protocol was initially developed by Google in June 2005, but it has now been adopted by most commercial search engines as a standard. Some search engines like Google and Yahoo have also added support for reading Real Simple Syndication (RSS) and Atom feeds [179] and harvesting data from OAI-PMH requests [172].

Other researchers have proposed a push model [70, 163]. In this scenario, a website would notify an interested third party when its holdings were updated. This model has never been adopted by any commercial search engine.

4 PRESERVING WEBSITES

As noted in previous sections, there are a number of strategies and systems that have been built to preserve web resources and complete websites. Web archivists have been focused on preserving websites that they deem important, and search engines provide short-term preservation of web resources in their caches. Services like Furl and Hanzo:Web allow users to save important web resources, and various systems like TTApache, iPROXY and InfoMonitor provide versioning of website URLs and web server files.

Each of these preservation systems can be divided into two categories based on the level of preservation they provide:

1. *client view* - This is the view the client is presented after making an HTTP request for a web resource.

2. *server view* - These are the generative mechanisms or server components (scripts, files, databases, etc.) that are responsible for generating the client view of the website.

Preserving the client view is generally sufficient to those who are interested in what a particular website said or looked like at the time of capture. Capturing the client view is straightforward for a third party, but the server view is generally only accessible to the website's administrator. In the event of losing a website, recovering the client view is generally sufficient for websites composed of static resources since the the client view is equivalent to the files that reside on the web server. But recovering the server view is often more important when the functionality of a dynamic website needs to be restored.

Table 1 summarizes the systems or mechanisms that are used or could be used for preserving web resources along with their limitations if using the mechanism to restore a lost website. Some systems are designed specifically for recovering lost websites (e.g., filesystem backup and InfoMonitor) but are useless to third parties wanting to recover the website. Some of the systems generally serve other

TABLE 1: Systems for preserving and recovering web resources.

System	Summary	Limitations
1. Filesystem backup	Long-term storage of website on permanent storage media.	Backup may be lost or performed incorrectly, each website must be configured for backup, backups are inaccessible to third parties.
2. InfoMonitor	Archival of website web server.	Backup of archived content may be required, backups are inaccessible to third parties.
3. Web archives	Long-term archival of “important” websites.	Website may not be considered “worthy” of inclusion, slow updating.
4. Search engine caches	Most up-to-date resource accessed from last crawl.	Not all resources may be cached, non-HTML resources migrated to lossy formats.
5. LOCKSS	Long-term storage of websites in P2P system from selected publisher websites.	Only publisher websites are preserved, only libraries have access to LOCKSS.
6. CiteSeer	Automated indexing and migration of academic articles.	Only preserves academic articles.
7. TTApache	Apache module offering long-term archival of resources that are requested through a browser from a specific website.	Resources that have not been accessed (browsed) will not be archived, backup to archive may be lost or damaged.
8. iPROXY	Proxy server offering long-term archiving of requested resources from a variety of websites.	Backup of archive may be necessary, browsing through a proxy server.
9. Furl/Spurl	Archival of web resources by manual selection.	Complete coverage of any website is unlikely.
10. Hanzo:Web/ Archive-It	Archival of websites by manual selection.	Websites must be selected in advance.
11. Browser cache	Storage of requested resources.	Individual browser caches are not publicly accessible.

functions (e.g., CiteSeer and a browser’s cache), but they could also be used for recovering at least some portions of a website. Systems 1-2 preserve the server view of a website, and 3-11 preserve only the client view. If a complete website is lost, each of these systems could be used to reconstruct the website with variable degrees of success.

5 CONCLUSIONS

The ephemeral nature of the Web creates a number of challenges for preserving it. Link rot is a pervasive problem, and a number of approaches have been developed to combat it that range from archiving individual web servers and important web pages to caching large portions of the surface web to preserving large snapshots of the Web. The next chapter discusses how some of these approaches can be utilized together to provide a layer of preservation with wide-coverage of the Web and minimal work on behalf of web content creators.

CHAPTER III

LAZY PRESERVATION AND THE WEB INFRASTRUCTURE

Conventional web preservation projects and techniques require a significant investment of time, money and effort and are thus applicable only to collections of known value. The limited scope of such projects may leave a number of potentially important web collections unprotected. For these unprotected collections, *lazy preservation* can provide a broad preservation service with no cost to individual content producers. Lazy preservation make use of the Web Infrastructure (WI), the distributed efforts of multiple organizations and companies that store web content, often as by-products of their user services. This chapter defines lazy preservation and investigates how stored content can be mined from the WI members (or web repositories).

1 LAZY PRESERVATION

It is difficult, if not impossible, to know in advance when digital data is soon to be lost. Catastrophic events like hard drive crashes, viruses, fires and even death are unpredictable. And it is also not always clear what is important and should be saved. The importance of online content will usually vary depending on the intended audience and subject matter. Some material may be deemed important on a national level, like websites documenting the 9-11 terrorist attacks. Other material may have significant importance to a smaller group of individuals, like the blog of a deceased family member or a website specializing in a particular genre of music. A considerable amount of institutional effort may be applied to preserving 9-11 content, but content that is important to a much smaller audience may not be afforded such protections.

An individual or organization may take steps to preserve their own content by utilizing sound backup procedures. Thus the publisher incurs a cost in time and money to ensure their content remains accessible. Studies have found that most individuals do not make regular backups of their personal data [31], and organizations that do make backups are not immune to backup failures or fires [59]. Even the most secure backup procedures will not ensure that a website remains available once the publisher loses interest in the website, goes bankrupt or dies.

Because individuals and organizations are sometimes unable (or unwilling) to incur personal cost to ensure long-term access to their websites, there is a need for a preservation service which can provide wide coverage of the Web with minimal expense to the publisher. Revisiting the systems described in the previous chapter, most cost the producer time or money to protect their websites from loss.

Figure 2 maps the systems of the previous chapter according to the publisher's relative cost (in time, effort or equipment) to have their website preserved and the coverage that the systems provide to the entire Web. Backup systems for the server view require setup and configuration, paying subscription fees or hardware costs. TTApache and iPROXY require installation and extra measures to ensure a complete website is preserved. Personal archiving services like Furl, Hanzo:Web and Archive-It can be used to preserve web content *a priori*, but they are not helpful if action was

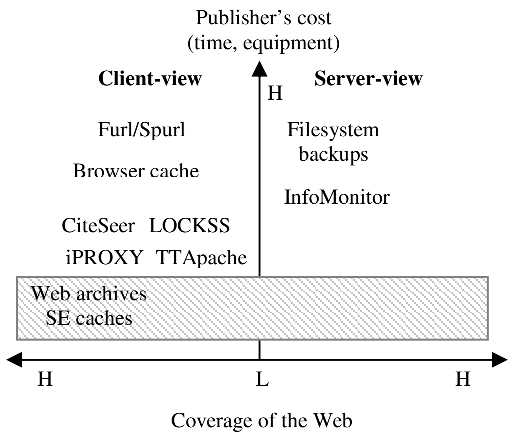


FIG. 2: Publisher's cost and Web coverage of preservation systems/mechanisms.

not taken prior to the loss. Browser caches may have stored a large range of requested resources, but access to browser caches are generally not made public. Other systems like LOCKSS are dedicated to preserving content from a small group of sites and do not provide large coverage of the Web.

However, search engines and web archives preserve a wide range of web resources for the client view. There is no cost to the publisher except to ensure that their content is crawlable. Other systems that are publicly accessible like Furl, Hanzo:Web and CiteSeer may not have as large a coverage, but they could still be used in addition to search engine caches and web archives to find some web resources that may have been missed.

These repositories of web resources (or **web repositories**) form the backbone of the **Web Infrastructure (WI)**. The WI refreshes and migrates web content, often as side effects of their user services. **Lazy preservation** utilizes the WI to provide a passive but broad preservation service for the Web. Rather than relying on an institutional commitment to preserve small collections of known worth, lazy preservation makes use of the distributed, uncoordinated nature of the WI to provide large-scale preservation of web resources of unknown importance.

The shaded portion of Figure 2 shows there is no equivalent preservation service for preserving the server view of a website. However, the WI could be used to provide such a service if the server components could be discreetly injected into resources already ingested by the WI. Chapter VIII will explore how this can be done in more detail.

There are several scenarios where lazy preservation would be the only preservation mechanism in place to recover a lost website:

- When the website owner has not backed-up their website.
- When the backup of a lost website is not accessible, was not performed properly or is incomplete.
- When a third party with no access to a backup wishes to recover a lost website.

TABLE 2: Sample of reconstructed websites.

Date	Website description	How it was lost
Oct. 2005	WWW 2006 conference website	Fire destroyed the building housing the web server
Jan. 2006	Kickball organization	Web server's hard drive crashed
Mar. 2006	Christian academic article archive	ISP hosting the site for free discontinued their service
Apr. 2006	Educational site about Roman history	Website owner died, and website eventually ceased operating
Apr. 2006	Fan site of pop singer Shiri Maimon	Website was hacked, and owner did not have backups
Aug. 2006	Personal website	ISP accidentally deleted the site and did not have a backup
Oct. 2006	Limo company	All ISP's sites were pulled by police because ISP was hosting illegal content
Oct. 2006	US Congressman Mark Foley's websites*	Sites were pulled when Foley resigned over inappropriate conduct
Oct. 2006	Supports sexual assault victims of Darfur*	Unknown
Apr. 2007	Academic law organization in India	Owner accidentally deleted the site
Apr. 2007	Professional technical organization	Web server crashed and backups only partially worked

*Reconstructed by request of the Library of Congress.

Some examples of lazy preservation being used to recover lost websites are listed in Table 2 [110]. The examples serve to illustrate the wide variety of reasons why websites are lost: hard drive crashes, hacking, accidents, death, etc.

2 LIMITATIONS

The WI is limited by a number of factors. Web repositories like search engine caches and web archives primarily use web crawling to find content and update their holdings. Web crawling is limited to the surface web, the portion of the Web that is connected with hyperlinks. As mentioned in the previous chapter, there is a large part of the web, the deep (or invisible) web, which is often inaccessible to crawlers. Therefore pages that are returned in response to a query, hidden behind JavaScript, Flash and CAPTCHAs, and pages that are not connected to others are usually inaccessible to the WI. Additionally, search engines and web archives respect the robots exclusion protocol which makes certain URLs off-limits to the crawlers, and they will not store web pages that use `noarchive` meta tags [79, 81, 122, 145]. They also limit the depth they crawl websites to avoid crawler traps.

Fortunately, search engines have become so ubiquitous today that websites are designed to be "crawler friendly" [68, 173]. In fact, an entire industry (search engine optimization or SEO) has emerged that tunes websites for optimal crawling and positioning in search engine results. Many websites have also started to use special tools like the Sitemap Protocol [155] to allow crawlers to locate deep web content on their sites.

The search engine's goal of providing information to users may sometimes conflict with preservation goals. Search engines often avoid crawling duplicate content [154] or content determined to be spam [56], and they are not interested in indexing or caching some resource types as will be discussed later in this chapter. They may not crawl deeply into a website since such pages may not be useful to search engine users [12]. And they are not likely to keep pages cached long after they detect the pages are no longer accessible on the Web as will be explored in Chapter IV. The caching mechanism, in fact, is the most limiting factor for using search engine caches for preservation: missing content is quickly purged from cache, and older versions of a resource cannot be located in the cache.

Web archives face many of the same technical challenges as search engines when crawling the Web, but they are not as likely to avoid duplicate content, spam or any particular resource format. Because the archive's goal is to preserve the Web just as it was found, web archives are very useful web repositories when reconstructing lost websites. Unfortunately, web archives may not have the same amount of resources (human and technical) available as commercial search engines, and their coverage of the Web may be somewhat limited.

3 WEB REPOSITORIES

Web repositories are members of the WI that crawl and store a sizeable portion of the Web and provide URI granularity to access their stored resources. Web repositories are generally accessible to automated queries through the HTTP protocol, using direct GET queries or an API, web service, or any other method that can be automated.

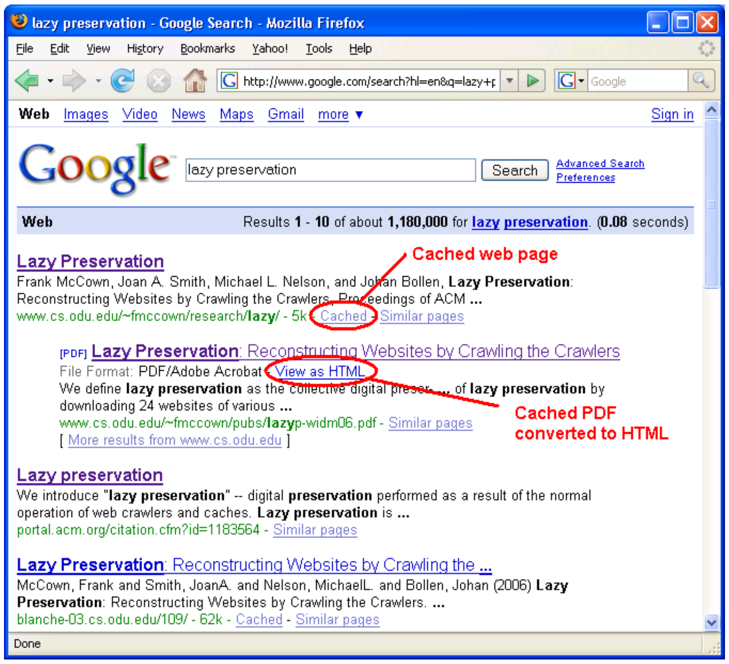
3.1 Web Repository Types

Web repositories may be characterized by the depth of their holdings. Search engine caches like Google's are repositories that store only the latest resource crawled; when a resource is re-crawled, the new resource replaces the older version in the repository. The depth d of such repositories is one since only one copy of a resource is maintained. Search engine caches are thus examples of **flat repositories** ($d = 1$). A **deep repository** ($d > 1$) maintains older versions of resources; it stores a datestamp along with each version of the resource with the same URI. Web archives like the Internet Archive (with $d = \infty$) are examples of deep repositories.

Web repositories may also be categorized by the access granted to their holdings [167]. Dark repositories do not provide public access to their holdings. Some repositories are made dark due to legal restrictions (e.g., national web archives) or because they are merely serving as a fail-safe in case the original resource is no longer accessible. Light repositories, however, place minimal access controls on their holdings. Search engine caches and IA are examples of light repositories. In some cases though, IA holdings may be considered dark since IA will not allow public access to an archived resource when the website from which the resource was obtained contains a robots.txt entry blocking access to the item [80]. Some repositories may be called "grey" because they are limited to a small number of individuals, like Cornell's Yesternet [8, 9] which is limited to researchers.

3.2 Search Engine Caches and Web Archives

Google, Live Search (previously MSN Search) and Yahoo are three of the largest and most popular search engines, garnering over 86% of all web searches in May 2007 [19]. Recent overlap studies have


lazy preservation - Google Search - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://www.google.com/search?hl=en&q=lazy+p
Web Images Video News Maps Gmail more ▼ Sign in
Google
lazy preservation
Search
Advanced Search
Preferences
Cached web page
Cached PDF converted to HTML
Done

Web Results **1** - **10** of about **1,180,000** for **lazy preservation**. (**0.08** seconds)

Lazy Preservation
Frank McCown, Joan A. Smith, Michael L. Nelson, and Johan Bollen, **Lazy Preservation**: Reconstructing Websites by Crawling the Crawlers, Proceedings of ACM ...
www.cs.odu.edu/~fmccown/research/**lazy**/ - 5k - Cached - Similar pages

[PDF] **Lazy Preservation**: Reconstructing Websites by Crawling the Crawlers
File Format: PDF/Adobe Acrobat - View as HTML
We define **lazy preservation** as the collective digital preser... of **lazy preservation** by downloading 24 websites of various ...
www.cs.odu.edu/~fmccown/pubs/**lazy**p-widm06.pdf - Similar pages
[More results from www.cs.odu.edu]

Lazy preservation
We introduce "**lazy preservation**" -- digital **preservation** performed as a result of the normal operation of web crawlers and caches. **Lazy preservation** is ...
portal.acm.org/citation.cfm?id=1183564 - Similar pages

Lazy Preservation: Reconstructing Websites by Crawling the ...
McCown, Frank and Smith, JoanA. and Nelson, MichaelL. and Bollen, Johan (2006) **Lazy Preservation**: Reconstructing Websites by Crawling the Crawlers. ...
blanche-03.cs.odu.edu/109/ - 62k - Cached - Similar pages

FIG. 3: Google provides "Cache" and "View as HTML" links to cached resources.

shown that all three search engines have indexed significantly different portions of the Web [16, 69]. All three search engines make their indexed pages available from the "Cached" link (or "View as HTML" for other document types) provided next to each search result (Figure 3). These cached resources can be used for recovering lost pages if caught in time. Ask, a notable commercial search engine, also makes some resources available from their cache, but as will be seen in the next chapter, they do not make enough of their cached resources available to be a very useful web repository.

The Internet Archive is currently the world's largest publicly accessible web archive. Although there are several large national web archives in existence [49], none are focused on saving the entire Web. An overlap analysis of the Internet Archive's holdings (more in Chapter IV) indicates there are numerous resources found in search engines caches that are not found in the Internet Archive.

Although there are a number of web repositories that could be used for lazy preservation, these four repositories are the largest and are therefore the primary focus in this dissertation:

1. **Google -** Google is the most widely used search engine [19] and has likely indexed more of the Web than any other search engine [16, 69]. In 1997, it was the first search engine to make "cached" pages available from their search results, and they have continued to do so despite the copyright concerns their innovation raised [133]. Google was also the first search engine to make available a free SOAP-based API for accessing their index in an automated fashion [131]. Unfortunately, they were also the first search engine to deprecate their API despite a large install base [41]. And as mentioned before, Google pioneered the use of the Sitemap Protocol [155], a technique which allows web servers to advertise the URLs they would like search engines to index; the protocol

has since been adopted by Live Search and Yahoo.

2. **Live Search** - Microsoft's search engine, previously known as MSN Search until 2006, has struggled behind Google and Yahoo in terms of index size and usage. Nevertheless, they have continued to improve their search engine over the last several years, adding their own internal image search functionality in 2006 [153]. The Live Search web search API is the least restrictive of all the search engine APIs in terms of query limits, and their API has been shown to be the most synchronized with what regular users see [116, 118].

3. **Yahoo** - Yahoo is the oldest of the three web search companies, and they currently have the second largest index. Their REST-based API has flexible query limits and can be used to acquire both textual and image resources from their cache. Yahoo was one of the first search engines to pioneer the use of OAI-PMH to gather deep web resources [169], and it was estimated in 2005 that they had indexed more of the OAI-PMH corpus than the other two search engines [113].

4. **Internet Archive** - The only deep repository on the list, IA is also the only non-commercial repository of the four. IA has traditionally relied on crawl data from Alexa Internet, a commercial company primarily focusing on web traffic monitoring, but recently they have augmented their archive with crawls from their own web crawler Heritrix [82]. IA's holdings have previously been 6-12 months out-of-date, but recent improvements have narrowed the gap to three months. IA does not have an API.

3.3 Storage Formats

The Internet Archive strives to maintain an accurate snapshot of the Web as it existed when crawled. Therefore they archive each resource in the same format in which it was crawled. Search engines have traditionally been HTML-centric, but as the amount of non-HTML resources has grown, so has their ability to index and cache these types of resources.

When adding PDF, PostScript and Microsoft Office (Word, Excel, PowerPoint) resources to their cache, the search engines create HTML versions of the resources which are stripped of all images. In most cases it is not possible to recreate the canonical version of the document from the HTML version. Figure 4 shows a PDF as it was cached by MSN (Live Search), Yahoo and Google. Although "IMPROVING" looks like an image in two of the caches, it is text displayed in HTML using a style sheet.

The search engines have separate search interfaces for their images, and they keep only a thumbnail version of the images they cache due to copyright law [132] (Figure 5). As shown in Table 3, most resources are not stored in their canonical format in the search engine caches. In some cases like Flash, text from the resource may be indexed, but the binary resource is not accessible from the repository. Only HTML appears to be stored in its canonical format across all four repositories.

There are many other resource types not mentioned in Table 3 that may be found on the Web: binary programs, video, audio and archive files. While the IA attempts to preserve these resource types, search engines are generally not interested in them since there is little or no textual content that can be indexed.

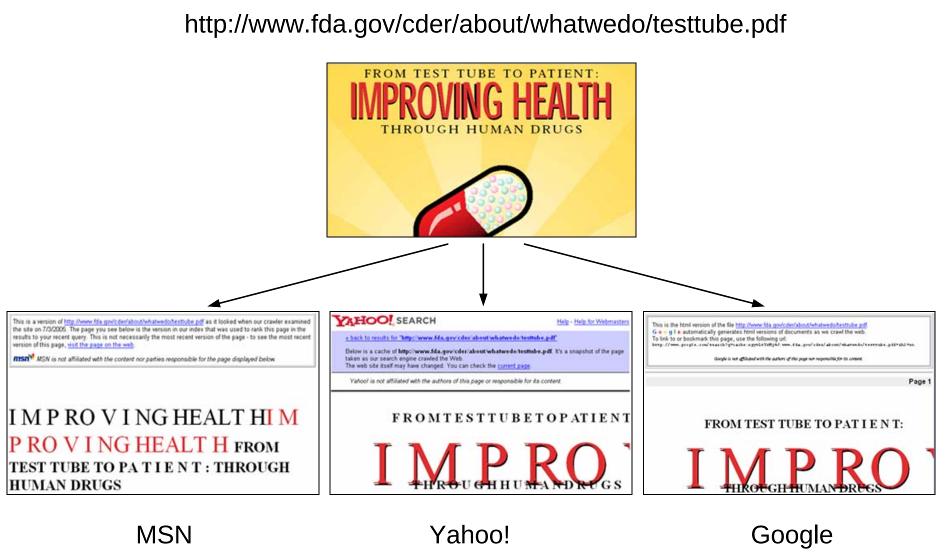
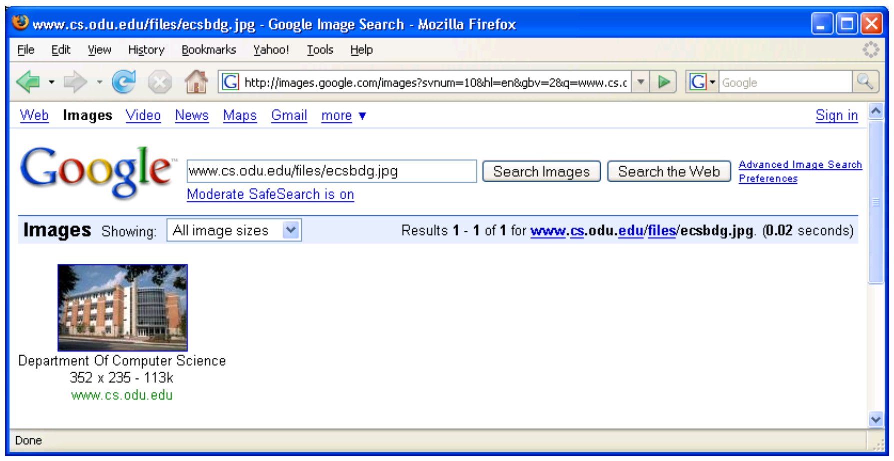

http://www.fda.gov/cder/about/whatwedo/testtube.pdf
FROM TEST TUBE TO PATIENT:
IMPROVING HEALTH
THROUGH HUMAN DRUGS
MSN
Yahoo!
Google

FIG. 4: Original PDF and the HTML cached versions.


File Edit View History Bookmarks Yahoo! Tools Help
Web Images Video News Maps Gmail more ▾
Sign in
Search Images
Search the Web
Advanced Image Search
Preferences
Moderate SafeSearch is on
Images Showing: All image sizes
Done

www.cs.odu.edu/files/ecsbdg.jpg - Google Image Search - Mozilla Firefox

http://images.google.com/images?svnum=10&hl=en&gbv=2&q=www.cs.c

www.cs.odu.edu/files/ecsbdg.jpg

Results **1** - **1** of **1** for **www.cs.odu.edu/files/ecsbdg.jpg**. (**0.02** seconds)

Department Of Computer Science
352 x 235 - 113k
www.cs.odu.edu

FIG. 5: Google's cached thumbnail of http://www.cs.odu.edu/files/ecsbdg.jpg.

TABLE 3: Web repository-supported data types as of July 10, 2007.

Type	Google	Yahoo	Live	IA
HTML	C	C	C	C
Plain text	M	M	M	C
GIF, PNG, JPG	M	M	M	C
JavaScript	M		M	C
MS Excel	M	∼S	M	C
MS PowerPoint	M	M	M	C
MS Word	M	M	M	C
PDF	M	M	M	C
PostScript	M	∼S		C
Flash	∼S			C
XML	C	∼S		C

C = Canonical version is stored
M = Modified version is stored (image thumbnails or HTML conversions)
∼S = Indexed but stored version is not accessible

3.4 Accessing

Web repositories may be accessed using HTTP requests through a web browser or automated program. Some search engines like Google and Live prohibit automated queries against their web user interface (WUI) in their Terms of Service [66, 123]. Google and Yahoo have also been known to quit responding to IP addresses that they suspected were issuing automated queries [106, 108]. Figure 6 shows a web page returned by Google when it detected a query was automated.

Because querying search engine WUIs by automated methods is problematic, use of their APIs is often preferable. Unfortunately, the APIs are limited to a fixed number of queries per day, and recent research has shown that Google's and Yahoo's APIs may be offering results from smaller indexes; in many cases, the APIs do not produce the same results as the WUIs [116, 118]. Google's API does not provide access to their images, so querying their WUI is necessary. Google also appears to be phasing-out their SOAP-based API since they are no longer issuing access keys [41].

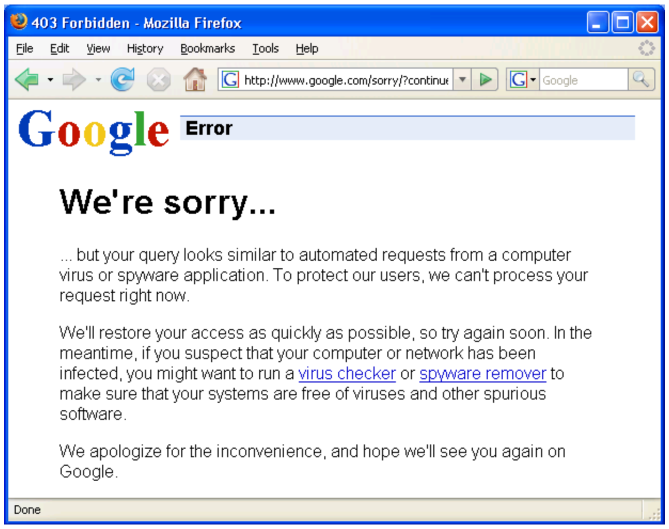
3.5 Crawl Interface

A web repository can be used for lazy preservation when it provides an interface which allows it to be crawled just like a website. In order to be crawled, a web repository must support, at a minimum, the ability to handle the query, "What resource r do you have stored for the URI u?" where u is the resource's URL when it was obtained by the repository on the Web:

$$r \leftarrow \texttt{getResource}(u) \tag{1}$$

The repository will respond to this query with the resource in the same format (canonical format) in which it was crawled from the Web or in some altered format, such as a thumbnail version of an image. If the resource has not been stored, the repository will respond with some negative response.

The format of the resource may be supplied as metadata with the resource, or it may be inferred


403 Forbidden - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.google.com/sorry/?continue
Google
Done

Google Error

We're sorry...

... but your query looks similar to automated requests from a computer virus or spyware application. To protect our users, we can't process your request right now.

We'll restore your access as quickly as possible, so try again soon. In the meantime, if you suspect that your computer or network has been infected, you might want to run a virus checker or spyware remover to make sure that your systems are free of viruses and other spurious software.

We apologize for the inconvenience, and hope we'll see you again on Google.

FIG. 6: Google error screen returned when automated queries are detected.

from the URI, its usage in the source HTML page or the repository from which the resource is obtained. For example, if Google were queried for the resource at `http://foo.org/hello.gif`, it could be inferred that the resources is an image because of the URI's .gif ending or because it was referenced in an <img> tag. Since Google Images is known only to store thumbnail images, it can additionally be inferred that the resource is not in its canonical format. Had the resource been obtained from the Internet Archive, it could be assumed the image was in its canonical format.

Deep repositories should allow resources to be obtained using a URI u and the datestamp ds, the day on which the resource was crawled, to distinguish among multiple versions of the same resource:

$$r \leftarrow \texttt{getResource}(u, ds) \tag{2}$$

The repository will only respond with the resource at URI u that was crawled on date ds. To obtain a list of available datestamps that are acceptable, the repository should ideally support a query which returns the stored resources for the given URI u:

$$D \leftarrow \texttt{getResourceList}(u) \tag{3}$$

where D is the set of all datestamps stored in the repository for the resource.

Flat repositories should ideally provide the date the returned resource was crawled, perhaps as metadata in r from getResource, or by supporting such a query:

$$d \leftarrow \texttt{getCrawlDate}(u) \tag{4}$$

Having a datestamp for each resource allows a web-repository crawler to chose between multiple resources from multiple repositories that have been crawled at different times, like, for example,

TABLE 4: Implementation summary of web-repository interfaces.

Queries	IA	Google	Live	Yahoo
getResource	✓	✓	✓	✓
getResourceList	✓	N/A	N/A	N/A
getCrawlDate	✓	Limited to HTML	Images not supported	Last modification date, images not supported
getAllUris	✓	First 1000 results only	First 1000 results only	First 1000 results only

when wanting to select the most up-to-date resource. It also allows the crawler to reject resources that are not from a particular time frame.

An additional query type which allows for more efficient crawling is, "What resources R do you have stored from the site s?":

$$R \leftarrow \texttt{getAllUris}(s) \tag{5}$$

The returned value R is a set of all URIs and datestamps stored in the repository $\{(u_1, d_1)$, (u_2, d_2), ..., $(u_n, d_n)\}$. This type of query, called a **lister query**, can greatly decrease the number of queries that a repository must handle since it can be asked for only those resources it is known to contain (the effects of this speed-up are discussed in Chapter VII). Additionally, deep repositories may provide an enhanced query to limit the returned resources to a particular date range dr:

$$U \leftarrow \texttt{getAllUris}(s, dr) \tag{6}$$

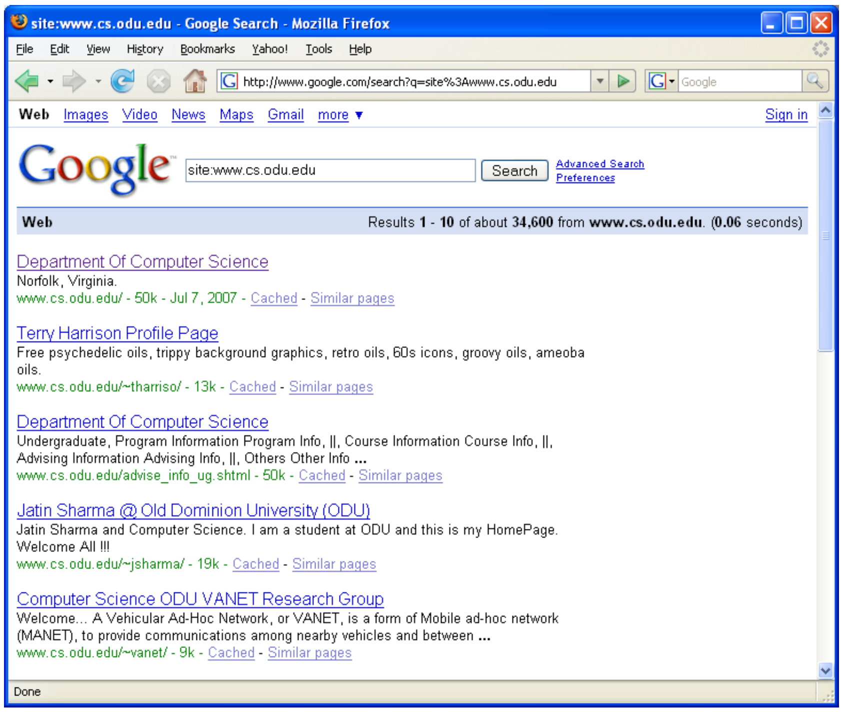
3.6 Implementing the Interface

Each of the repositories implement the interface queries of the previous section in different ways. Some only partially support the interface. A summary of the web-repository interfaces supported by the four web repositories is given in Table 4.

Flat Repository

The three search engines implement the repository interfaces in a similar manner with a few significant differences. All three search engines support `getAllUris`, `getCrawlDate` and `getResource` queries. To perform `getAllUris` (lister queries), the query parameter "site:" is used. For example, to retrieve all URLs for the website `www.cs.odu.edu` from Google, the query `site:www.cs.odu.edu` is used as illustrated in Figure 7. All three search engines will return only the first 1000 results, so lister queries are of limited use for large websites.

The `getCrawlDate` query is not supported directly by Google and Live, but it is indirectly supported by examining the metadata returned from the `getResource` query. Figure 8 shows how the cached page from `http://www.cs.odu.edu/` was found in Google by using the "cache:" query parameter (Live's cached page heading is similar). Notice how the crawl date is located in the Google header of the page. Unfortunately, Google does not make the cached date available for non-HTML resources, so `getCrawlDate` is only partially supported by Google.


site:www.cs.odu.edu - Google Search - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://www.google.com/search?q=site%3Awww.cs.odu.edu
Web Images Video News Maps Gmail more ▾
Sign in
Google
site:www.cs.odu.edu
Search
Advanced Search
Preferences
Done

Web Results **1** - **10** of about **34,600** from **www.cs.odu.edu**. (**0.06** seconds)

Department Of Computer Science
Norfolk, Virginia.
www.cs.odu.edu/ - 50k - Jul 7, 2007 - Cached - Similar pages

Terry Harrison Profile Page
Free psychedelic oils, trippy background graphics, retro oils, 60s icons, groovy oils, ameoba oils.
www.cs.odu.edu/~tharriso/ - 13k - Cached - Similar pages

Department Of Computer Science
Undergraduate, Program Information Program Info, ||, Course Information Course Info, ||, Advising Information Advising Info, ||, Others Other Info **...**
www.cs.odu.edu/advise_info_ug.shtml - 50k - Cached - Similar pages

Jatin Sharma @ Old Dominion University (ODU)
Jatin Sharma and Computer Science. I am a student at ODU and this is my HomePage. Welcome All !!!
www.cs.odu.edu/~jsharma/ - 19k - Cached - Similar pages

Computer Science ODU VANET Research Group
Welcome... A Vehicular Ad-Hoc Network, or VANET, is a form of Mobile ad-hoc network (MANET), to provide communications among nearby vehicles and between **...**
www.cs.odu.edu/~vanet/ - 9k - Cached - Similar pages

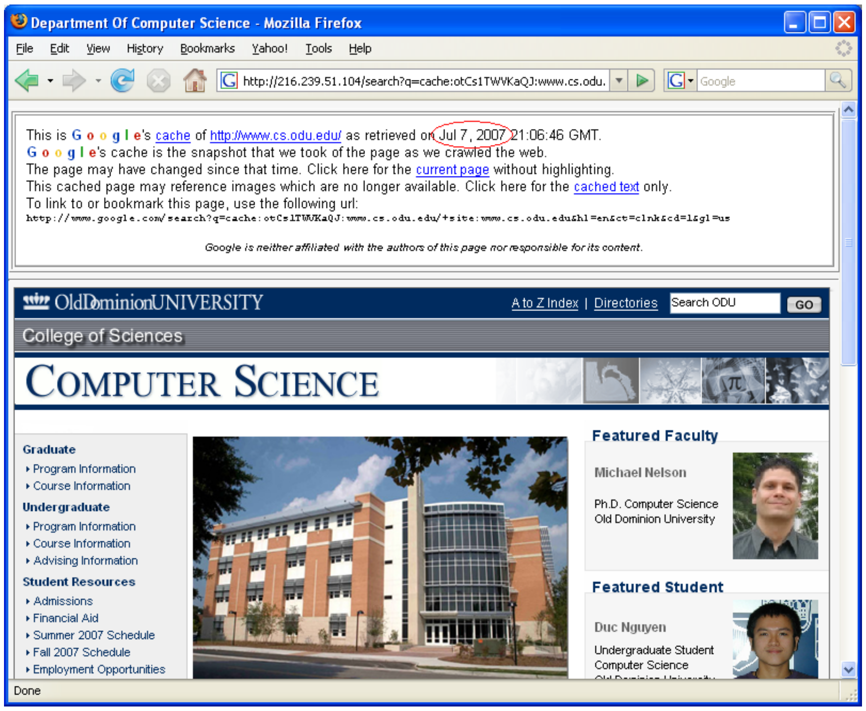
FIG. 7: Google query for `site:www.cs.odu.edu` returning "about 34,600" results.

Yahoo does not place the crawled date in their cached page's heading; it must be retrieved using their API. The date returned by Yahoo's API is not the date the resource was crawled but instead the date they last noticed it had changed (the ModificationDate). Yahoo does not specify how they measure change, but it is likely based on the resource's Last Modified HTTP header.

In order to access images from the search engines, a different but similar procedure is involved. The `getAllUris` query is invoked against Google Images (or images API for Live and Yahoo). To illustrate, Figure 9 shows Google Images being queried for all images on `www.cs.odu.edu`. Again, only the first 1000 results can be obtained. The `getResource` query is implemented by finding the URL of the thumbnail image and accessing it directly. Note that no datestamp is available for images from any of the search engines.

Deep Repository

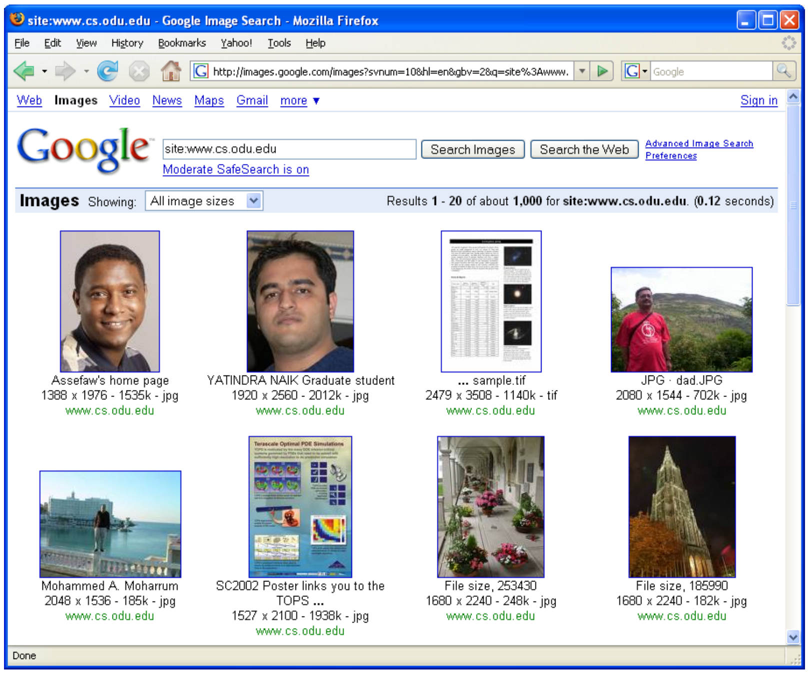
The interface queries `getAllUris`, `getResourceList` and `getResource` are all supported by IA. To perform `getAllUris`, an HTTP request is made in the form: `http://web.archive.org/web/*sr_0nr_20/http://www.cs.odu.edu/*`. IA will respond to this query with a listing of all URIs it has stored that match the given URI prefix as shown in Figure 10. IA does not limit the number of results returned, so paging through all resources stored is possible. The `getAllUris` query with date range is not completely supported by IA, but a date range of one year (e.g., limiting to 2006 is `http://web.archive.org/web/2006*/http://www.cs.odu.edu/`) or one month (e.g., limiting to



This is Google's cache of http://www.cs.odu.edu/ as retrieved on Jul 7, 2007 21:06:46 GMT.
Google's cache is the snapshot that we took of the page as we crawled the web.
The page may have changed since that time. Click here for the current page without highlighting.
This cached page may reference images which are no longer available. Click here for the cached text only.
To link to or bookmark this page, use the following url:
`http://www.google.com/search?q=cache:otCs1TWVKaQJ:www.cs.odu.edu/+site:www.cs.odu.edu&hl=en&ct=clnk&cd=1&gl=us`

Google is neither affiliated with the authors of this page nor responsible for its content.

FIG. 8: Google query for `cache:http://www.cs.odu.edu/` returning the cached page as it was crawled on July 7, 2007.



site:www.cs.odu.edu - Google Image Search - Mozilla Firefox
http://images.google.com/images?svnum=10&hl=en&gbv=2&q=site%3Awww.
Web Images Video News Maps Gmail more
Sign in
site:www.cs.odu.edu
Search Images
Search the Web
Advanced Image Search
Preferences
Moderate SafeSearch is on
Images Showing: All image sizes
Done

Results 1 - 20 of about 1,000 for site:www.cs.odu.edu. (0.12 seconds)

Assefaw's home page
1388 x 1976 - 1535k - jpg
www.cs.odu.edu

YATINDRA NAIK Graduate student
1920 x 2560 - 2012k - jpg
www.cs.odu.edu

... sample.tif
2479 x 3508 - 1140k - tif
www.cs.odu.edu

JPG · dad.JPG
2080 x 1544 - 702k - jpg
www.cs.odu.edu

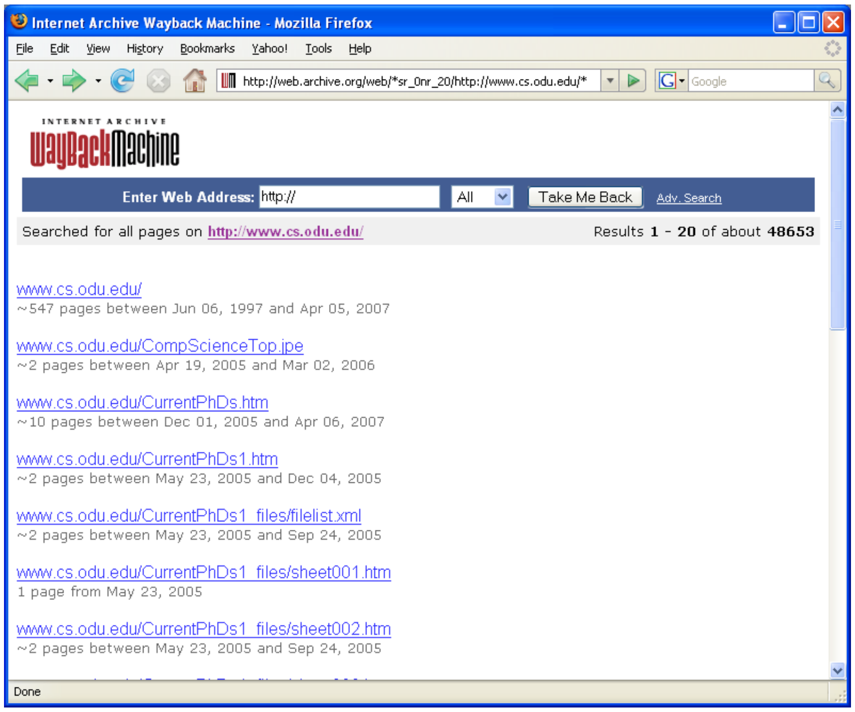
Mohammed A. Moharrum
2048 x 1536 - 185k - jpg
www.cs.odu.edu

SC2002 Poster links you to the TOPS ...
1527 x 2100 - 1938k - jpg
www.cs.odu.edu

File size, 253430
1680 x 2240 - 248k - jpg
www.cs.odu.edu

File size, 185990
1680 x 2240 - 182k - jpg
www.cs.odu.edu

FIG. 9: Google Image query for `site:www.cs.odu.edu` returning “about 1,000” results.


Internet Archive Wayback Machine - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://web.archive.org/web/*sr_0nr_20/http://www.cs.odu.edu/*
Google
INTERNET ARCHIVE
WayBackMachine
Enter Web Address: http://
All
Take Me Back
Adv. Search
Done

Searched for all pages on http://www.cs.odu.edu/ Results 1 - 20 of about 48653

www.cs.odu.edu/
~547 pages between Jun 06, 1997 and Apr 05, 2007

www.cs.odu.edu/CompScienceTop.jpe
~2 pages between Apr 19, 2005 and Mar 02, 2006

www.cs.odu.edu/CurrentPhDs.htm
~10 pages between Dec 01, 2005 and Apr 06, 2007

www.cs.odu.edu/CurrentPhDs1.htm
~2 pages between May 23, 2005 and Dec 04, 2005

www.cs.odu.edu/CurrentPhDs1_files/filelist.xml
~2 pages between May 23, 2005 and Sep 24, 2005

www.cs.odu.edu/CurrentPhDs1_files/sheet001.htm
1 page from May 23, 2005

www.cs.odu.edu/CurrentPhDs1_files/sheet002.htm
~2 pages between May 23, 2005 and Sep 24, 2005

FIG. 10: IA response to http://web.archive.org/web/*sr_0nr_20/http://www.cs.odu.edu/*.

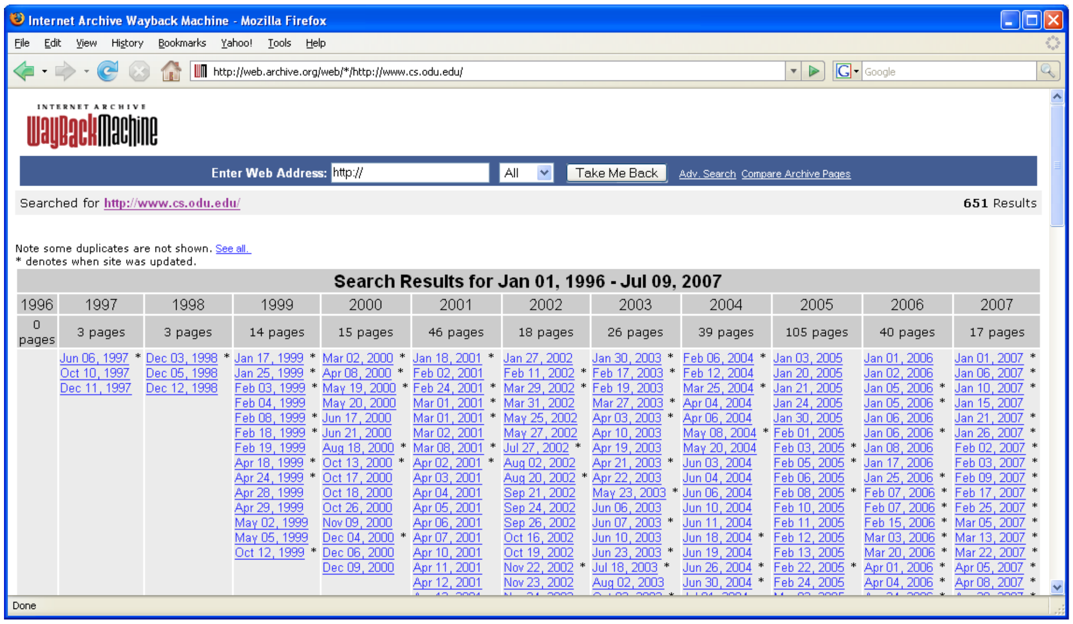
January 2006 is `http://web.archive.org/web/200601*/http://www.cs.odu.edu/`) is supported.

IA also supports the `getResourceList` query by returning a page listing all the stored versions of a page. Figure 11 shows an example of a `getResourceList` query for the root page of the www.cs.odu.edu website.

The `getResource` query is implemented by directly accessing any of the stored resources. For example, the page stored on January 24, 2005, can be accessed at `http://web.archive.org/web/20050124084644/http://www.cs.odu.edu/`. Note the date (ds) in YYYYMMDD format (20050124) embedded in the URL.

4 CONCLUSIONS

The Web Infrastructure is a combination of many distributed players who migrate and refresh web content, often as a side effects of their primary mission. Lazy preservation can utilize the WI as a passive preservation service for recovering a large portion of the Web that may be left unprotected. In the advent of complete loss, lazy preservation may be the only means by which a missing website can be recovered. Lazy preservation makes use of web repositories to recover missing content, and the interfaces to four web repositories (Google, Live Search, Yahoo and the Internet Archive) were explored this chapter. The next chapter will investigate the holdings and behavior of these four repositories to better understand how they can be utilized for lazy preservation. Chapter V shows how these repositories can be crawled by a web-repository crawler.



Internet Archive Wayback Machine - Mozilla Firefox

http://web.archive.org/web/*/http://www.cs.odu.edu/

Searched for http://www.cs.odu.edu/ **651** Results

Note some duplicates are not shown. See all.
* denotes when site was updated.

Search Results for Jan 01, 1996 - Jul 09, 2007

1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007
0 pages	3 pages	3 pages	14 pages	15 pages	46 pages	18 pages	26 pages	39 pages	105 pages	40 pages	17 pages
	Jun 06, 1997 *	Dec 03, 1998 *	Jan 17, 1999 *	Mar 02, 2000 *	Jan 18, 2001 *	Jan 27, 2002	Jan 30, 2003 *	Feb 06, 2004 *	Jan 03, 2005	Jan 01, 2006	Jan 01, 2007 *
	Oct 10, 1997	Dec 05, 1998	Jan 25, 1999 *	Apr 08, 2000 *	Feb 02, 2001	Feb 11, 2002 *	Feb 17, 2003 *	Feb 12, 2004	Jan 20, 2005	Jan 02, 2006	Jan 06, 2007 *
	Dec 11, 1997	Dec 12, 1998	Feb 03, 1999 *	May 19, 2000 *	Feb 24, 2001 *	Mar 29, 2002 *	Feb 19, 2003	Mar 25, 2004 *	Jan 21, 2005	Jan 05, 2006 *	Jan 10, 2007 *
			Feb 04, 1999	May 20, 2000	Mar 01, 2001 *	Mar 31, 2002	Mar 27, 2003 *	Apr 04, 2004	Jan 24, 2005	Jan 05, 2006 *	Jan 15, 2007
			Feb 08, 1999 *	Jun 17, 2000	Mar 01, 2001 *	May 25, 2002	Apr 03, 2003 *	Apr 06, 2004	Jan 30, 2005	Jan 06, 2006	Jan 21, 2007 *
			Feb 18, 1999 *	Jun 21, 2000	Mar 02, 2001	May 27, 2002	Apr 10, 2003	May 08, 2004 *	Feb 01, 2005	Jan 06, 2006 *	Jan 26, 2007 *
			Feb 19, 1999	Aug 18, 2000 *	Mar 08, 2001 *	Jul 27, 2002 *	Apr 19, 2003	May 20, 2004	Feb 03, 2005 *	Jan 08, 2006	Feb 02, 2007 *
			Apr 18, 1999 *	Oct 13, 2000 *	Apr 02, 2001 *	Aug 02, 2002	Apr 21, 2003 *	Jun 03, 2004	Feb 05, 2005 *	Jan 17, 2006	Feb 03, 2007 *
			Apr 24, 1999 *	Oct 17, 2000	Apr 03, 2001	Aug 20, 2002 *	Apr 22, 2003	Jun 04, 2004	Feb 06, 2005	Jan 25, 2006 *	Feb 09, 2007 *
			Apr 28, 1999	Oct 18, 2000	Apr 04, 2001	Sep 21, 2002	May 23, 2003 *	Jun 06, 2004	Feb 08, 2005 *	Feb 07, 2006 *	Feb 17, 2007 *
			Apr 29, 1999	Oct 26, 2000	Apr 05, 2001	Sep 24, 2002	Jun 06, 2003	Jun 10, 2004	Feb 10, 2005	Feb 07, 2006 *	Feb 25, 2007 *
			May 02, 1999	Nov 09, 2000	Apr 06, 2001	Sep 26, 2002	Jun 07, 2003 *	Jun 11, 2004	Feb 11, 2005	Feb 15, 2006 *	Mar 05, 2007 *
			May 05, 1999	Dec 04, 2000 *	Apr 07, 2001	Oct 16, 2002	Jun 10, 2003	Jun 18, 2004 *	Feb 12, 2005	Mar 03, 2006 *	Mar 13, 2007 *
			Oct 12, 1999 *	Dec 06, 2000	Apr 10, 2001	Oct 19, 2002	Jun 23, 2003 *	Jun 19, 2004	Feb 13, 2005	Mar 20, 2006 *	Mar 22, 2007 *
				Dec 09, 2000	Apr 11, 2001	Nov 22, 2002 *	Jul 18, 2003 *	Jun 26, 2004 *	Feb 22, 2005 *	Apr 01, 2006 *	Apr 05, 2007 *
					Apr 12, 2001	Nov 23, 2002	Aug 02, 2003	Jun 30, 2004 *	Feb 24, 2005	Apr 04, 2006 *	Apr 08, 2007 *

Done

FIG. 11: IA response to http://web.archive.org/web/*/http://www.cs.odu.edu/.

CHAPTER IV

CHARACTERIZING THE WEB INFRASTRUCTURE

As mentioned in previous chapters, the WI may refresh and migrate web content at will. But how quickly and thoroughly will the WI consume new web content, and how long will it retain content that is removed from the Web? How much content stored in the caches of search engines is also accessible from the IA? This chapter investigates these questions using two separate experiments; one experiment was designed to characterize the WI's ability to discover and maintain new and decaying web resources [119], and the other experiment was designed to examine more deeply the contents held in search engine caches and to measure overlap with IA's holdings [117]. From these experiments, a more holistic view of the web repositories comprising the WI emerges.

1 A MODEL FOR RESOURCE AVAILABILITY

A resource that is made available on the Web must be discovered and consumed by the WI for it to be lazily preserved. If a resource is discovered and archived by a web archive (deep repository), it will likely remain accessible from the WI even when the resource disappears from the Web. However, if a resource is discovered and cached by a search engine (flat repository) and then disappears from the Web, it will only have a limited time of protection before it is purged from the search engine's cache.

Figure 12 illustrates the life span of a web resource from when it is first made available on a web server to when it leaves a search engine cache. A web resource's time-to-live on the web server (TTL_{ws}) is defined as the number of days from when the resource is first made accessible on the server (t_0) to when it is removed (t_r). The period beginning when the resource is accessible from a search engine's cache (t_a) to when it is finally purged (t_p) defines a resource's time-to-live in the search engine cache (TTL_c). The following classifications are used to describe the resource's availability:

- **Vulnerable** - A new resource which has not yet been discovered by a search engine (t_d) and made available in the search engine cache (t_0 to t_a).

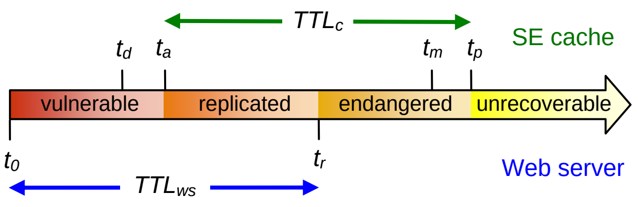


FIG. 12: Timeline of search engine resource acquisition and release.

TABLE 5: Resource availability states.

Web \ Cache	Accessible	Not Accessible
Accessible	Replicated	Vulnerable
Not Accessible	Endangered	Unrecoverable

- **Replicated** - A resource which remains accessible on the web server and has also been cached by a search engine (t_a to t_r).
- **Endangered** - A resource which is no longer accessible on the web server but still remains cached (t_r to t_p).
- **Unrecoverable** - A resource which has been discovered to be no longer accessible on the web server (t_m) and has been evicted from cache (t_p).

A resource is *recoverable* if it is currently cached (i.e., is replicated or endangered). Ideally, a cached resource will always be accessible from the cache at any given time, but as will be seen in Section 2.3, this is not always so. Therefore, a recoverable resource can only be recovered during the TTL_c period with a probability of P_r, the observed number of days that a resource is retrievable from the cache divided by TTL_c.

It should be noted that the TTL_{ws} and TTL_c values of a resource may not overlap. A search engine that is slow in updating its cache, perhaps because it obtains crawling data from a third party, may experience late caching where $t_r < t_a$. This is particularly true for a web archive like IA that makes resources available from their archive months after they have been crawled.

For a website to be lazily preserved with minimum vulnerability, its resources need to be cached soon after they appear on a website. Search engines may also share this goal if they want to index newly discovered content as quickly as possible. For lazy preservation to be maximally effective, resources should remain cached long after they have been deleted from the web server (remain endangered) so they can be recovered for many days after their disappearance. Search engines, on the other hand, may want to minimize the endangered period in order to purge missing content from their index. Resources which are "popular" by search engine standards and change frequently may be re-crawled more often than unpopular, static content, and therefore may be purged quickly once they go missing. Unfortunately, inducing a search engine to crawl a website at a specific time is not currently possible, and there is no way to externally exert control over cache eviction policies.

2 WEB INFRASTRUCTURE PRESERVATION CAPABILITY

To characterize the WI's ability to discover and retain web content, an experiment was designed to measure how long it took for three search engines (Google, MSN and Yahoo) to discover new web content and keep the content cached after it had disappeared. Four web collections were created by Joan Smith, a Ph.D. student at ODU, to control the creation and decay rate of the content. After deploying the collections, the three search engines were queried daily to determine how much of the content had been discovered and remained cached, and the crawl logs were analyzed to determine how long it took for crawled content to become accessible from the search engines' caches. The experiment provided measurements for the resource availability model developed in the previous section and revealed very different crawling and caching policies for the three search engines.

2.1 Web Collection Design

Four synthetic web collections were created with the same number of HTML, PDF and image resources. Although the content of the HTML and PDF resources varied, the link structure was identical for all four collections. The web collections were deployed in June 2005 at four different websites:

1. `http://www.cs.odu.edu/~fmccown/lazy/`
2. `http://www.cs.odu.edu/~jsmit/lazy/`
3. `http://www.cs.odu.edu/~mln/lazy/`
4. `http://www.owenbrau.com/`

The .com website was new and had never been crawled before, but the three .edu websites had existed for over one year and had been previously crawled by multiple search engines. In order for the web collections to be found by the search engines, links were placed to the root of each web collection from the .edu websites, and owenbrau's base URL was submitted to Google, MSN and Yahoo one month prior to the experiment. For 90 days resources were systematically removed from each collection.

The web collections were organized into a series of *update bins* or directories which contained a number of HTML pages referencing the same three inline images (GIF, JPG and PNG) and a number of PDF files. An index.html file (with a single inline image) in the root of the web collection pointed to each of the bins. An index.html file in each bin pointed to the HTML pages and PDF files so a web crawler could easily find all the resources. All these files were static and did not change throughout the 90 day period except the index.html files in each bin which were modified when links to deleted web pages were removed. At no time did the websites point to any missing resources.

The number of resources in the web collections were determined by the number of update bins B, the last day that resources were deleted from the collection T (the *terminal day*), and the bin I which contained three images per HTML page. Update bins were numbered from 1 to B, and resources within each bin b were numbered from 1 to $\lfloor T/b \rfloor$. Resources were deleted from the web server according to their bin number. Every n days one HTML page (and associated images) and one PDF file from bin n were deleted. For example, resources in bin 1 were deleted daily, resources in bin 2 were deleted every other day, etc. Links were also removed to the deleted HTML and PDF files from bin n's index.html file.

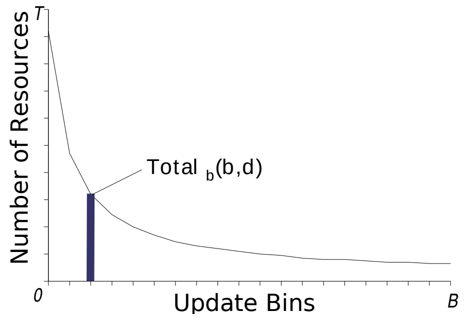


FIG. 13: Number of resources in web collection.

On any given day d during the experiment (where $d = 0$ is the starting day and $d \leq T$), the total number of resources in the web collection was defined as:

$$Total(d) = 2 + \sum_{i=1}^{B} Total_b(i, d) \tag{7}$$

The total number of HTML, PDF and image files in bin b on any day d was defined as:

$$Total_b(b, d) = HTML(b, d) + PDF(b, d) + IMG(b, d) \tag{8}$$

The total number of resources in each update bin deceased with the bin's periodicity as show in Figure 13. The number of HTML, PDF and image files in each bin b on any day d was defined as:

$$HTML(b, d) = \lfloor T/b \rfloor - \lfloor d/b \rfloor + 1 \tag{9}$$

$$PDF(b, d) = \lfloor T/b \rfloor - \lfloor d/b \rfloor \tag{10}$$

$$IMG(b, d) = \begin{cases} 3(HTML(b, d) - 1) & \text{if } b = I \\ 0 & \text{if } HTML(b, d) = 1 \\ 3 & \text{otherwise} \end{cases} \tag{11}$$

Each web collection had 30 update bins ($B = 30$) that completely decayed by day 90 ($T = 90$), and bin 2 ($I = 2$) contained supplemental images. So the total number of files in each collection on day 0 was $Total(0) = 954$. The web collections were limited to less than 1000 resources in order to limit the number of daily queries to the search engines (the following section discusses these limitations). There were fewer images than HTML and PDF pages because it was hypothesized that images were not cached as frequently as other resources, and the cost of querying for images (number of queries issued per resource) was higher than for HTML and PDF resources.

The TTL_{ws} for each resource in the web collection was determined by its bin number b, page number p, and the web collection terminal day T:

$$TTL_{ws} = b(\lfloor T/b \rfloor - p + 1) \tag{12}$$

An example PDF page from one of the web collections is shown in Figure 14. HTML pages looked very similar. Each HTML and PDF page contained a unique identifier (UID) at the top of each page that included four identifiers: the web collection (e.g., 'mlnODULPT2' means the 'mln' collection), bin number (e.g., 'dgrp18' means bin 18), page number and resource type (e.g., 'pg18-2-pdf' means page number 2 from bin 18 and PDF resource). The UID contained spaces to allow for more efficient querying of the search engines.

To generate the content of the synthetic collection, it was necessary to produce pages that appeared to be legitimate so that search engines would not find them suspicious or classify them as spam [129]. Designing such a collection is challenging because the search engines' algorithms for detecting spam are proprietary and constantly changing. Although the pages could have been generated from other indexed, pre-existing web pages, the amount of duplicate content could have led a search engine to reject indexing the pages during the de-duping process [154]. Since search engines regularly index word lists which contain unique sets of words [34], it was decided that pages would be generated from a standard English dictionary and would not contain popular search terms (e.g., "Pamela Anderson") that might result in the pages being flagged as spam. Using random words also guaranteed that every page would appear unique.

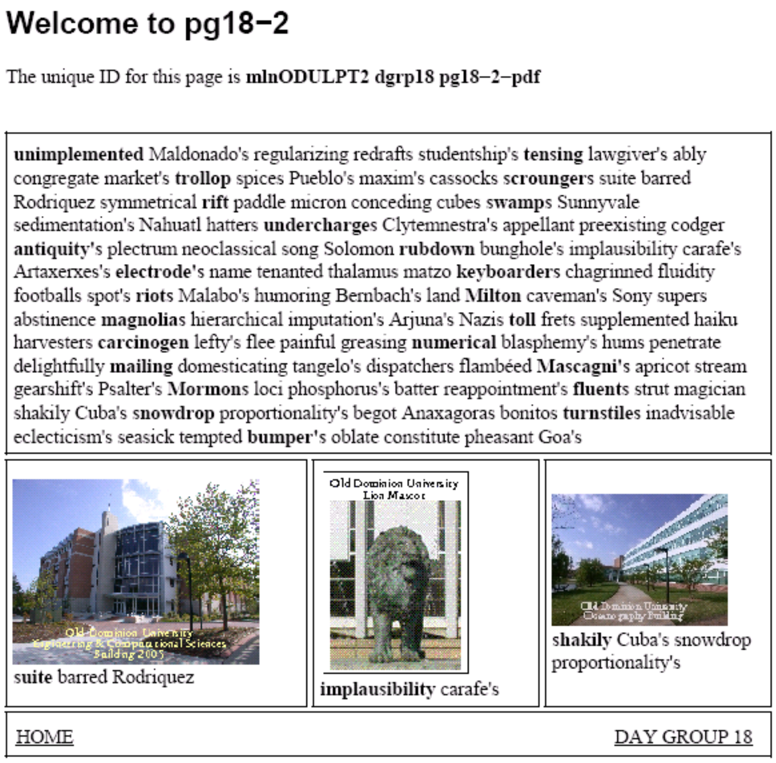
The experiment was not designed to measure the effect of top level domain, page depth, or other URL attributes on the crawling and caching behavior of the search engines; the collections were designed to only simulate "typical" web collections that may be encountered on any website. Although a number of factors might influence search engine behavior, this type of experiment is limited in scope because of the amount of queries that can be reasonably made to search engines on a daily basis. A variety of factors are examined in more depth in an experiment in Chapter VII where 300 websites are reconstructed from the WI over a period of three months.

2.2 Daily Search Engine Queries

Queries were issued to the three search engines every day of the experiment to determine if the web collections were accessible from the caches. Screen-scraping (WUI queries) were used since MSN and Yahoo had not released APIs before the experiment was initiated, and the Google API only produced 10 results per query. Therefore care was taken to minimize the number of daily queries so as not to overburden the search engines (a more complete discussion of search engine query limits is discussed in Sections 3.3 of Chapter V).

Each HTML and PDF resource's UID was used to querying each search engine. Although the complete UID could have been queried for each resource, this would have placed a large load on the search engines. Therefore queries were used which asked for the top 100 results for those items matching the resource type ('PDF' or 'HTML') using the first two parts of the UID (e.g., 'mlnODULPT2 dgrp18'). This uniquely identified the web collection and bin number[1]. Each result

[1]MSN at the time limited the results page to 50 results, so large bins sometimes required more than one query.



Welcome to pg18−2

The unique ID for this page is **mlnODULPT2 dgrp18 pg18−2−pdf**

unimplemented Maldonado's regularizing redrafts studentship's **tensing** lawgiver's ably congregate market's **trollop** spices Pueblo's maxim's cassocks **scroungers** suite barred Rodriquez symmetrical **rift** paddle micron conceding cubes **swamps** Sunnyvale sedimentation's Nahuatl hatters **undercharges** Clytemnestra's appellant preexisting codger **antiquity's** plectrum neoclassical song Solomon **rubdown** bunghole's implausibility carafe's Artaxerxes's **electrode's** name tenanted thalamus matzo **keyboarders** chagrinned fluidity footballs spot's **riots** Malabo's humoring Bernbach's land **Milton** caveman's Sony supers abstinence **magnolias** hierarchical imputation's Arjuna's Nazis **toll** frets supplemented haiku harvesters **carcinogen** lefty's flee painful greasing **numerical** blasphemy's hums penetrate delightfully **mailing** domesticating tangelo's dispatchers flambéed **Mascagni's** apricot stream gearshift's Psalter's **Mormons** loci phosphorus's batter reappointment's **fluents** strut magician shakily Cuba's **snowdrop** proportionality's begot Anaxagoras bonitos **turnstiles** inadvisable eclecticism's seasick tempted **bumper's** oblate constitute pheasant Goa's

suite barred Rodriquez

implausibility carafe's

shakily Cuba's snowdrop proportionality's

HOME

DAY GROUP 18

FIG. 14: Example PDF resource from synthetic collection.

page was parsed for links to cached versions of the resources. More formally, the total number of daily queries Q_b was:

$$Q_b = w \cdot b \cdot t \cdot s \tag{13}$$

where w is the number of web collections with b bins and t types of resources in each bin, and s is the total number of search engines queried daily.

Querying the search engine caches for images required a different strategy than the one used for querying for text-based resources. Each image was given a globally unique filename so that a query for the image's filename would result in a found or not found result. For w web collections with i images and s search engines to be queried daily, the total number of daily queries Q_i was:

$$Q_i = w \cdot i \cdot s \tag{14}$$

2.3 Crawling and Caching Observations

The web server logs were analyzed to capture the crawling behavior of Alexa Internet (who provides crawls to IA), Google, Inktomi (who provided crawls to Yahoo at the time of this experiment) and MSN on the synthetic web collections. The logs only showed a single access from Alexa which was attributed to the author's use of the Alexa toolbar. A separate IA robot accessed less than 1% of the collections which was traced back to several queries the author made with the Wayback Machine's advanced search interface early in the experiment. Therefore the only crawls observed from IA-related crawlers were artificially induced and of very limited scope.

Both HTML and PDF resources were crawled and cached in great number, but images were crawled and cached far less frequently; Google and Picsearch (the MSN Images provider at the time) were the only ones to crawl a significant number of images. The three .edu collections had 29% their images crawled, and owenbrau had 14% of its images crawled. Only four unique images appeared in Google Images, all from the mln collection. Google likely used an image duplication detection algorithm to prevent duplicate images from different URLs from being cached. Only one image (from fmccown) appeared in MSN Images. None of the cached images fell out of cache during the experiment.

Table 6 summarizes the performance of each search engine to crawl and cache 350 HTML resources from the four web collections (PDF results were similar)[2]. The table does not include index.html resources which had an infinite TTL_{ws}. There was an error in the MSN query script which caused fewer resources to be found in the MSN cache, but the percentage of crawled URLs provides an upper bound on the number of cached resources; this has little to no effect on the other measurements reported.

The three search engines showed equal desire to crawl HTML and PDF resources. Inktomi (Yahoo) crawled two times as many resources as MSN, and Google crawled almost three times as many resources than MSN. Google was the only search engine to crawl and cache any resources from the new owenbrau website.

[2]Due to a technical mishap by departmental system administrators, crawling data was unavailable for days 41-55 for owebrau and parts of days 66-75 and 97 for the .edu web collections. Cache queries could not be made on days 53, 54, 86 and 87.

TABLE 6: Caching of HTML resources from four web collections.

		fmccown	jsmit	mln	owenbrau	Ave
% URLs crawled	Google	91	92	94	18	74
	MSN	41	31	33	0	26
	Yahoo	56	92	84	0	58
% URLs cached	Google	91	92	94	20	74
	MSN	16	14	14	0	11
	Yahoo	36	65	49	0	37
t_a	Google	13	12	10	103	35
	MSN	65	65	65	N/A	66
	Yahoo	47	47	54	N/A	50
TTL_c / P_r	Google	90 / 0.78	86 / 0.82	87 / 0.83	40 / 0.98	76 / 0.86
	MSN	20 / 0.87	20 / 0.91	21 / 0.90	N/A	20 / 0.89
	Yahoo	35 / 0.57	36 / 0.55	24 / 0.46	N/A	32 / 0.53
Endangered period	Google	51	47	47	61	51
	MSN	9	7	8	N/A	8
	Yahoo	24	25	19	N/A	23

From a preservation perspective, Google out-performed MSN and Yahoo in nearly every category. Google cached the highest percentage of HTML resources (76%) and took only 12 days on average to cache new resources from the edu web collections. On average, Google cached HTML resources for the longest period of time (76 days), consistently provided access to the cached resources (86%), and were the slowest to remove cached resources that were deleted from the web server (the endangered period averaged 51 days). Although Yahoo cached more HTML resources and kept the resources cached for a longer period than MSN, the probability of accessing a resource on any given day was only 53% compared to 89% for MSN.

Figure 15 provides an interesting look at the crawling and caching behavior of Google, Yahoo and MSN. These graphs illustrate the crawling and caching of HTML resources from the mln collection; the other two edu collections exhibited similar behavior. The resources are sorted by TTL_{ws} with the longest-living resources appearing on the bottom. The index.html files which were never removed from the web collection have an infinite TTL ('inf'). The red diagonal line indicates the decay of the web collection; on any particular day, only resources below the red line were accessible from the web server. For the three graphs on the top row of Figure 15, blue dots indicate resources that were crawled on a particular day. When resources were requested that had been deleted, the web server responded with a 404 (not found) code represented by green dots above the red line. The bottom row of graphs in Figure 15 shows the cached HTML resources (blue) resulting from the crawls. Some pages in Yahoo were indexed but not accessible from cache (green).

As Figure 15 illustrates, both Google and MSN were quick to make resources available in their cache soon after they were crawled, and they were quick to purge resources from their cache when a crawl revealed the resources were no longer available on the web server. A surprising finding is that many of the HTML resources that were previously purged from Google's cache reappeared on day 102 and remained cached for the remainder of the experiment. The other two edu collections

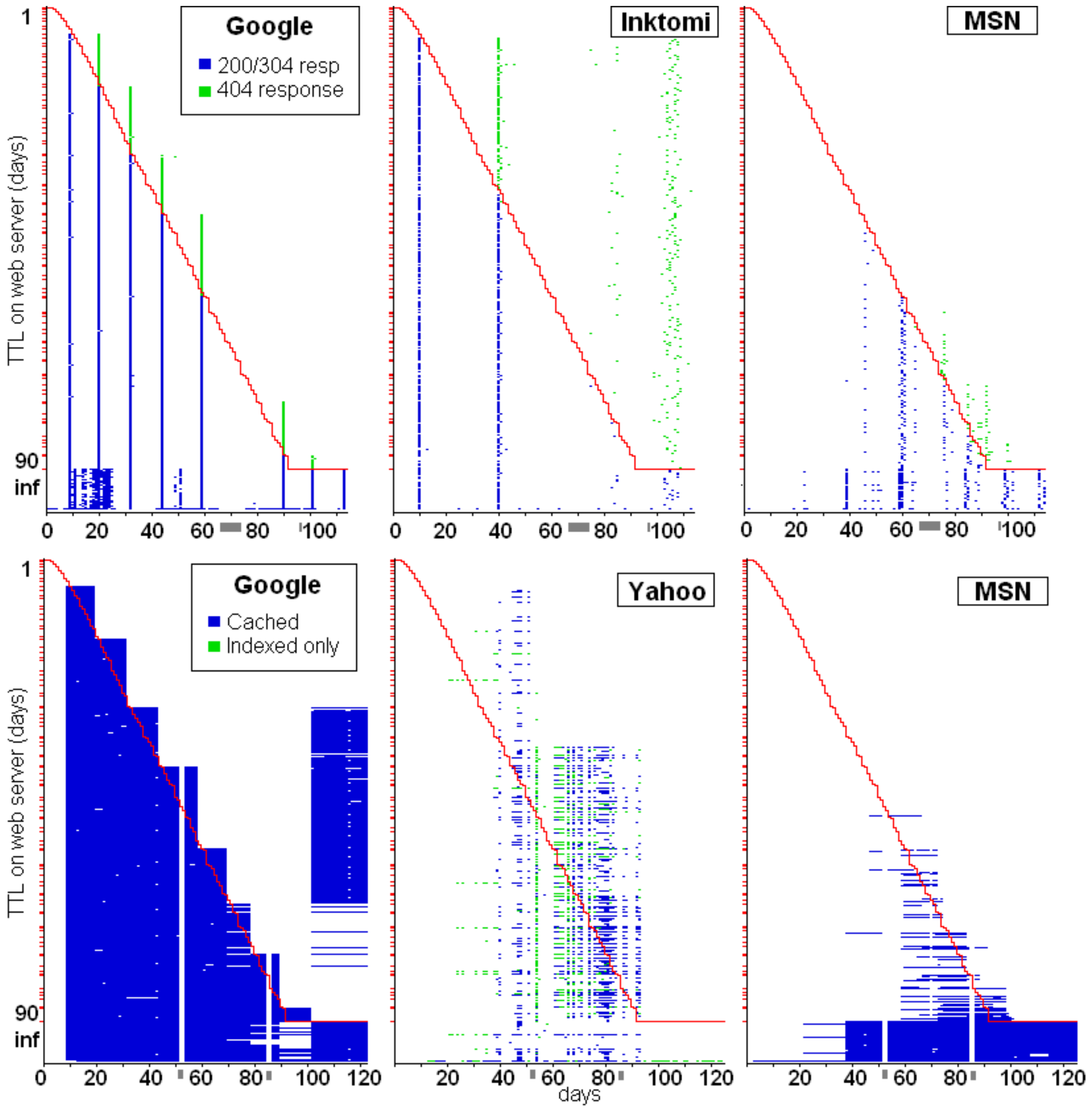


FIG. 15: Crawling (top) and caching (bottom) of HTML resources from the mln web collection.

exhibited similar behavior for HTML resources. HTML and PDF resources from owenbrau appeared in the Google cache on day 102 for the first time; these resources had been deleted from the web server 10-20 days before day 102. Manual inspection weeks after the experiment had concluded revealed that the pages remained in Google's cache and only fell out months later.

Yahoo was very sporadic in caching resources; there was often a lag time of 30 days between the crawl of a resource and its appearance in cache. Many of the crawled resources never appeared in Yahoo's cache. Although Inktomi crawled nearly every available HTML resource on day 10, only half of those resources ever became available in the Yahoo cache. In subsequent interaction with Yahoo, it has been observed that links to cached content may appear and disappear when performing the same query just a few seconds apart. This likely accounts for the observed cache inconsistency.

3 WEB INFRASTRUCTURE CONTENTS

In this second experiment, the contents of the WI are examined. An experiment was designed to examine the distribution of contents held in search engine caches and the overlap of those contents with the Internet Archive.

3.1 Methodology

The four most popular search engines that cache content were investigated: Google, MSN, Yahoo and Ask. In the previous experiment, only Google had a web search API, but at the time this experiment was executed, MSN and Yahoo had both released an API. Therefore the APIs provided by Google, MSN and Yahoo were used to access their search results and page scraping the WUI was used to access Ask's search results (Ask does not provide a free API).

In February 2007, 5200 one-term queries (randomly sampled from an English dictionary) were submitted to each search engine, and one of the first 100 results were randomly selected. An attempt was then made to download the selected URL from the Web and also the cached resource from the search engine. Finally, IA was queried to see how many versions of the URL it had stored for each year, if any. All search engine responses, HTTP headers, web pages, cached pages and IA responses were stored for later processing.

The sampling method produced several biases since it favors pages in English, long and content-rich pages which are more likely to match a query than smaller documents, and those pages that are more popular than others. New methods [16, 17] have recently been developed to reduce these biases when sampling from search engine indexes and could be used in future experiments.

3.2 Accessibility of Indexed Resources

Table 7 lists the percent of resources from each search engine that were replicated (cached and accessible from the Web), endangered (cached but not accessible from the Web), vulnerable (accessible from the Web but not cached) and unrecoverable (missing from the Web and cache). Resources were judged to be accessible from the Web if they were successfully retrieved with an HTTP 200 response, and inaccessible if any other HTTP response was received.

TABLE 7: Web and cache accessibility.

	Cached		Not cached		Miss
	Web (Replicated)	Missing (Endangered)	Web (Vulnerable)	Missing (Unrecoverable)	rate
Ask	8.2%	0.4%	77.5%	13.9%	0.05%
Google	76.2%	4.3%	18.4%	1.1%	3.88%
MSN	88.7%	4.5%	5.0%	1.8%	0.01%
Yahoo	76.5%	3.7%	17.9%	2.0%	1.53%

Less than 9% of Ask's indexed contents were cached (adding the Cached-Web and Cached-Missing columns together), but the other three search engines had at least 80% of their content cached. Over 14% of Ask's indexed content could not be successfully retrieved from the Web, and since most of these resources were not cached, the utility of Ask's cache is questionable. Google, MSN and Yahoo had far less missing content indexed, and a majority of it was accessible from their cache. However, both Google and Yahoo had a rather high percentage of resources that were vulnerable to being lost (18% for both).

The miss rate column in Table 7 is the percent of time the search engines advertised a URL to a cached resource but returned an error page when the cached resource was accessed. Ask and MSN appeared to have the most reliable cache access (although Ask's cache is very small). Note that Google's miss rate was probably higher because Google's API does not advertise a URL to the cached resource; the only way of knowing if a resource is cached or not is to attempt to access it.

3.3 Top Level Domain

Figure 16 shows the distribution of the top level domains (TLDs) of the sampled URLs from each search engine's index (only the top 15 are shown). These distributions are very similar to those in [16]. All four search engines tend to sample equally from the same TLDs with .com being the largest by far.

3.4 Content Type

Table 8 shows the distribution of resources sampled from each search engine's index (*Ind* column). The percent of those resources that were extracted successfully from cache is given under the *Cac* columns. HTML was by far the most indexed of all resource types. Google, MSN and Yahoo provided a relatively high level of access to all cached resources, but only 10% of HTML and 11% of plain text resources could be extracted from Ask's cache, and no other content type was found in their cache.

Several media types were found indexed (but not cached) that are not typically indexed by search engines. Two videos were indexed in Google using the Advanced Systems Format (ASF), and an audio file (MPEG) and Flash file indexed by Yahoo. Several XML resource types were also discovered (and some cached): XML Shareable Playlist (Ask), Atom (Google) and RSS feeds (Ask,

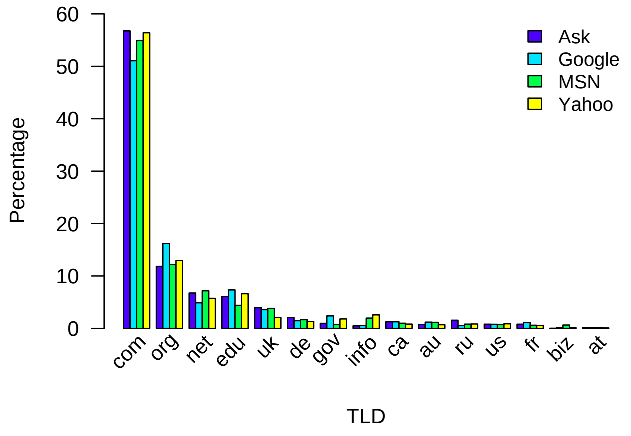


FIG. 16: Distribution of TLDs from sample.

TABLE 8: Indexed and cached content by type.

	Ask		Google		MSN		Yahoo	
	Ind	Cac	Ind	Cac	Ind	Cac	Ind	Cac
HTML	94%	10%	88%	81%	96%	95%	94%	80%
PDF	2%	0%	7%	69%	3%	89%	4%	92%
Plain text	4%	11%	3%	93%	1%	95%	1%	96%
MS Office	0%	0%	0.7%	76%	0.4%	73%	0.6%	100%
Other	3%	0%	8%	69%	3%	89%	4%	92%

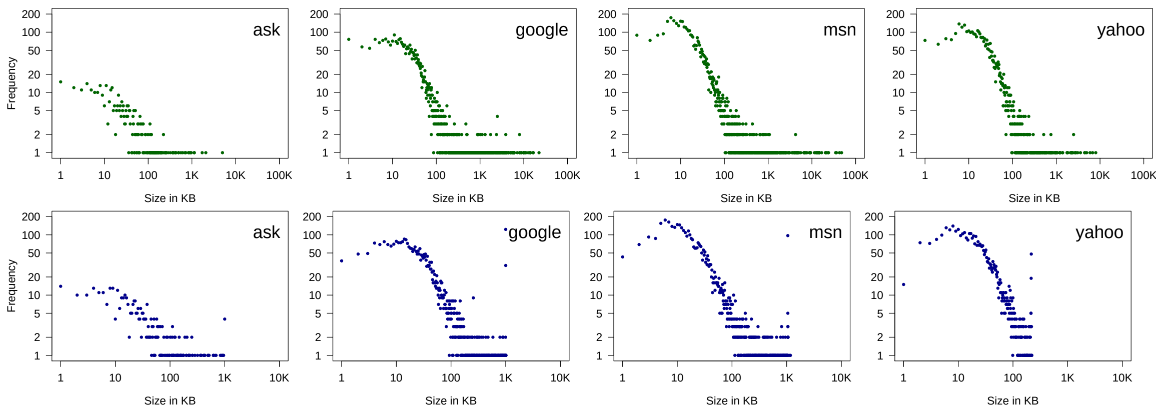


FIG. 17: Distribution of Web file sizes (top) and cached file sizes (bottom) on log-log scale. Web file size means: Ask = 88 KB, Google = 244 KB, MSN = 204 KB, Yahoo = 61 KB. Cached file size means: Ask = 74 KB, Google = 104 KB, MSN = 79 KB, Yahoo = 34 KB.

Google and Yahoo), and OAI-PMH responses (Ask and Google). No XML types were found in MSN.

3.5 File Sizes

Figure 17 plots the file size distribution of the live web resources (top row) and cached resources (bottom row). The graphs use log-log scale to emphasize the power-law distribution of page sizes which has been observed on the Web [13]. Before calculating the cached resource size, each resource was stripped of the search engine header. All four search engines appeared to limit the size of their caches. The limits observed were: Ask: 976 KB, Google: 977 KB, MSN: 1 MB and Yahoo: 215 KB. Despite the limits, only 3% of all resources were truncated. On average, Google and MSN indexed and cached the largest web resources.

3.6 Cache Directives

As discussed in Chapter II, search engines and IA use an opt-out policy approach to caching and archiving. All crawled resources are cached unless a web master uses the Robots Exclusion Protocol (robots.txt) to indicate URL patterns that should not be indexed (which also prevents them from being cached) or if `noarchive` meta tags are placed in HTML pages. There is currently no mechanism in place to permit a search engine to index a non-HTML resource but not cache it.

Only 2% of the HTML resources from the Web used `noarchive` meta tags; 6% specifically targeted googlebot, and 96% targeted all robots (none were targeting the other three search engines). Only a hand-full of resources were found with `noarchive` meta tags that were cached by Google and Yahoo, but it is likely the tags were added after the search engine crawlers had downloaded the resources since none of the tags were found in the cached resources.

HTTP 1.1 has a number of cache-control directives that are used to indicate if the requested resource is to be cached, and if so, for how long. Although these directives were not originally

TABLE 9: Staleness of search engine caches (in days).

	% Stale	Mean	Median	Std	Min	Max
Ask	16%	7.5	12.0	5.5	2	25
Google	20%	15.1	7.0	26.2	1	343
MSN	12%	6.7	5.0	5.2	1	27
Yahoo	17%	92.1	17.0	220.9	1	1422

intended to address the availability of resources in a search engine cache, some argue that perhaps they should [20]. One quarter (24%) of the sampled resources had an HTTP header with Cache-Control set to no-cache, no-store or private, and 62% of these resources were cached. None of the search engines appeared to respect the cache-control directives since all four cached these resources at the same rate as resources without the header.

3.7 Cache Freshness

A cached copy of a web resource is *fresh* if the resource has not changed since the last time it was crawled and cached. Once a resource has been modified, the cached resource becomes *stale* (or ages [36]). The staleness of the cache increases until the search engine re-crawls the resource and updates its cache.

To measure the staleness the of caches, the Last-Modified HTTP header of the live resource from the Web was compared to the date the resource was cached. Although most web servers do not return last modified dates for dynamically produced resources, and some return incorrect values [40], this is the best known technique for determining when a resource was last modified. Another difficulty of determining staleness is that cached dates are not available for all cached resources; Google only reports cached dates for HTML resources, and Yahoo only reports last modified dates through their API which may not be the same as when the resource was actually cached.

Staleness was calculated (in days) by counting the number of days that had passed from the cached date to the last modified date. If the cache date was more recent the last modified date (the resource was cached after it was last modified), a value of 0 was assigned to staleness. More formally:

$$\text{staleness} = \max(\text{last modified date} - \text{cached date}, 0) \tag{15}$$

Only 46% of the live pages had a valid HTTP Last-Modified timestamp, and of these, 71% also had a cached date. A majority of the resources (84%) were up-to-date. The descriptive statistics for resources that were at least one day stale are given in Table 9, and the distribution is shown in Figure 18. Although Google had the largest amount of stale cached pages, Yahoo's pages were on average more stale. MSN had the fewest amount of stale pages and nearly the most up-to-date set of pages.

Nineteen percent of the cached resources were identical to their live counterparts, 21% if examining just HTML resources. To measure the similarity between two resources that were not identical, the percentage of shared shingles (shingle size of 10) was calculated. Shingling [29] is a popular

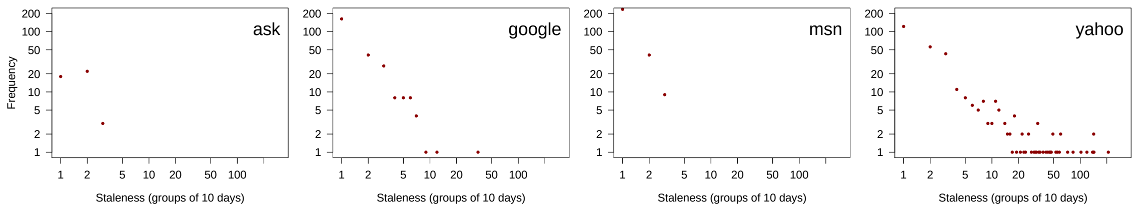


FIG. 18: Distribution of staleness on log-log scale.

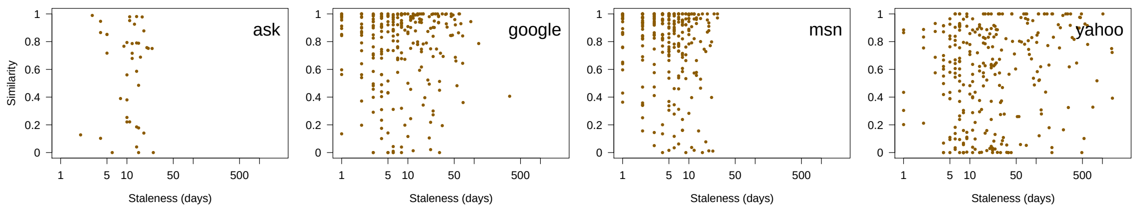


FIG. 19: Scatterplots of similarity vs. staleness (x-axis is on a log scale).

technique for quantifying similarity of text documents when word-order is important. Before calculating shingles, all HTML (if present) was stripped from both resources so only textual content was compared. On average, resources shared 72% of their shingles. This implies that although most web resources are not replicated in caches byte-for-byte, most of them are very similar to what is cached.

The relationship between similarity and staleness was explored by examining the two values in a scatterplot (Figure 19). The busy scatterplots indicate there is no clear relationship between similarity and staleness; a cached resource is likely to be just as similar to its live Web counterpart if it is one or 100 days stale.

3.8 Search Engine Overlap with Internet Archive

A question that may be raised is, "How much of what the search engines have cached is also archived by the Internet Archive?" Is there a large overlap? Figure 20 shows a Venn diagram illustrating how some resources held by the IA are indexed by search engines (I) and/or cached (II). But there are some indexed (IV) and cached (III) resources that are not available in the IA.

As mentioned earlier, the IA was queried with each URL sampled from the search engines. Table 10 shows the overlap of sampled URLs within IA. MSN had the largest overlap with IA (52%) and Yahoo the smallest (41%). On average, only 46% of the sampled URLs from all four search engines were available in IA.

Figure 21 plots the distribution of the archived resources. There is nearly an exponential increase each year except in 2006, likely because at the time of this experiment there was a lag of 6-12 months before IA made new content available in their archive. The hit-rate line in Figure 21 is the percent of time the IA had at least one resource archived for that year. It is interesting to note that although the number of resources archived in 2006 was half that of 2004, the hit rate of 29% almost matched

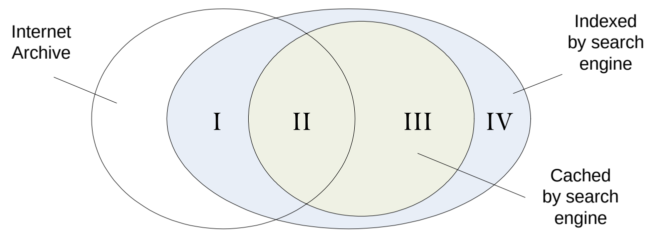


FIG. 20: Venn diagram showing overlap of search engine caches with IA.

TABLE 10: Search engine overlap with the Internet Archive.

	In IA		Not in IA	
SE	Cached (II)	No cache (I)	Cached (III)	No cache (IV)
Ask	9.2%	36.0%	0.3%	54.5%
Google	40.7%	3.7%	50.3%	5.3%
MSN	51.1%	1.1%	43.7%	4.1%
Yahoo	39.3%	1.8%	47.7%	11.2%

2004's 33% hit rate.

4 DISCUSSION

The search engines components of the WI were observed to behave very differently when caching four synthetic web collections. Although Google crawled and cached all new HTML and PDF resources placed on known websites several days after they were made accessible on the Web, resources on a new website were not cached for months. Yahoo and MSN were 4-5 times slower than Google to acquire new resources, and Yahoo incurred a long transfer delay from Inktomi's crawls into their cache. Images tended to be largely ignored by all three search engines.

It was also observed that cached resources were often purged from all three caches as soon as a crawl revealed the resources were missing, but in the case of Google, many HTML resources reappeared weeks after being removed. Google's odd behavior may be contributed to internal transferring of crawl data to more permanent data stores, and it is unknown if the behavior is typical or repeatable. Moreover, crawling and caching policies may also change over time. Nevertheless, this experiment provides a glimpse into caching behavior of the top three search engines that has not been documented before. The findings suggest that search engines vary greatly in the level of access they provide to cached resources, and that websites are likely to be reconstructed more successfully if they are reconstructed quickly after being lost. Reconstructions should also be performed several days in a row to ensure maximum access to web repository holdings. In some cases, it may even be beneficial to attempt recovering resources months after they have been lost.

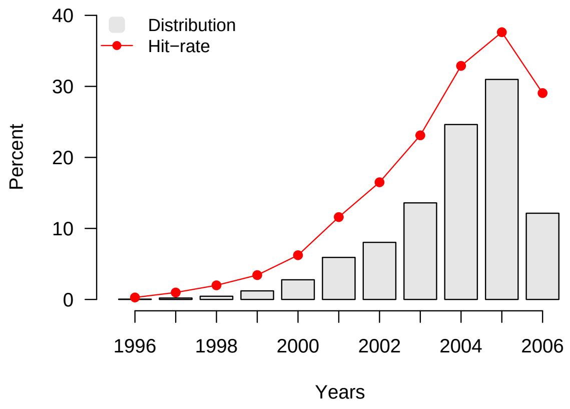


FIG. 21: Distribution and hit rate of sampled URLs with IA.

The second experiment showed that from a digital preservation perspective, Ask was of limited utility; it had the fewest resources cached (9%) and a majority of the resources were vulnerable (78%) or unrecoverable (14%). Top level domains appeared to be represented in all four search engine caches with roughly the same distribution, so no single TLD appeared to be favored more than another. `Noarchive` meta tags were infrequently used (2%) in sampled HTML resources, and search engines did not appear to respect HTTP cache-control headers, two advantages from a preservation perspective. All search engines seemed to have an upper bound on cached resources of about 1 MB except for Yahoo which appears to have an upper bound of 215 KB; however, this only affected 3% of all cached resources. The staleness of the cached resources ranged from 12% (MSN) to 20% (Google), and median staleness ranged from 5 days (MSN) to 17 days (Yahoo).

While the IA provides a preservation service for public web pages, its limitations of crawling frequency and 6-12 month delay in processing crawled resources limits its effectiveness. In the first experiment, IA did not crawl or archive a single resource from the four decaying web collections. In the second experiment, IA was found to contain only 46% of the resources available in search engine caches. IA has recently made improvements in their time-to-ingest [83], thus minimizing the delta between t_d and t_a (Figure 12). This improvement will be evident later in Chapter VIII. Therefore, if these experiments were repeated, it is likely that IA's performance would improve.

5 CONCLUSIONS

This chapter investigated the WI's behavior at finding and storing decaying web content and has examined the types of content found in the WI. The three search engines showed very different crawling and caching behaviors, and IA was unable to store any of the decaying content. An overlap analysis showed IA only contained 46% of the resources that were found cached in the search engines. These experiments have verified that the Web is too large and too dynamic for a single entity to capture everything. Chapter VII will emphasize this point by demonstrating how widely the repositories contribute to various website reconstructions.

CHAPTER V

WEB-REPOSITORY CRAWLING

Web crawlers are programs that systematically download web pages from the Web, and they have been around nearly as long as the Web has been in existence. As the Web has increased in size and diversity, so have the types of crawlers that have been developed. In this chapter, a new type of crawler is introduced: the *web-repository crawler*. Instead of looking for web resources to download from the Web, a web-repository crawler searches for resources in the vaults of web repositories.

1 CRAWLER ARCHITECTURE

A web-repository crawler shares many characteristics with a traditional web crawler, but there are a number of key differences which make crawling web repositories fundamentally different than crawling the Web [110]. The left side of Figure 22 illustrates the architecture of a simple web crawler [37]. The crawler is initialized with one or more seed URLs which are placed into the crawl frontier. URLs are extracted from the frontier, downloaded from the Web and (usually) stored in a local repository. HTML resources (and sometimes other resource types) are searched for additional URLs which are canonicalized and placed on the frontier if they have not yet been visited. This process usually continues until the frontier is empty.

Figure 22 (right) shows a similar architecture for a web-repository crawler. A web-repository crawler must also maintain a frontier and list of visited URLs, but instead of downloading from the Web, resources are extracted from web repositories. This requires a web-repository crawler to decide between multiple versions of a resource when more than one are found with the same URI. The canonical version of a resource may be chosen over a non-canonical version in some cases (e.g., PDF from a web archive vs. HTMLized version from a search engine cache), or the most recently stored version may be chosen instead of an older version. URL canonicalization is also necessary for a web-repository crawler, but it is more difficult to do because the same resource may be represented by different URIs in different repositories (Section 3 will discuss these issues in greater detail).

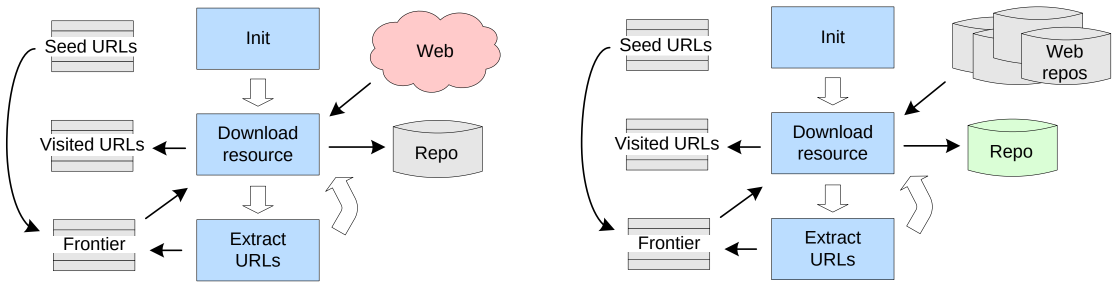


FIG. 22: Architecture of a traditional web crawler (left) and web-repository crawler (right).

Just as a web crawler "respects" a web server by delaying between requests and issuing a limited number of requests in a particular time interval, a web-repository crawler may also limit daily requests to web repositories. In some cases, query limits may be imposed by the repository as it is with search engine APIs that only allow a limited number of queries in a 24 hour period. For other repositories, a limit can be selected that is in line with limits used by other repositories. More formally, the number of queries issued to a web repository in a time period P should be less than or equal to L. If the crawler makes L queries during P, it should sleep until P has elapsed.

Another similarity between web crawlers and web-repository crawlers can be seen in how they find resources from their crawling target. Although web crawlers have traditionally been unable to ask a web server, "What is the set of URLs on your website?" [27], newer methods like the Sitemaps Protocol [155] (supported by Google, Live and Yahoo) and mod_oai [126] have been designed to give this functionality to web crawlers. Similarly, web repositories can be asked to list the set of URLs they have stored for a particular website using lister queries (discussed next).

2 LISTER QUERIES AND CRAWLING POLICIES

When web repositories do not support lister queries (Section 3.5 of Chapter II), numerous requests for non-existing resources may greatly slow down a web-repository crawler. For example, if the crawler is reconstructing a website that has no resources stored in a particular repository, the crawler will make numerous `getResource` queries to the repository, wasting the limited number of daily queries allocated to the crawler. This naïve behavior can be avoided if a repository supports lister queries, since the crawler will only make requests for resources it knows in advance are stored in the repository.

Another advantage of lister queries is that a repository crawler may find more resources from a lost website than if it were to only examine links in recovered pages. For example, suppose the graph on the left side of Figure 23 represented the resources (nodes) and links (edges) of website W. If the website W 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, they could be recovered as shown on the right of Figure 23, even though they are not directly connected to W'.

A disadvantage of recovering all resources discovered through lister queries is that potentially many queries could be spent recovering old and useless resources that no longer make-up the reconstructed website. This could happen for a site that changes its link structure frequently. Additionally, resources that belong to other subsites may be falsely aggregated into a single site, so resources are recovered that the recover is not interested in.

The ability to perform lister queries permits a web-repository crawler to reconstruct websites using one of three crawling policies [115]:

1. **Naïve Policy** - Do not issue lister queries and only recover resources found by links from recovered pages.

2. **Knowledgeable Policy** - Issue lister queries but only recover resources found by links from recovered pages.

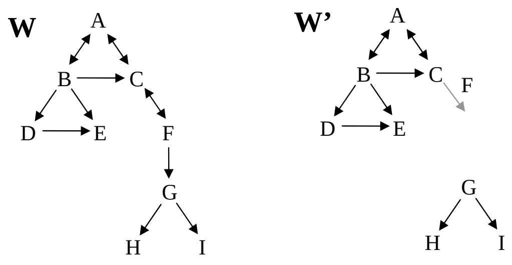


FIG. 23: Website W (left) and reconstructed website W' (right) that is missing resource F.

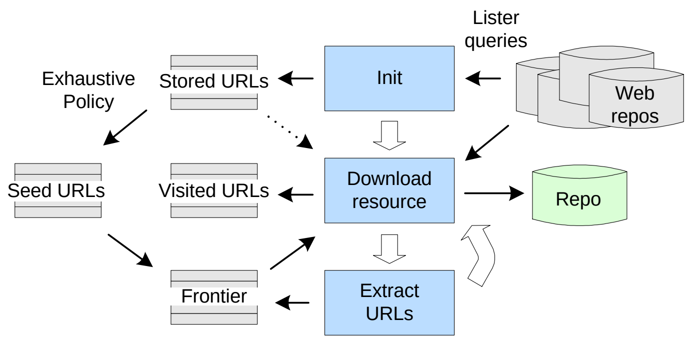


FIG. 24: Architecture of a web-repository crawler using lister queries.

3. **Exhaustive Policy** - Issue lister queries and recover all resources found in all repositories.

When web repositories support lister queries, an additional database must be maintained to support the lister query responses. Figure 24 shows the addition of `Stored URLs` which may be fed to the `Seed URLs` if the crawler is using the Exhaustive Policy. The `Stored URLs` database is referenced when choosing which repositories to download a resource from.

3 URL CANONICALIZATION

3.1 Tracking Visited URLs

In order to effectively keep track of which URLs have been visited, a web-repository crawler must be able to distinguish between two syntactically different URLs that point to the same resource. For example, the URLs `http://www.Foo.org/bar.html#abc` and `http://www.foo.org/../bar.html` resolve to the same resource. The crawler resolves these differences by canonicalizing (or normalizing) each URL it extracts from recovered HTML pages. The following rules are widely employed by all types of crawlers when performing URL canonicalization [23, 97, 136] and should also be adopted by a repository crawler:

- Convert the domain name to lower case. Additionally, convert the entire URL to lowercase if the web server's file system is case insensitive.
  Original URL: `http://FOO.ORG/BAR.html`
  Canonical URL: `http://foo.org/BAR.html`

- Remove index.html, index.htm and default.htm from the end of a URL.
  Original URL: `http://foo.org/index.html`
  Canonical URL: `http://foo.org/`

- Remove the fragment (section link) from the URL.
  Original URL: `http://foo.org/bar.html#section1`
  Canonical URL: `http://foo.org/bar.html`

- Remove all instances of '/../' and '/./' from the URL by collapsing it.
  Original URL: `http://foo.org/../a/b/../c/./bar.html`
  Canonical URL: `http://foo.org/a/c/bar.html`

- Remove multiple occurrences of slashes in the URL before the query string.
  Original URL: `http://foo.org//a///b`
  Canonical URL: `http://foo.org/a/b`

- Convert all hex-encoded characters (%XX) in the range [0x20 to 0x7E] before the query string to standard ISO-8859-1 (Latin-1) characters (using '+' for spaces), and capitalize all remaining hex characters.
  Original URL: `http://foo.org/%7Ebar/a%20b/?test=g%20%8a`
  Canonical URL: `http://foo.org/~bar/a+b/?test=g%20%8A`

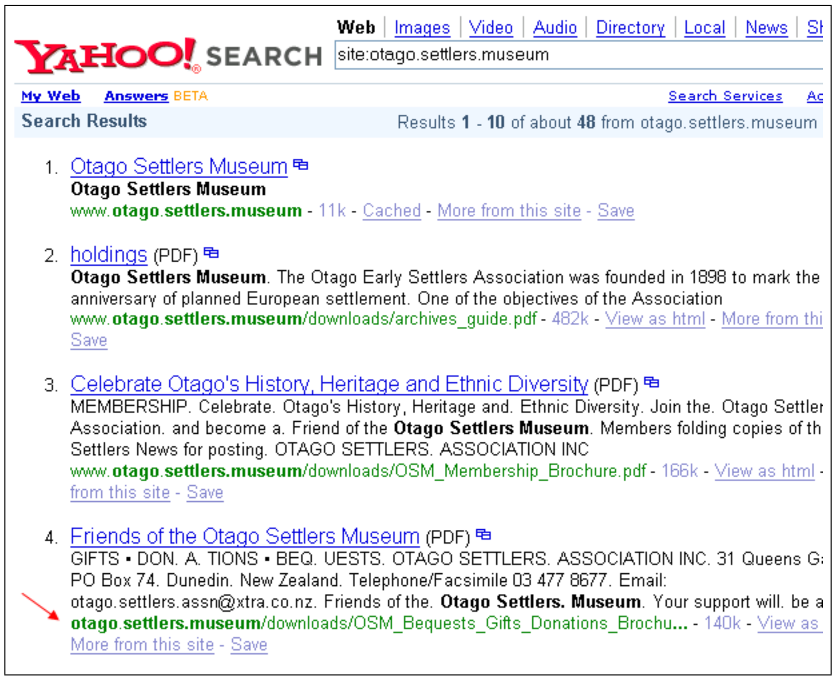
- Remove common session IDs.
  Original URL: `http://foo.org/?page=123&jsessionid=999A9EF028317A82AC83F0FDFE59385A`
  Canonical URL: `http://foo.org/?page=123`

3.2 Harmonizing Different Repository URL Canonicalization Rules

Each web repository may perform URL canonicalization in different ways when crawling the Web [115]. For example, one repository may canonicalize the URL `http://www.foo.org/index.html` as `http://www.foo.org/` and another as `http://foo.org/index.html`. This presents a number of difficulties to a web-repository crawler when trying to determine if a repository has stored a particular URL. Lister queries are useful for resolving some of the problems created by repositories using different canonicalization policies. Several canonicalization issues are discussed next.

'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


YAHOO! SEARCH
Web | Images | Video | Audio | Directory | Local | News | Sh
My Web Answers BETA
Search Services Ac

site:otago.settlers.museum

Search Results Results **1** - **10** of about **48** from otago.settlers.museum

1. Otago Settlers Museum
Otago Settlers Museum
www.**otago.settlers.museum** - 11k - Cached - More from this site - Save

2. holdings (PDF)
Otago Settlers Museum. The Otago Early Settlers Association was founded in 1898 to mark the anniversary of planned European settlement. One of the objectives of the Association
www.**otago.settlers.museum**/downloads/archives_guide.pdf - 482k - View as html - More from thi
Save

3. Celebrate Otago's History, Heritage and Ethnic Diversity (PDF)
MEMBERSHIP. Celebrate. Otago's History, Heritage and. Ethnic Diversity. Join the. Otago Settler
Association. and become a. Friend of the **Otago Settlers Museum**. Members folding copies of th
Settlers News for posting. OTAGO SETTLERS. ASSOCIATION INC
www.**otago.settlers.museum**/downloads/OSM_Membership_Brochure.pdf - 166k - View as html -
from this site - Save

4. Friends of the Otago Settlers Museum (PDF)
GIFTS • DON. A. TIONS • BEQ. UESTS. OTAGO SETTLERS. ASSOCIATION INC. 31 Queens G
PO Box 74. Dunedin. New Zealand. Telephone/Facsimile 03 477 8677. Email:
otago.settlers.assn@xtra.co.nz. Friends of the. **Otago Settlers. Museum**. Your support will. be a
otago.settlers.museum/downloads/OSM_Bequests_Gifts_Donations_Brochu**...** - 140k - View as
More from this site - Save

FIG. 25: Yahoo search results for `site:otago.settlers.museum`.

code. In fact, Google and other search engines request this behavior to simplify web crawling [46]. Google, Live 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 25. This figure shows 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.

Case Insensitivity

Although section 6.2.2.1 of RFC 3986 states the path component of a URI should be case-sensitive [23], 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 web server's 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`. Live 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 mining HTML resources into lowercase so case is no longer an issue.

Missing the Terminating Slash

When extracting links from an HTML page to recover, some URLs that point to a directory may lack the proper terminating slash (cf. RFC 3986 section 6.2.3). 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. But 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, Live 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 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`, it may be inferred that the URL is pointing to a directory since Google's canonicalization policy dictates proper terminal slashes for directories. But if Yahoo is the only repository storing the resource `http://foo.org/bar`, the crawler must arbitrarily decide to either treat the URL as a directory or not.

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. And while the URL may be pointing to index.html at time t_1, it may point to default.htm at time t_2. When recovering a URL pointing to the root level of a directory, a web-repository crawler cannot automatically discover the name of the file the URL was pointing to when the resource was crawled by the repository. When a link pointing to `http://foo.org/bar/index.html` directly is discovered in a recovered resource, the crawler may infer that `http://foo.org/bar/` is referring to index.html, but it 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. Live 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 Live reports that it has a URL stored as `http://foo.org/bar/index.html`, the crawler may assume that this URL refers to `http://foo.org/bar/` (although this is not *always* true).

Other Canonicalization Rules

Bar-Yossef et al. [18] have developed an algorithm that can be applied to a list of URLs to determine which URLs are pointing to the same content without actually examining the content. The algorithm could be applied by a web-repository crawler to the final set of URLs returned by lister queries. Here are a few examples of duplicate URLs that could be resolved:

- `http://foo.edu/~bar/` = `http://foo.edu/people/bar/`
- `http://foo.com/story?id=num` = `http://foo.com/story_num`
- `http://foo.com/news` = `http://news.foo.com/`

3.3 Theoretical Bounds on Website Reconstruction

Cost to Web Repositories

The collective cost incurred by the web repositories for reconstructing a website is the total number of queries they must respond to from a web-repository crawler. The query cost C can be defined as the total number of lister queries needed to discover all the resources from all n repositories plus the total number of queries needed to download r resources from n web repositories. More formally, $C(r)$ is:

$$C(r) = \sum_{i=1}^{n} \lceil S(i)/R(i) \rceil + r \sum_{i=1}^{n} Q(i) \tag{16}$$

where $S(i)$ is the number of resources stored in repository i, $R(i)$ is the number of results the repository will return to a single lister query, and $Q(i)$ is the number of queries required to download the resource from the repository. The $S(i)$ and r variables are usually not known in advance, but all other variables usually are.

Each of these variables varies from repository to repository and can sometimes be difficult to calculate. For example, $R(i)$ for IA might be one if the direct URL to a resource was obtained from the lister queries, but often two queries are required, one to ask for the set of URLs stored for the resource, and another to download the resource. Also, IA sometimes advertises direct URLs for a resource, but when accessed, the resource is not present, and another query for an older version of the resource must be made.

The variable $Q(i)$ also varies depending if lister queries have revealed all known resources from the repository being queried. For example, if a large website is being reconstructed and more than 1000 resources are stored in Yahoo, then the crawler will not know if a given URL which does not appear in the first 1000 results is stored in Yahoo or not; it will have to query Yahoo twice, once to see if the resource is stored in Yahoo and another to download the resource.

Daily Query Limits

As mentioned in Section 1, web-repository crawlers normally respect web repositories by limiting the number of queries they make in a given time period. Search engines like Google, Live and Yahoo provide APIs for accessing their contents which limit the number of queries a client can make in a 24 hour span. Because a web-repository crawler must crawl in round-robin style where each repository is queried for the same resource before moving on to the next resource, the repository with the fewest number of allotted queries is often the one limiting the number of resources that can be recovered in a day. More formally, the maximum queries M that can be issued per day is:

$$M = \min(limit_1, limit_2, ..., limit_n) \tag{17}$$

where $limit_i$ is the daily limit of each web repository that contains resources that need to be recovered. For the repositories {Google, Live, Yahoo, IA} implemented in Warrick, the limits are {1000, 10,000, 5000, 1000}. Therefore Google and IA are usually the repositories limiting the overall recovery rate.

The total days T to reconstruct a website is then the total number of queries to recover all resources divided by the maximum number of queries that can be issued each day:

$$T = C(r)/M \tag{18}$$

4 CONCLUSIONS

A web-repository crawler shares many characteristics with a regular web crawler; the differences primarily lie in how to canonicalize URLs between various repositories, how to crawl a repository, and how to choose the “best” resource between several versions of the same resources. The next chapter introduces Warrick, an instantiation of a web-repository crawler which has been used by the general public to reconstruct a number of lost websites.

CHAPTER VI

WARRICK, A WEB-REPOSITORY CRAWLER

The preceding chapter laid the groundwork for developing a web-repository crawler. In this chapter, the Warrick crawler is introduced, and implementation details are presented. A queueing system called Brass was developed for Warrick in early 2007 which has allowed the public at large to run Warrick without downloading and installing it locally. Usage statistics for Warrick and Brass indicate a large number of individuals are interested in recovering websites from the WI.

1 BRIEF HISTORY

Inspired by discussions with Dr. Michael Nelson and Dr. Johan Bollen, Warrick was created in the fall of 2005 by the author for demonstrating how a "website reconstructor" could recover pages from the Internet Archive, Google, MSN and Yahoo for a "lazy" webmaster who lost their website. Dr. Nelson named the tool Warrick after a fictional forensic scientist from the CSI television show who had a penchant for gambling– someone too lazy to backup their website is taking a gamble if they are relying on Google to do it for them.

The first opportunity to use Warrick to reconstruct a lost website came in October 2005 when a fire at the the University of Southampton rendered the 15th International World Wide Web 2006 Conference website (`www2006.org`) inaccessible [59]. The website was lost one week before the deadline for paper submissions. The author ran Warrick and was able to recover 77 resources, 36 from Google, 3 from Yahoo and 38 from MSN. After contacting the conference organizers, the author 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 [115]: "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 months later, the author was made aware of a tool developed by Aaron Swartz called arcget [160] which could recover a website from the Internet Archive. Swartz was unaware of Warrick at the time, so he created arcget to extract pages for an old website called the Lingua Franca Archive, an archive of scholarly review articles. Swartz later re-hosted the recovered website at `http://linguafranca.mirror.theinfo.org/`.

In January 2006, the author placed Warrick online for others to download and use. A few weeks later, an individual named Dwight (not his real name) used Warrick to reconstruct two websites that he lost when his web server's filesystem crashed. One site was Dwight's personal blog, and the other an adult kickball league website. After losing his websites, Dwight made clear his admonition to others who had not backed-up their data [107]:

"The main point that I want to get across is this: BACK UP YOUR DATA! The shock

TABLE 11: Repository request methods and limits.

Web repository	Request method	Daily limit
Internet Archive	WUI	1000
Google	Google API or WUI	1000
Yahoo	Yahoo API	5000
Live Search	Live Search API	10,000

> of losing a year's worth of blood and sweat (regarding the code that powered [my site]) still has yet to fully sink in."

Although Warrick was not able to recover all of Dwight's blog, he expressed some relief at getting back something he thought was forever lost:

> "It's unclear how many posts never got recovered with Warrick in the first place. Eyeballing it, I'd say I have at least 80% of my posts. And you know what? I'll take that."

These were the first two websites reconstructed by someone outside of ODU.

Warrick continued to be improved, and instructions for downloading, installing and running Warrick were expanded, but the author continued to receive emails and phone calls from individuals who were not technically inclined and needed help recovering their lost websites. An online queueing system named Brass[1] was finally developed with the help of Amine Benjelloun, a graduate student at Old Dominion University. An easy-to-use front end was developed to make the reconstruction process simple. Brass went online on July 2, 2007, with five nodes running reconstruction jobs. More details about Brass will be discussed in Section 5.

2 IMPLEMENTATION

Warrick was written in the Perl programming language because of its powerful string manipulation capabilities and the availability of a wide-range of libraries on the CPAN network (`www.cpan.org`). Warrick requires a complete Perl 5 installation along with a few supplemental libraries. Anyone can download and install Warrick from `http://warrick.cs.odu.edu/`, and the code may be modified and freely distributed since it is licensed under the GNU General Public License.

Warrick uses the four repositories IA, Google, Live and Yahoo. The search engine APIs are used, and page-scraping the web user interface (WUI) is used for IA. Warrick also will page-scrape the WUI for Google if the Warrick user does not have a Google API key (keys do not need to be obtained for using the Live and Yahoo APIs). The search engine APIs allow only a limited number of daily queries from distinct IP addresses or from certain keys which Warrick must adhere to (Table 11). Although IA does not publish a request limit, Warrick does not make more than 1000 requests per day per IP address, the same limit used by the Google API.

[1]Brass is Warrick's boss on the CSI television show.

3 OPERATION

Warrick follows the algorithm presented in Figure 3 to reconstruct a website. A starting URL is supplied by the user which is typically the root URL for the website (e.g., `http://foo.edu/` or `http://foo.edu/~bar/`).

If the Knowledgeable or Exhaustive policy is being used, Warrick issues lister queries to each repository using the base URL (crawling policies were introduced in Section 4 of the previous chapter). Lister queries must be performed on two different interfaces since all three of the flat repositories (search engines) use separate interfaces for accessing their images. The returned URLs are canonicalized and saved for later getResource queries, and if the Exhaustive policy is being used, the URLs are placed on the frontier. Note that since search engines will not return more than 1000 results, Warrick must keep track of repositories that have potentially more resources stored in them. This information is used later when determining if a repository should be queried for a resource.

When recovering each URL in the frontier, Warrick must initially decide if the URL is pointing to an image or not because of the separate image interfaces for the flat repositories. Warrick examines the ending of the URL, and if it ends with a known image type (e.g., .gif, .jpg, .png, etc.) then the image interfaces are queried. Taking advantage of the fact that IA stores canonical images and the datestamp of images cannot be obtained from the flat repositories, IA is queried first, and if the image is not found, then each flat repository is queried until one image is found.

If a non-image resource is being recovered, the resource is downloaded from each repository since the crawl date of the resource is usually only found by examining the HTML that the repositories add to the heading of the cached resource. Warrick uses an optimization that is not shown in the algorithm: since IA is several months out-of-date, the search engine caches are searched first for HTML resources since they are more likely to have the most recent version of an HTML resource and since they all store the canonical HTML resource. But when non-HTML resources are being searched, IA is examined first since it contains canonical resources for PDFs, Office documents, etc. This optimization may save a significant number of queries.

Once the canonical or most recent version of the resource is found, it is saved using a filename and path that mirrors the URL. For example, the resource `http://foo.org/bar/hello.html` is stored as `foo.org/bar/hello.html`. HTML resources are examined for links to other potentially missing resources from the website, and after being canonicalized, URLs which have not been visited (and are not already in the frontier) are added to the frontier. If any of the repository query limits have been exceeded, Warrick sleeps for 24 hours. Once it awakens, Warrick has another full set of queries to exhaust.

Warrick creates a **reconstruction summary file** that lists the URLs that were successfully or unsuccessfully recovered. For each resource the following data (if available) is recorded:

1. **Timestamp** – When the resource was recovered.

2. **URL** – The resource's canonical URL on the Web.

3. **MIME type** – The MIME type returned from the web repository when downloading the resource.

4. **Filename** – The full path to the saved resource.

```
SeedUrls ← StartingUrl
RepoSet ← {IA,Google,Live,Yahoo}
// Perform lister queries
if Using(KnowledgeablePolicy) or Using(ExhaustivePolicy) then
    foreach Repo r in RepoSet do
        StoredUrls ← getAllUris(site)
        if repoUriLimitExceeded(r) then $additionalUris_r$ ← true
    end
end
if Using(ExhaustivePolicy) then SeedUrls ← SeedUrls ∪ StoredUrls
Frontier ← SeedUrls
foreach URL u in Frontier do
    FoundResources ← ∅
    if isImageUrl(u) then
        // Find an image in one repository
        foreach Repo r in RepoSet do
            if $u$ ∈ StoredUrls(r) or $additionalUris_r$ then
                image ← getResource($r$,$u$)
                if image is found then
                    FoundResources ← image
                    break
                end
            end
        end
    else
        // Search all four repositories for non-image resource
        foreach Repo r in RepoSet do
            if $u$ ∈ StoredUrls(r) or $additionalUris_r$ then
                resource ← getResource($r$,$u$)
                date ← extractStoredDate(resource)
                FoundResources ← FoundResources ∪ (resource,date)
            end
        end
    end
    // Find the most recent or canonical resource
    resource ← chooseBestResource(FoundResources)
    SaveResource(resource)
    VisitedUrls ← VisitedUrls ∪ $u$
    if htmlResource(resource) then Frontier ← Frontier ∪ getUnvistedUrls(resource)
    if anyRepoQueryLimitExceeded(RepoSet) then Sleep24Hours
end
```

FIG. 26: Warrick's algorithm for reconstructing a website.

```
2006-11-17 10:23:04
http://foo.edu/~joe/
text/html
foo.edu/~joe/index.html
yahoo
2006-11-15
google:2006-08-11,live:2006-07-20

2006-11-17 10:23:08
http://foo.edu/~joe/images/hello.gif
MISSING

2006-11-17 10:23:13
http://foo.edu/~joe/resume.pdf
text/html
foo.edu/~joe/resume.pdf
google
2006-11-09
yahoo:2006-10-02

2006-11-17 10:23:29
http://foo.edu/~joe/styles.css
text/css
foo.edu/~joe/styles.css
ia
2005-02-14
```

FIG. 27: Reconstruction summary file.

5. **Web repository** – The repository from which the canonical or most recent resource was saved.
6. **Stored date** – The date the resource was crawled on the Web (the datestamp from the repository).
7. **Other repositories** – Other repositories that had a stored version of the resource.

An example is shown in Figure 27 (although each entry is separated by a tab in the file, it is displayed on separate lines for clarity). Note that the MIME type recorded in the summary file may not always match up with the original resource's MIME type. For example, if a PDF is recovered from a search engine cache as shown in the example, the MIME type is 'text/html' since it was recovered as HTML from the repository.

4 RUNNING

Warrick may be downloaded and ran from the command line on any operating system with a valid Perl 5 installation. Warrick may also be accessed through the web interface of the queueing system which is discussed in the next section. The queueing system runs Warrick from the command line using the proper switches.

Warrick uses many of the same switches as the popular GNU Wget web crawler [65]. A complete list of switches is given in Appendix A, but some of the commonly used switches are discussed in this section. The following is a typical invocation:

```
warrick.pl --recursive --debug --output-file log.txt http://foo.edu/~joe/
```

The `--recursive` switch tells Warrick to recursively look for more links to recover from recovered HTML pages. If the switch is not supplied, only the single URL will be recovered. The `--debug` switch turns on debugging output, and the `--output-file` switch indicates all output should be diverted to the output file `log.txt`. This can be useful for monitoring long reconstruction jobs. The ending URL is the root page to the website being reconstructed. Warrick will by default only recover resources that are under the root page and match the given URL. So only URLs matching `http://foo.edu/~joe/*` will be recovered in this example. This matches the organizational structure of most websites.

The default crawling policy used by Warrick is the Knowledgeable policy. The Naïve policy can be set using `--no-lister-queries` and `--complete` for Exhaustive. Warrick can be ran to supplement a previous reconstruction using `--no-clobber`; this is useful since IA may add additional resources to their archive each month which were not previously accessible.

Reconstructions may also be limited to certain repositories. For example, if the website to recover has been lost for years and is only in the Internet Archive, the `--web-repo` switch could be used to limit the reconstruction to only IA. Additionally the `--date-range` could be used to limit the date range of recovered resources. The following command indicates only IA should be used, and only resources archived between February 1, 2004 and August 8, 2005 should be recovered:

```
warrick.pl --recursive --web-repo ia
   --date-range 2004-02-01:2005-08-31 http://www.cs.odu.edu/
```

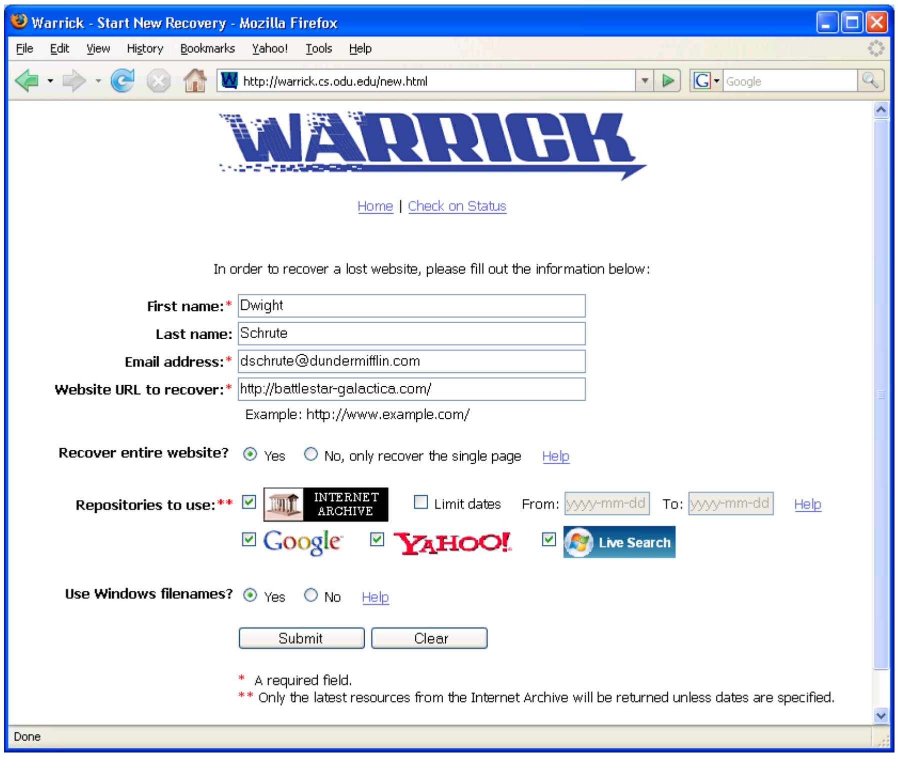
After a reconstruction has completed and all resources have been saved to disk, the website may not be immediately browsable for two reasons: 1) the recovered resources contain links to where the resources *used* to reside on the Web (and currently do not), and 2) some resources may have file extensions that do not match the content of the file (e.g., a recovered PDF recovered from a search engine cache is in HTML format, but a browser will attempt to read it as a PDF document if it has a .pdf extension).

Warrick may be ran with two switches to rectify these problems. The `--convert-links` switch will convert all absolute URLs to relative URLs. For example, the absolute URL pointing to `http://foo.edu/~joe/car.html` will be converted to point to the recovered car.html file, "`../car.html`". The `--view-local` switch will make a reconstructed website completely browsable off-line. This option does four things: 1) converts all absolute links to relative links (just like the `--convert-links` switch), 2) appends ".html" to all filenames of HTML resources that do not already have a .htm or .html extension, 3) converts all '?' characters in filenames into dashes, and 4) changes all links in all HTML resources to point to the newly renamed files.

5 BRASS

5.1 User Interface

Nontechnical users or those who prefer not to download, install and learn Warrick's many command-line switches may use Brass, a queueing system developed to run Warrick on a network of machines [110]. When a user wants to recover a lost website with Brass, they are presented with the screen shown in Figure 28. The user's email address is requested so the user can verify the job through


Warrick - Start New Recovery - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://warrick.cs.odu.edu/new.html
WARRICK
Home | Check on Status

In order to recover a lost website, please fill out the information below:

First name:*	Dwight
Last name:	Schrute
Email address:*	dschrute@dundermifflin.com
Website URL to recover:*	http://battlestar-galactica.com/
	Example: http://www.example.com/
Recover entire website?	[✓ Yes] No, only recover the single page Help
Repositories to use:**	[✓ INTERNET ARCHIVE] Limit dates From: yyyy-mm-dd To: yyyy-mm-dd Help
	[✓ Google] [✓ YAHOO!] [✓ Live Search]
Use Windows filenames?	[✓ Yes] No Help

Submit Clear

* A required field.
** Only the latest resources from the Internet Archive will be returned unless dates are specified.

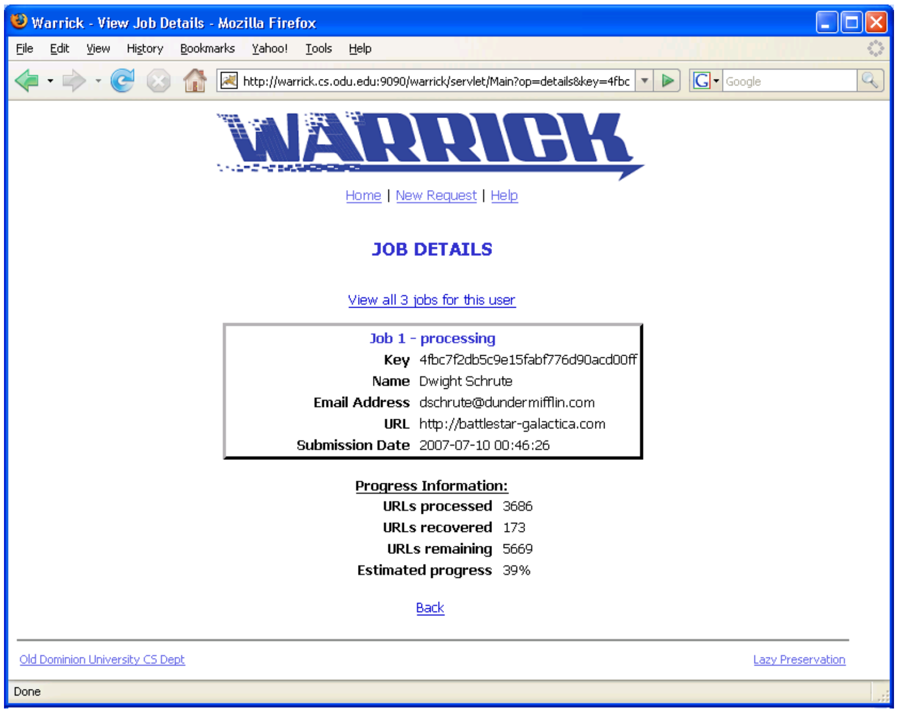
Done

FIG. 28: Screenshot of Brass's submit job interface.

email (so automated attempts to start jobs are thwarted) and so the user can be notified when the website has been fully recovered. The user must also provide the base URL of the lost website and indicate if just the single resource residing at the URL should be recovered or the entire website. Next the user selects which web repositories should be used in the recovery. A date range can be given if IA is selected by itself, otherwise the most recent version of resources from IA is selected by Warrick.

The user may also indicate if the resources should be stored using Windows file-naming conventions. Windows does not allow certain characters like '?', '—' and ':' in filenames, so when they are encountered in a URL, they are converted to acceptable characters like '@', '%7C' and '+' when used to name the file, respectively.

Once the user submits the job, Brass assigns a unique key to the job (using an MD5 hash of the email address and the timestamp when the job was submitted) and sends a job verification email to the user. Once the user replies to the verification email, the job is placed in a queue to be executed once a free machine is available. The job may take several days to process if there are many jobs in front of it or if recovering a very large website. The user may check on the status of submitted jobs at any time (Figure 29). Once the job completes, an email is sent to the user with a link, allowing the user to download the completed reconstruction as a gzipped tar file or zip file. Periodic reminders are sent to the user for several weeks if the job is not picked up.


Warrick - View Job Details - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://warrick.cs.odu.edu:9090/warrick/servlet/Main?op=details&key=4fbc
Google
WARRICK
Home | New Request | Help
Back
Old Dominion University CS Dept
Lazy Preservation
Done

JOB DETAILS

View all 3 jobs for this user

Job 1 - processing

Key	4fbc7f2db5c9e15fabf776d90acd00ff
Name	Dwight Schrute
Email Address	dschrute@dundermifflin.com
URL	http://battlestar-galactica.com
Submission Date	2007-07-10 00:46:26

Progress Information:

URLs processed	3686
URLs recovered	173
URLs remaining	5669
Estimated progress	39%

FIG. 29: User checking status of current job.

5.2 Administrative Interface

The Brass administrative interface (Figure 30) allows the admin to view jobs in their various states. A progress bar is used to show the status of processing jobs using the following equation:

$$\text{Progress} = \frac{\text{processed URLs}}{\text{processed URLs} + \text{queued URLs}} \tag{19}$$

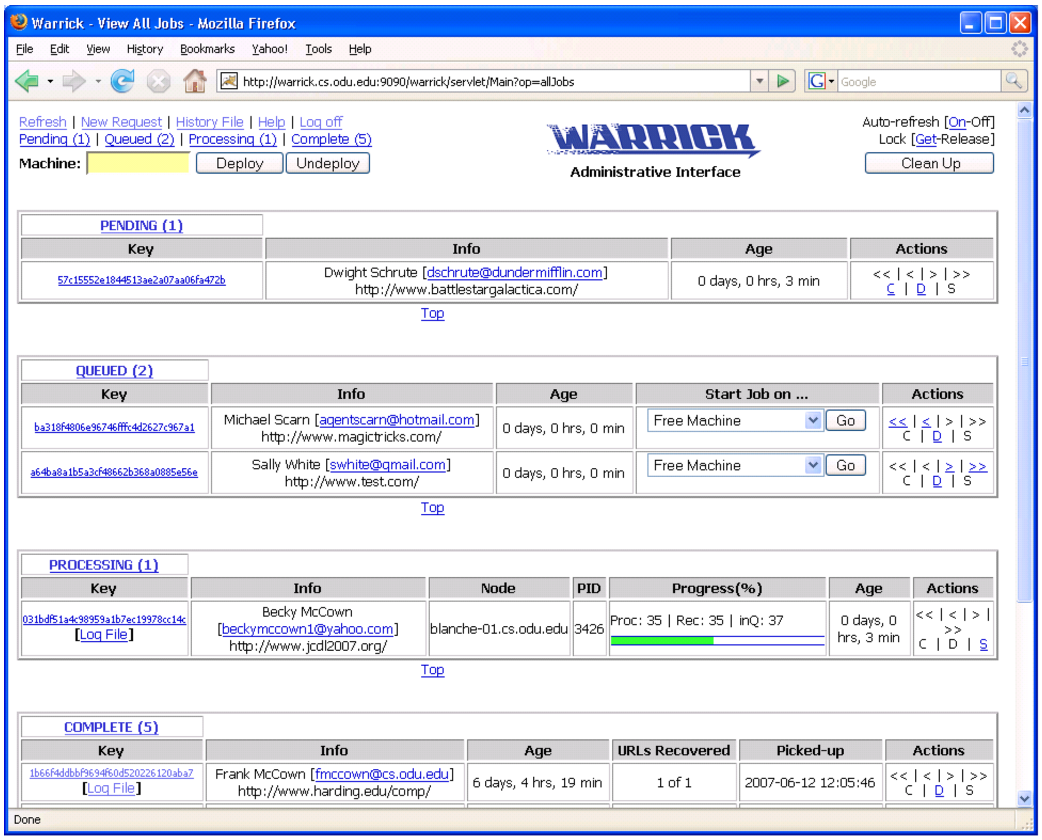
The admin screen is updated when the page is refreshed manually or by turning on the auto-refresh. The admin can move jobs in the queue, manually start jobs on specific machines, delete jobs and add new worker machines. A history is kept of all jobs and is accessible through the browser or by downloading an XML file.

5.3 Architecture

As illustrated in the administrative interface, a job moves through four states in Brass:

1. *Pending*: The job has been submitted by the user (Figure 28), but has not yet been confirmed via email. After ten days, pending jobs are removed from the system.

2. *Queued*: The job has been confirmed by the user and is scheduled to run. Although the administrator can reposition jobs in the queue, normally they are scheduled to run in the order they were confirmed.


Warrick - View All Jobs - Mozilla Firefox
File Edit View History Bookmarks Yahoo! Tools Help
http://warrick.cs.odu.edu:9090/warrick/servlet/Main?op=allJobs
Google
Refresh | New Request | History File | Help | Log off
Pending (1) | Queued (2) | Processing (1) | Complete (5)
Machine:
Deploy
Undeploy
WARRICK
Administrative Interface
Auto-refresh [On-Off]
Lock [Get-Release]
Clean Up
Done

PENDING (1)

Key	Info	Age	Actions
57c15552e1844513ae2a07aa06fa472b	Dwight Schrute [dschrute@dundermifflin.com] http://www.battlestargalactica.com/	0 days, 0 hrs, 3 min	<< | < | > | >> C | D | S

Top

QUEUED (2)

Key	Info	Age	Start Job on ...	Actions
ba318f4806e96746fffc4d2627c967a1	Michael Scarn [agentscarn@hotmail.com] http://www.magictricks.com/	0 days, 0 hrs, 0 min	Free Machine Go	<< | < | > | >> C | D | S
a64ba8a1b5a3cf48662b368a0885e56e	Sally White [swhite@gmail.com] http://www.test.com/	0 days, 0 hrs, 0 min	Free Machine Go	<< | < | > | >> C | D | S

Top

PROCESSING (1)

Key	Info	Node	PID	Progress(%)	Age	Actions
031bdf51a4c98959a1b7ec19978cc14c [Log File]	Becky McCown [beckymccown1@yahoo.com] http://www.jcdl2007.org/	blanche-01.cs.odu.edu	3426	Proc: 35 | Rec: 35 | inQ: 37	0 days, 0 hrs, 3 min	<< | < | > | >> C | D | S

Top

COMPLETE (5)

Key	Info	Age	URLs Recovered	Picked-up	Actions
1b66f4ddbbf9694f60d520226120aba7 [Log File]	Frank McCown [fmccown@cs.odu.edu] http://www.harding.edu/comp/	6 days, 4 hrs, 19 min	1 of 1	2007-06-12 12:05:46	<< | < | > | >> C | D | S

FIG. 30: Screenshot of Brass's administrative interface.

```
<job>
  <key>uniquekey</key>
  <fname>John</fname>
  <lname>Smith</lname>
  <email>john@hotmail.com</email>
  <commandLine>--recursive --debug --complete
     --output-file recondir/uniquekey/log.txt
     --target-directory recondir/uniquekey/
     --google-api http://my-lost-website.com/
  </commandLine>
  <submitDate>2007-04-12 09:41:53</submitDate>
  <status>processing</status>
  <reminderToConfirm>1</reminderToConfirm>
  <lastConfReminder>2007-04-12 09:45:26</lastConfReminder>
  <assignDate>2007-04-13</assignDate>
  <processId>124</processId>
  <urlsProcessed>462</urlsProcessed>
  <urlsRecovered>390</urlsRecovered>
  <reminderToPickup>0</reminderToPickup>
  <lastPickupReminder />
  <pickupDate />
  <pickedUp>no</pickedUp>
</job>
```

FIG. 31: Job information stored in XML.

3. *Processing*: The job is currently running. Due to the query limitations discussed in Section 2, large jobs can remain in this state for several days.

4. *Completed*: The job is finished running and is ready for pick-up by the user (in the form of a zipped file or gzipped tar file). An email reminder is sent periodically, and if the job is not picked up in several weeks, it is deleted from the system.

Jobs are stored in an XML file on the web server; an example is shown in Figure 31. The Apache Tomcat web server [6] runs on the primary web server and each worker machine. When a job is to be started on a free machine, Brass performs an HTTP GET request with the proper parameters to the Tomcat instance on the target machine. The GET request starts the reconstruction controller (RC), a script which launches Warrick, zips up the recovered files when Warrick completes and finally makes an HTTP GET request back to the primary web server. This in turn triggers an email to be sent to the user and changes the job's status to complete.

Each instance of Warrick uses the Exhaustive policy when recovering a complete website and outputs its current status to a log file. For simplicity, Brass and all the worker machines share a common filesystem, so checking on the current status of each Warrick process can be done from the primary web server by examining the log files which all reside in the same networked directory. The log files are accessible to the admin via the web browser for potential trouble-shooting.

TABLE 12: Brass usage statistics from 2007.

Month	Completed Jobs	Resources Recovered
July	118	119,596
August	75	83,435
September	132	107,014
Average	108	103,348

6 USAGE STATISTICS

Before Brass was deployed in early July, Warrick users had to download and run Warrick on their own systems. From February 1, 2006 to August 1, 2007, Warrick was downloaded approximately 2500 times (not including automated bots and crawlers like Google). Warrick does not communicate its usage information back to ODU, so that data is not available. However, Brass usage statistics have been kept since its deployment in early July and is summarized in Table 12. In the past three months, Warrick has been used to recover a total of 325 websites composed of 310,045 resources. For 8.3% of the jobs submitted, only a single resource was requested for recovery instead of a complete website. Of those completed jobs that recovered at least one resource, 10.7% were not picked-up by the user. In these cases, the users may have been able to recover the lost websites another way or lost interest in recovering the websites.

Table 13 gives descriptive statistics on the number of URLs that were discovered by crawling the recovered HTML pages and the number of those URLs that were recovered. The table also summarizes the number of days that it took for Warrick to complete the submitted jobs. On average, 38.2% of the websites' resources were recovered. The percent jumps to 47.2% if only considering websites with at least one recovered resource. The distribution of recovered resources per website is shown in Figure 32. For 19.7% of the jobs, none of the resources could be recovered. Less than 10% of the resources were recovered for a quarter of the jobs (26.8%), but a small minority (12.6%) of jobs have an 80% recovery rate or better.

In the next chapter, the percentage of recovered resources in all three experiments is significantly higher than the 38% seen here. As will be demonstrated in the next chapter (Section 5.3), web resources that are not well-connected in the web graph are less likely to be reconstructed from the WI. Perhaps the websites being reconstructed by Brass were not well-connected. In the experiments in the next chapter, URLs that were blocked by the robots exclusion protocol were not considered, but the real usage data presented here does not account for these blocked URLs; this likely accounts for a significant number of missing resources. Additionally, some website URLs may not have been typed correctly and therefore did not point to once valid websites. Some URLs may have been spam. It is also impossible to tell if the resources that were not recovered were as important as those that were found; possibly the resources that were found were the most significant and important resources.

The distribution of reconstructed websites by top level domain (TLD) is shown in Figure 33 along with the percent of recovered resources for each TLD. Websites with .com TLDs make up

TABLE 13: Brass recovery summary.

	Ave	Min	Max	Std
URLs discovered	3218.8	0	196,449	14226.5
URLs recovered	954.0	0	19,872	3085.2
Days to complete	0.86	0	26	3.26

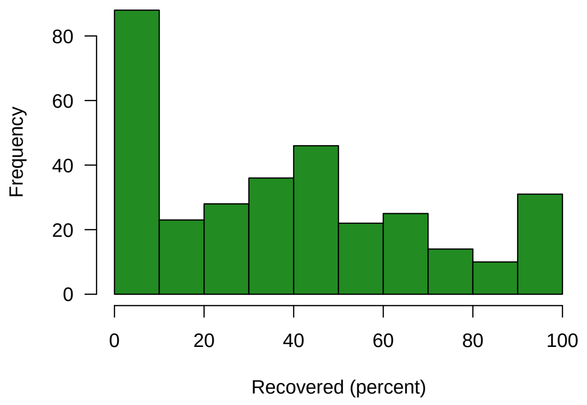


FIG. 32: Distribution of websites by percentage of recovered resources.

approximately half of all reconstructions with country code TLDs coming in second, making up a quarter of all jobs. Recovery success differs some by TLD. Websites with a .net TLD experienced nearly a 50% recovery rate compared to only 30% for the five .edu sites that were reconstructed. As will be shown in the next chapter (Section 5.2), websites from the .edu domain are much more reconstructable than websites from other domains, but there are numerous other factors that account for recovery success, not just TLD.

All four repositories were used in a majority of the reconstructions (Table 14), but IA contributed significantly more resources per reconstruction than the other three repositories. In 20% of the reconstructions, IA is the *only* contributor. This is not surprising considering IA is the only deep archive and therefore the only source of web resources that have been lost for a considerable amount of time. And since the Exhaustive policy is used by Brass, many old URLs that may no longer be a part of the lost website are recovered from IA. An experiment from the next chapter (Section 5.2) finds IA's contribution to be significantly lower when reconstructing websites that have not yet been lost.

7 CONCLUSIONS

Warrick can be used to reconstruct a website from the WI and can perform post-processing on the recovered files to make them browsable off-line. The development of Brass has allowed Warrick to be

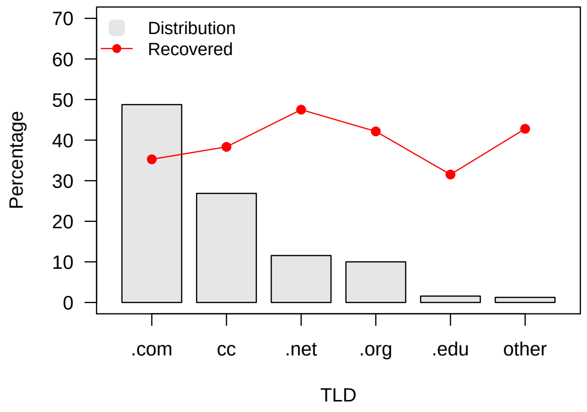


FIG. 33: Brass recovered websites by TLD.

TABLE 14: Repository use, contributions and requests.

	Used in recons	Contribution	Requests per recon (ave)
IA	99.7%	84.8%	2150.4
Google	96.9%	6.8%	657.7
Live	96.9%	3.3%	131.3
Yahoo	96.9%	5.1%	819.4

far more accessible to the general public; usage statistics show that the public is currently recovering 108 websites a month on average and recovering an average of 38% of their resources. Although this percentage is not very high, it is not known if the missing portions are as significant as the portions that were found or if websites that were not recoverable were simply misspelled or spam. In order to gauge just how effective Warrick is at reconstructing websites, a more rigorous evaluation is necessary than just examining usage statistics. The next chapter introduces an evaluation method for quantifying website reconstruction success and investigates how effective Warrick is at reconstructing a variety of websites.

CHAPTER VII

EVALUATING LAZY PRESERVATION

The web-repository crawler Warrick, introduced in the previous chapter, can be used to reconstruct websites from the WI. But how effectively can a "typical" website be reconstructed? This is a difficult question to answer for several reasons. First, it is not always possible to know how much of a website was really recovered when the original is truly lost; as mentioned in the previous chapter, the first individual to use Warrick was unsure how much of his blog was really recovered. Second, reconstructing synthetic websites like those developed in Chapter IV and later in Chapter VIII might be detected as spam and treated very differently than "real" websites. And third, reconstructing websites which have not yet been lost may produce a more favorable outcome since the search engine caches would not have yet purged any missing web resources. Despite the potential for overly optimistic results, the strategy used to evaluate website reconstructions in this chapter uses real websites which have not yet been lost.

This chapter attempts to gauge the effectiveness website reconstruction using three reconstructions experiments. The first experiment used Warrick to reconstruct 24 hand-picked websites that varied in size, structure and subject matter. A follow-up experiment reconstructed the same 24 websites using the three crawling policies introduced in Chapter V, and the policies were evaluated in terms of the recovered resources. Finally, a large-scale experiment was performed where 300 randomly selected websites were reconstructed over a three month period. From this final experiment, the factors contributing most to successful reconstructions were found.

1 WEBSITE DEFINITIONS

1.1 Website Definition

According to the W3C's Web Characterization Terminology and Definition Sheet [94], a **website** is defined as:

> A collection of interlinked Web pages, including a host page, residing at the same network location. "Interlinked" is understood to mean that any of a Web site's constituent Web pages can be accessed by following a sequence of references beginning at the site's host page; spanning zero, one or more Web pages located at the same site; and ending at the Web page in question.

A web page is defined as the "collection of information, consisting of one or more Web resources, intended to be rendered simultaneously, and identified by a single URI." By these definitions, the Faculty web page `http://www.cs.odu.edu/faculty.shtml` is part of the Computer Science (CS) Department's website since a link from the site's host page at `http://www.cs.odu.edu/` points to it and since the page is accessible from the same network location (a web server hosted by the department). The style sheets, JavaScripts, images and other resources required to render the Faculty page are also part of the web page and therefore the website.

However, Michael Nelson's website at `http://www.cs.odu.edu/~mln/` may be judged by some to be a different website than the CS department's since it is published and maintained by a different entity than the CS website. Nelson's website is called a **subsite** by the W3C's definition, but for simplicity, it will be called a website in this dissertation. The curatorial aspects of a website make it possible to distinguish between websites that share the same domain name but are housed on different hosts. For example, the Warrick website housed at `http://warrick.cs.odu.edu/` can be considered a separate website from the CS website.

The W3C's definition of website does not take into account those pages that are dynamically generated and are not linked directly to other pages of the site. These deep web pages which are often invisible to a web crawler may be discovered by the WI using other methods as discussed earlier in Chapter II. The definition of website in this dissertation includes these deep web resources which can be discovered from web repositories using lister queries.

1.2 Lost Websites

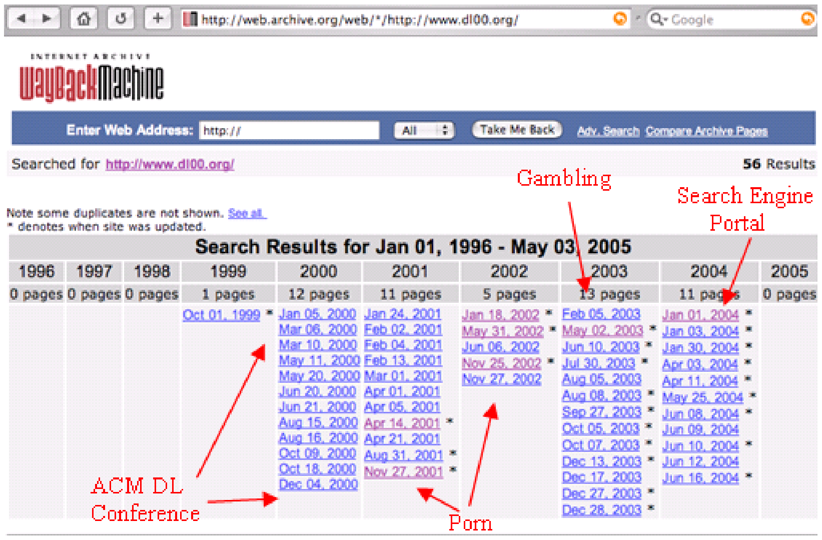
A **lost website** is one that is no longer accessible from its original location and is not being mirrored at any other location[1]. Such a website could be in one of several categories:

- **Unresponsive server** – The server is not responding to any HTTP requests. This could happen if the web server was accidentally shut down, hacked or was the victim of a denial-of-service attack.
- **Missing content** – The content of the website is no longer accessible at the same address, yet the website appears to be under the same ownership. The content could have been removed because it was no longer needed or relevant, or the website has been reorganized. An example of a website whose content is no longer accessible is TechLocker.com, shown in Figure 35. Although the site is still accessible from its original URL, the contents have been lost; the site is now redirecting users to `www.campusemporium.com`.
- **Expired domain name** – The domain name lease for the website has expired and is waiting to be re-purchased. Like the previous category, the content is no longer accessible. Figure 36 shows how the defunct website of former Congressman Mark Foley (`www.markfoley.com`) is now being "parked... courtesy of GoDaddy.com," a domain name registrar.
- **Hijacked** – The domain name for the website has expired and been claimed by another entity. An example of a hijacked domain is given in Figure 34. The original DL website was lost in 2001 to a pornographer. The website was later used for gambling in 2003 and a search portal in 2004.

1.3 Reconstructed Websites

A **reconstructed website** is 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 files, recovered resources would be equivalent to the files that were lost. For sites produced dynamically using CGI, PHP, etc., the recovered resources would match the client's view

[1] A notable log of lost websites is maintained by Steve Baldwin at `http://www.disobey.com/ghostsites/`.



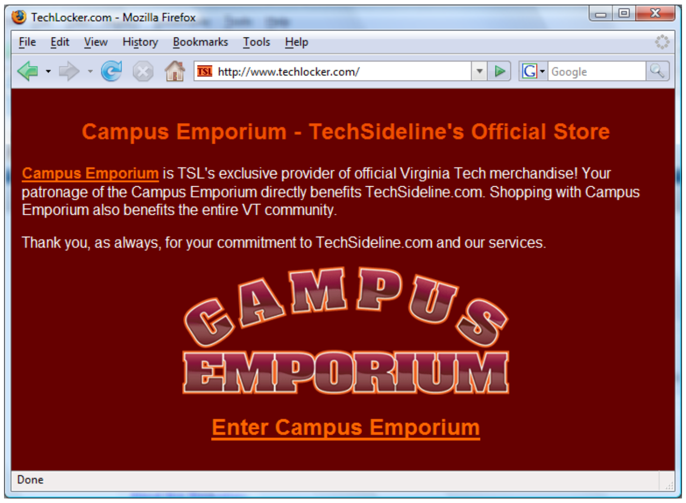
Searched for http://www.dl00.org/ 56 Results

Note some duplicates are not shown. See all.
* denotes when site was updated.

Search Results for Jan 01, 1996 - May 03, 2005

1996	1997	1998	1999	2000	2001	2002	2003	2004	2005
0 pages	0 pages	0 pages	1 pages	12 pages	11 pages	5 pages	13 pages	11 pages	0 pages
			Oct 01, 1999 *	Jan 05, 2000	Jan 24, 2001	Jan 18, 2002 *	Feb 05, 2003	Jan 01, 2004 *	
				Mar 06, 2000	Feb 02, 2001	May 31, 2002 *	May 02, 2003 *	Jan 03, 2004 *	
				Mar 10, 2000	Feb 04, 2001	Jun 06, 2002	Jun 10, 2003 *	Jan 30, 2004 *	
				May 11, 2000	Feb 13, 2001	Nov 25, 2002 *	Jul 30, 2003 *	Apr 03, 2004 *	
				May 20, 2000	Mar 01, 2001	Nov 27, 2002	Aug 05, 2003	Apr 11, 2004 *	
				Jun 20, 2000	Apr 01, 2001		Aug 08, 2003 *	May 25, 2004 *	
				Jun 21, 2000	Apr 05, 2001		Sep 27, 2003 *	Jun 08, 2004 *	
				Aug 15, 2000	Apr 14, 2001 *		Oct 05, 2003 *	Jun 09, 2004	
				Aug 16, 2000	Apr 21, 2001		Oct 07, 2003 *	Jun 10, 2004 *	
				Oct 09, 2000	Aug 31, 2001 *		Dec 13, 2003 *	Jun 12, 2004	
				Oct 18, 2000	Nov 27, 2001 *		Dec 17, 2003	Jun 16, 2004 *	
				Dec 04, 2000			Dec 27, 2003 *		
							Dec 28, 2003 *		

FIG. 34: IA versions of http://www.dl00.org/.


TechLocker.com - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.techlocker.com/
Google
CAMPUS EMPORIUM
Done

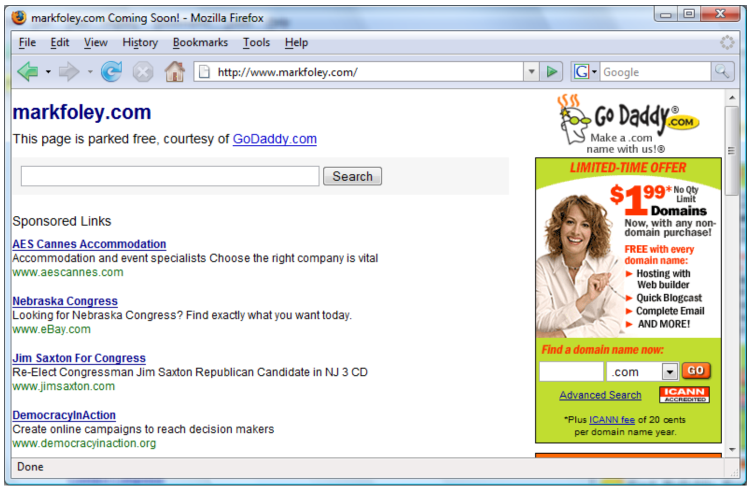
Campus Emporium - TechSideline's Official Store

Campus Emporium is TSL's exclusive provider of official Virginia Tech merchandise! Your patronage of the Campus Emporium directly benefits TechSideline.com. Shopping with Campus Emporium also benefits the entire VT community.

Thank you, as always, for your commitment to TechSideline.com and our services.

Enter Campus Emporium

FIG. 35: The website www.techlocker.com remains accessible, but the previous content is lost.


markfoley.com Coming Soon! - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.markfoley.com/
Google
Search
Done
Go Daddy .COM
Make a .com name with us!®
LIMITED-TIME OFFER
$1.99* No Qty Limit Domains
FREE with every domain name:
Hosting with Web builder
Quick Blogcast
Complete Email
AND MORE!
Find a domain name now:
.com
GO
Advanced Search
ICANN ACCREDITED

markfoley.com

This page is parked free, courtesy of GoDaddy.com

Sponsored Links

AES Cannes Accommodation
Accommodation and event specialists Choose the right company is vital
www.aescannes.com

Nebraska Congress
Looking for Nebraska Congress? Find exactly what you want today.
www.eBay.com

Jim Saxton For Congress
Re-Elect Congressman Jim Saxton Republican Candidate in NJ 3 CD
www.jimsaxton.com

DemocracyInAction
Create online campaigns to reach decision makers
www.democracyinaction.org

Now, with any non-domain purchase!

*Plus ICANN fee of 20 cents per domain name year.

FIG. 36: The domain name markfoley.com is awaiting re-purchase.

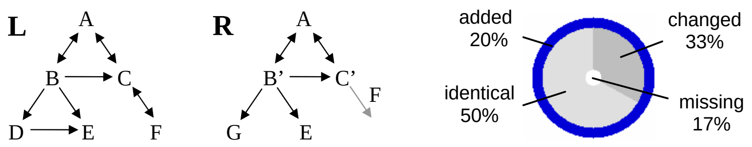


FIG. 37: Lost website (left), reconstructed website (center), and reconstruction diagram (right).

of the resources and would be useful to the webmaster in rebuilding the server-side components. Recovery of the server-side components with lazy preservation is discussed in Chapter VIII.

A reconstructed website may be made accessible once again by re-hosting it at the same URL it was previously available at or a new location. From anecdotal evidence collected by the author, websites that are reconstructed by the owner of the lost site are often made accessible once again at the same URL. Websites that are reconstructed by a third party are sometimes hosted at different locations since access to the original domain name is not always possible. The reconstructed website may also be used only for internal purposes. For example, a researcher who reconstructs an old website may want to only view its contents locally.

2 RECONSTRUCTION MEASUREMENTS

2.1 Quantifying a Reconstruction

To quantify the difference between a lost website (L) and a reconstructed website (R), all resources from L and R are matched according to their uniquely identifying URIs and placed into one of four categories. Each resource l_i in L and r_i in R that shares the same URI is categorized as *identical*

(r_i is byte-for-byte identical to l_i) or *changed* (r_i is not identical to l_i). All resources in L that do not share a URI with any resource in R are categorized *missing*, and those resources in R that do not share a URI with any resource in L are *added*. The union of L and R would be equivalent to the intersection of L and R if R was perfectly reconstructed from the WI, that is, it had no missing or added resources. Note that although the links connecting these resources are often important for finding them, they are not specifically evaluated in this classification scheme.

The four classifications are used to assign a three dimensional **recovery vector** (**r**) in the form of (changed, missing, added) to each resource: (0,0,0) for *identical* resources, (1,0,0) for *changed* resources, (0,1,0) for *missing* and (0,0,1) for *added*. A **difference vector** is then calculated as a measure of change between the lost website L and the reconstructed website R by summing the recovery vectors and normalizing them like so:

$$\text{difference}(L, R) = \left(\frac{r_c}{|L|}, \frac{r_m}{|L|}, \frac{r_a}{|R|} \right) \tag{20}$$

The difference vector is intuitively the percentage of resources that were changed, missing and added. A website that was reconstructed with all identical resources (a perfect reconstruction) would have a difference vector of (0,0,0). A completely unrecoverable website would have a difference vector of (0,1,0).

To illustrate, the left side of Figure 37 shows a web graph for some lost website L where each node represents a resource (HTML, PDF, image, etc.), and each directed edge is a hyperlink or reference from one resource to another. Suppose L was lost and reconstructed forming the website W represented in the center of Figure 37. The resources A, G and E were reconstructed and are identical to their lost versions, but an older version of B was found (B') that pointed to G, a resource that does not currently exist in R. Since B' does not reference D, D may not be recovered unless unless a lister query (defined in Section 3.5 of Chapter III) revealed its existence. An older version of C was found, and although it still references F, F could not be found in any web repository. For Figure 37, the difference vector is (2/6, 1/6, 1/5) = (0.333, 0.167, 0.200).

The difference vector for a reconstructed website can be illustrated as a **reconstruction diagram** as shown on the right side of Figure 37. 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 is useful for compactly summarizing a reconstruction.

2.2 Computing Success

When quantifying how successful a reconstruction is, several factors must be considered. A simple definition of success would be the percent of resources that were recovered ($1 - d_m$). But if some of the recovered resources were changed in such a way to make them less useful to the recoverer (e.g., a thumbnail was recovered instead of the full-sized image), some type of penalty to the changed resources (d_c) should be assessed. A penalty may also be assigned if the reconstruction resulted in many added resources (d_a) that hindered the recoverer's ability to separate the "important" parts of the website from the useless parts. By assigning penalties to the components of the difference vector, a reconstruction success level can be computed that matches one's intuitive notion of success.

To determine reconstruction success, a **penalty adjustment** may be applied to each individual recovery vector or to the final difference vector. The penalty adjustment is composed of weights (P_c, P_m, P_a) which are defined over the interval of [0,1] with 0 being no penalty and 1 being the maximum penalty. The weights can be adjusted depending upon the level of importance the recover wants to assign to resources in each recovery status category.

For example, suppose a website of mostly PDFs was lost, but 75% of the PDFs were recovered in an HTMLized format. A weight of 1 may be assigned to P_m to give the maximum penalty for not being able to recover the other 25% of the PDFs. A weight of 0.5 may be assigned to P_c since the text of the PDFs that were recovered were helpful, but the important PDF formatting of the text was lost. Another penalty adjustment of 1 for P_c could be used for those lost PDFs that contained only images since none of the images could be recovered from the HTMLized PDFs. Finally, a penalty of 0.2 could be assigned to P_a if the added resources caused the reconstruction to take a significantly longer amount of time or if the added resources were not useful to the recoverer or caused the recoverer additional time in locating the resources that were important.

Once the penalty adjustment weights have been selected, they can be applied individually to each recovery vector before computing the difference vector:

$$\mathbf{r} = (r_c \cdot P_c,\ r_m \cdot P_m,\ r_a \cdot P_a) \tag{21}$$

Alternatively, a single penalty adjustment could be applied to the final difference vector:

$$\text{difference}(L, R) = \left(\frac{r_c \cdot P_c}{|L|}, \frac{r_m \cdot P_m}{|L|}, \frac{r_a \cdot P_a}{|R|} \right) \tag{22}$$

The successful of the reconstruction is then the L1 norm (the sum of the vector components) of the difference vector (d_c, d_m, d_a) after applying the penalty adjustment:

$$\text{success} = d_c + d_m + d_a \tag{23}$$

The closer the value of success is to zero, the more successful the reconstruction. Note that $d_c + d_m$ is always ≤ 1, and $d_c + d_m + d_a$ is always ≤ 2.

Five general levels of success have been defined in Table 15 in increasing order of laxity by applying various penalty adjustments to the final difference vector and relaxing how some recovered resources are categorized.

In s3, an algorithm that computes similarity is needed. Shingling was used in this chapter since it takes into account the order of words between two textual resources. For clarity, **textual resources** are those resources with a MIME type of 'text/*' or those with MIME types associated with PDF, PostScript or Microsoft Office documents. Resources are classified as 'similar' if the crawled and recovered resources share at least 75% of their shingles of size 10 (after removing HTML markup).

The s5 category is useful when the Knowledgeable policy is used (crawling policies were introduced in Chapter V). Otherwise it is equivalent to the s4 category. To calculate s5, Warrick must be

TABLE 15: General levels of reconstruction success.

Success Level	Penalties	Description
s1	(1,1,1)	Missing, changed and added are equally undesirable.
s2	(1,1,0)	Missing and changed are equally undesirable, but added resources are not.
s3	(1,1,0)	The definition of changed is relaxed by removing textual resources that are 'similar' from changed.
s4	(0,1,0)	Missing resources are undesirable, but changed and added are not.
s5	(0,1,0)	The definition of missing is relaxed by removing potentially recoverable resources.

configured to track those resources that are known to be stored in at least one repository but were not recovered due to the selection algorithm of the Knowledgeable policy. Therefore if a resource could potentially be recovered, it could be removed from the 'missing' classification.

The s5 level is the most generous definition of success since it does not penalize for changed resources, and it eliminates the bias of using the Knowledgeable policy in the reconstructions. The value 1 - s5 is the percentage of recovered resources for a website, regardless if the resource was changed in some way.

2.3 Comparing Live Websites

In most cases, the lost website is not available to compare with the reconstructed website. Therefore, when evaluating reconstruction success, it is more practical to select a website that is live on the Web and reconstruct it as if it were suddenly lost. A snapshot of the website can be obtained by crawling it, and then a snapshot of the reconstructed website can be taken from the WI by crawling it with a web-repository crawler. Figure 38 illustrates taking a snapshot of a website by web crawling (left). The same website is reconstructed from the WI (right), and the crawled and recovered resources are then compared and categorized (bottom) using their URIs as discussed in Section 2.1.

The disadvantage of using live websites for reconstruction experiments is that it does not take into account the passage of time and the loss of resources from search engine caches. If a website was reconstructed a week after it was lost, for example, many of its resources might no longer be accessible in the search engine caches; as was demonstrated in Chapter IV, search engines are often quick to purge cached resources once they are detected to no longer be accessible on the Web.

3 INITIAL RECONSTRUCTION EXPERIMENT

3.1 Methodology

To gauge the effectiveness of lazy preservation for website reconstruction, an initial experiment was conducted comparing the snap-shot of 24 live websites with their reconstructions. Websites were selected that were either personally known to the author or randomly sampled from the Open Directory Project (dmoz.org). Sites that used robots.txt and Adobe Flash [3] exclusively as the

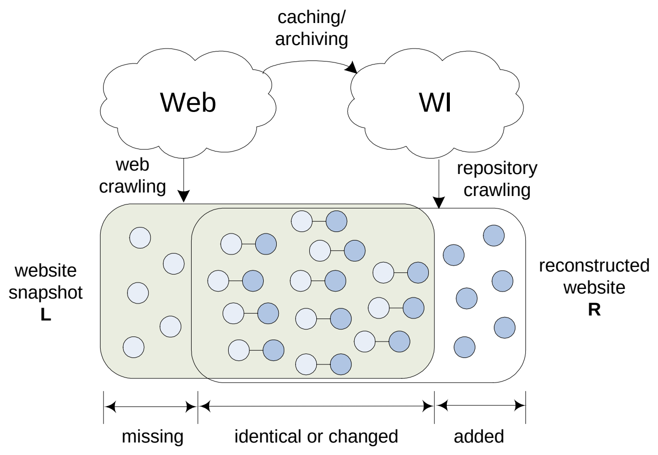


FIG. 38: Comparing a crawled website (left) to its reconstruction (right).

main interface were avoided. The selected websites were predominantly English, covered a range of topics, were from a number of top-level domains, and varied in size (8 small (<150 URIs), 8 medium (150-499 URIs) and 8 large ($\geq$500 URIs) were selected).

In order to see if there was any correlation between a website's relative importance and its recovery success, Google's PageRank (an integer between 0-10 where 10 is the highest importance) was manually recorded for the base URL of each website by using the Google Toolbar [67]. Although Google is the only search engine that publicly reports its 'importance' measure for a URL, it is possible other search engines assign similar importance values to the same websites. Google representatives in the past have reported the PageRank value from the toolbar is several months old [62], but it is the only importance metric easily available.

In August 2005 all 24 websites were crawled starting at their base URL using the GNU Wget crawler [65]. All links and references were followed that were in and beneath the base URL, with no limit to the path depth. This captured all pages, images, style sheets, JavaScripts, etc. that can be found by a typical web crawler. All exclusions found in robots.txt were honored. For simplicity, the crawler was restricted to port 80 and 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/bar/`, only URLs matching `http://www.foo.edu/bar/*` were downloaded. The same settings were used by Warrick when reconstructing the websites.

Immediately after crawling the websites, five different versions of each website were reconstructed with Warrick: four using each web repository separately and one using all four web repositories together. The different reconstructions helped to show how effective individual web repositories could reconstruct a website versus the aggregate of all four web repositories. The Naïve policy was used for all reconstructions since the Knowledgeable and Exhaustive policies were not yet implemented.

The reconstructions ran on six servers running Solaris, each with their own IP address. This allowed the limited repository queries to be divided among six machines, thus increasing the speed

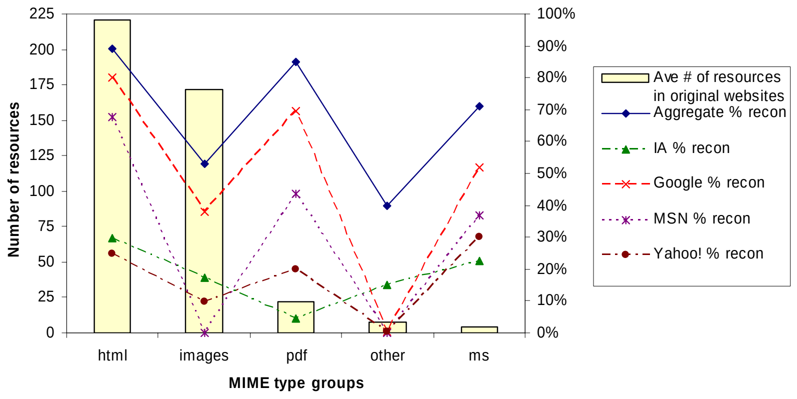


FIG. 39: Recovery success by MIME type.

at which reconstructions could be performed.

3.2 Results

The results of the 24 aggregate reconstructions (using all four web repositories) are shown in Table 16, ordered by percent of recovered URIs. The 'PR' column is Google's PageRank for the root page of each website at the time of the experiment. For each website, the total number of resources in the website is shown along with the total number of resources that were recovered and the percentage. This is equivalent to 1 - s4. Resources are also totalled by three MIME type groups: HTML, images and other. The difference vector reported for the websites uses the s1 level of success.

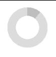
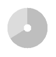
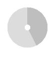
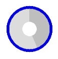
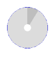
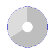
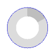
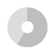
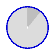
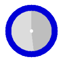
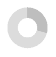
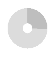
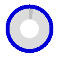
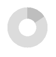
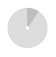
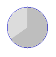
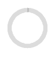
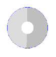
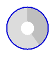
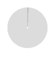
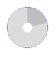
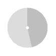
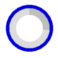
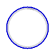
The 'Almost identical' column of Table 16 shows the percentage of textual resources (e.g., HTML, PDF, PostScript, Word, PowerPoint, Excel) that were *almost identical* to the originals. The last column shows the reconstruction diagram for each website using 1 - s3 (almost identical resources are moved from 'Changed' to 'Identical').

Summarizing the results, 68% of the resources were recovered on average (median=72%). For a quarter of the websites, more than 90% of the original resources were recovered. Of those resources recovered, 30% of them on average were not byte-for-byte identical. A majority (72%) of the 'changed' text-based files were almost identical to the originals (having at least 75% of their shingles in common). Sixty-seven percent of the 24 websites had obtained additional resources when reconstructed which accounted for 7% of the total number of resources reconstructed per website.

When all website resources are aggregated together and examined, dynamic pages (those that contained a '?' in the URL) were significantly less likely to be recovered than resources that did not have a query string (11% vs. 73%). URLs with a path depth greater than three were also less likely to be recovered (52% vs. 61%). A chi-square analysis confirms the significance of these findings ($p < .001$). There was no correlation between percentage of recovered resources with PageRank or website size.

The success of recovering resources based on their MIME type is plotted in Figure 39. The percentage of resources that were recovered from the five different website reconstructions (one

TABLE 16: Results of initial website reconstructions.

		MIME type groupings (orig/recovered)						
Website	PR	Total	HTML	Images	Other	Difference vector	Recon diag	Almost identical
1. smoky.ccsd.k12.co.us	4	63/27 43%	20/20 100%	23/5 22%	20/2 10%	(0.111, 0.571, 0.000)		100%
2. genesis427.com	2	65/53 82%	10/10 100%	50/40 80%	5/3 60%	(0.662, 0.185, 0.000)		33%
3. englewood.k12.co.us/schools/clayton	3	68/58 85%	32/29 91%	36/29 81%	0/0	(0.426, 0.147, 0.000)		N/A
4. harding.edu/hr	4	73/47 64%	19/19 100%	25/2 8%	29/26 90%	(0.438, 0.356, 0.145)		83%
5. raitinvestmenttrust.com	4	79/65 82%	24/24 100%	45/33 73%	10/8 80%	(0.089, 0.177, 0.015)		33%
6. mie2005.net	6	89/66 74%	16/15 94%	28/7 25%	45/44 98%	(0.663, 0.258, 0.015)		89%
7. otago.settlers.museum	5	111/48 43%	27/27 100%	82/19 23%	2/2 100%	(0.171, 0.568, 0.020)		40%
8. usamriid.army.mil	7	142/100 70%	38/38 100%	59/19 32%	45/43 96%	(0.585, 0.296, 0.000)		50%
9. searcy.dina.org	5	162/154 95%	96/95 99%	63/56 89%	3/3 100%	(0.111, 0.049, 0.078)		43%
10. cookinclub.com	6	204/187 92%	67/66 99%	136/121 89%	1/0 0%	(0.480, 0.083, 0.307)		100%
11. americancaribbean.com	4	287/152 53%	60/57 95%	222/90 41%	5/5 100%	(0.296, 0.470, 0.000)		100%
12. gltron.org	6	294/221 75%	20/19 95%	213/190 89%	61/12 20%	(0.259, 0.248, 0.005)		90%
13. privacy.getnetwise.org	8	305/163 53%	137/137 100%	48/25 52%	120/1 1%	(0.033, 0.466, 0.201)		70%
14. mypyramid.gov	0	344/193 56%	158/154 97%	141/5 4%	45/34 76%	(0.160, 0.439, 0.000)		32%
15. digitalpreservation.gov	8	414/378 91%	346/329 95%	42/25 60%	26/24 92%	(0.097, 0.087, 0.000)		44%
16. aboutfamouspeople.com	6	432/430 99%	267/267 100%	165/163 99%	0/0	(0.653, 0.005, 0.021)		100%
17. home.alltel.net/bsprowl	0	505/112 22%	173/112 65%	332/0 0%	0/0	(0.012, 0.778, 0.009)		100%
18. dpconline.org	7	552/384 70%	236/227 96%	195/37 19%	121/120 99%	(0.509, 0.304, 0.013)		66%
19. cs.odu.edu/~pothen	5	640/435 68%	160/151 94%	258/120 47%	222/164 74%	(0.402, 0.320, 0.062)		28%
20. eskimo.com/~scs	6	719/691 96%	696/669 96%	22/21 95%	1/1 100%	(0.011, 0.039, 0.001)		50%
21. financeprofessor.com	6	817/626 77%	455/404 89%	312/180 58%	50/42 84%	(0.211, 0.234, 0.011)		72%
22. fishingcairns.com.au	5	1152/1070 93%	259/259 100%	890/808 91%	3/3 100%	(0.466, 0.071, 0.000)		95%
23. techlocker.com	4	1216/406 33%	687/149 22%	529/257 49%	0/0	(0.267, 0.666, 0.175)		99%
24. kruderdorfmeister.com	4	1509/128 8%	1298/31 2%	211/97 46%	0/0	(0.056, 0.915, 0.066)		50%

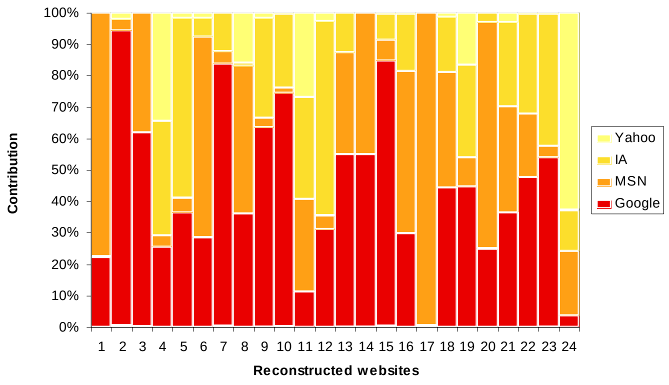


FIG. 40: Web repositories contributing to each website reconstruction.

using all four web repositories, and four using each web repository individually) are shown along with the average number of resources making up the 24 crawled (or original) websites. A majority (92%) of the resources making up the original websites are HTML and images. HTML resources were far more recoverable than images; 100% of the HTML resources were recovered for 9 of the websites (38%) using all four web repositories. It is likely fewer images were recovered because MSN at the time could not be used to recover images, and as the caching experiment revealed (Chapter IV), images are also much less likely to be cached than other resource types.

Figure 39 emphasizes the importance of using all four web repositories when reconstructing a website. By just using IA or just using Google alone, many resources were not recovered. This is further illustrated by Figure 40 which shows the percentage of each web repository's contribution in the aggregate reconstructions (sites are ordered by their numbering in Table 16). Although Google was the largest overall contributor to the website reconstructions (providing 44% of the resources) they provided none of the resources for site 17 and provided less than 30% of the resources for 9 of the reconstructions. MSN contributed on average 30% of the resources; IA was third with 19%, and Yahoo was last with a 7% contribution rate. Yahoo's poor contribution rate is likely due to their spotty cache access as exhibited in the caching experiment (Chapter IV) and because Yahoo's last-modified datestamps are frequently older than last-cached datestamps (Warrick chooses resources with the most recent datestamps).

The amount of time and the number of queries required to reconstruct all 24 websites (using all four repositories) is shown in Figure 41. There is nearly a 1:1 ratio of queries to seconds. Although the size of the original websites gets larger along the x-axis, the number of resources reconstructed and the number of resources held in each web repository determine how many queries were performed. In none of the reconstructions were the daily repository query limits exceeded (see Table 11 in Chapter VI for a listing of limits).

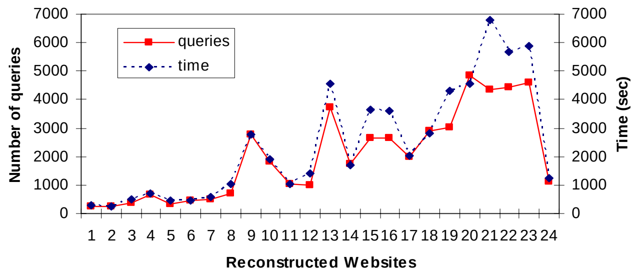


FIG. 41: Number of queries performed and time taken to reconstruct websites.

3.3 Discussion

Several lessons were drawn from this initial experiment. First, all four web repositories must be used to provide the most comprehensive reconstructions; using Google alone or IA alone will not provide adequate recovery for every website. Second, HTML resources appear to be most recoverable; images are far less recoverable from the WI. Third, website reconstruction from the WI can be very effective for many websites, but for others (e.g., kruderdorfmeister.com), it is nearly useless. Although a correlation was found with dynamic resources and path depth with recoverability, an experiment sampling far more sites is warranted.

4 CRAWLING POLICIES EXPERIMENT

4.1 Methodology

To better understand how the Naïve, Knowledgeable and Exhaustive crawling policies affect website reconstructions, another experiment was devised similar to the first. The same 24 websites were crawled using the same settings (honoring robots.txt, downloading only pages with URIs matching the base URL, etc.). After downloading each website, three concurrent reconstructions were immediately started using each of the crawling policies. The downloads and reconstructions were preformed in late February 2006 using the same six servers servers as the previous experiment.

4.2 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 Tables 17 and 18 ordered by total URIs (number of resources downloaded). 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 it was reconstructed and therefore represented a truly "lost" website. The root page no longer had links to other portions of the website, so downloading the website resulted in only 1 file.

TABLE 17: Results of crawling-policy website reconstructions (Part 1).

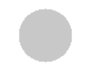
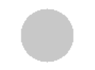
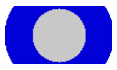
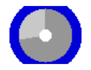
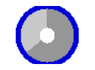
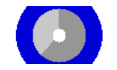
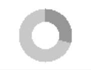
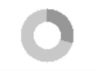
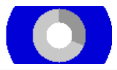
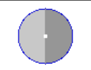
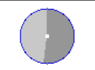
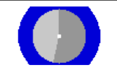
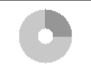
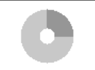
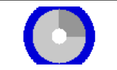
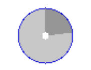
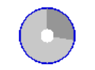
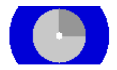
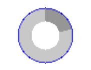
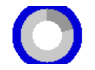
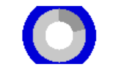
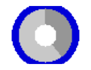
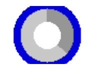
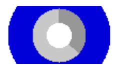
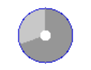
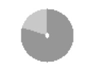
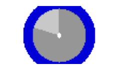
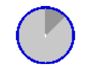
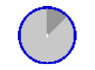
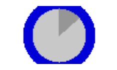
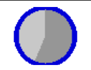
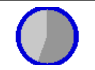
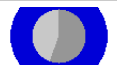
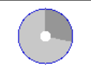
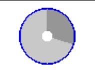
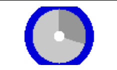
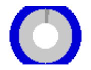
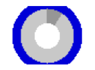
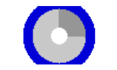
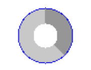
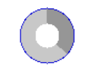
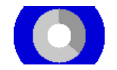
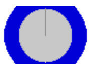
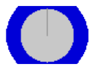
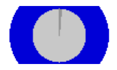
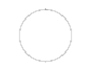
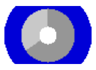
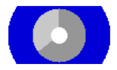
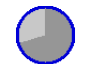
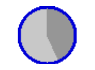
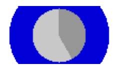
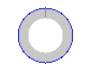
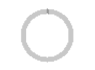
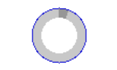
Website	URIs	Naïve		Knowledgeable		Exhaustive	
1. www.techlocker.com	1	(0.000, 0.000, 0.000)		(0.000, 0.000, 0.000)		(0.000, 0.000, 1.000)	
2. www.harding.edu/hr	50	(0.720, 0.220, 0.328)		(0.640, 0.280, 0.163)		(0.620, 0.240, 0.591)	
3. www.smoky.ccsd.k12.co.us	57	(0.298, 0.509, 0.000)		(0.298, 0.509, 0.000)		(0.316, 0.491, 0.970)	
4. www.genesis427.com	65	(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	(0.247, 0.286, 0.000)		(0.247, 0.286, 0.000)		(0.247, 0.286, 0.304)	
6. www.raitinvestmenttrust.com	79	(0.228, 0.127, 0.014)		(0.278, 0.253, 0.033)		(0.253, 0.127, 0.859)	
7. otago.settlers.museum	120	(0.208, 0.525, 0.017)		(0.208, 0.542, 0.286)		(0.208, 0.533, 0.341)	
8. www.usamriid.army.mil	121	(0.397, 0.413, 0.220)		(0.364, 0.512, 0.253)		(0.364, 0.471, 0.880)	
9. www.mie2005.net	136	(0.699, 0.199, 0.009)		(0.801, 0.096, 0.000)		(0.801, 0.096, 0.335)	
10. searcy.dina.org	164	(0.128, 0.049, 0.071)		(0.128, 0.049, 0.077)		(0.134, 0.037, 0.325)	
11. www.cookinclub.com	216	(0.565, 0.037, 0.140)		(0.556, 0.037, 0.148)		(0.560, 0.032, 0.790)	
12. www.gltron.org	306	(0.284, 0.180, 0.004)		(0.301, 0.183, 0.035)		(0.301, 0.183, 0.260)	
13. privacy.getnetwise.org	326	(0.021, 0.479, 0.306)		(0.092, 0.475, 0.296)		(0.261, 0.307, 0.321)	
14. www.americancaribbean.com	329	(0.380, 0.450, 0.005)		(0.368, 0.450, 0.005)		(0.368, 0.450, 0.689)	
15. www.eskimo.com/˜scs	357	(0.008, 0.006, 0.508)		(0.008, 0.008, 0.509)		(0.014, 0.006, 0.834)	
16. www.digitalpreservation.gov	389	(0.015, 0.946, 0.000)		(0.653, 0.321, 0.612)		(0.643, 0.308, 0.898)	
17. www.aboutfamouspeople.com	396	(0.705, 0.005, 0.088)		(0.434, 0.005, 0.086)		(0.419, 0.005, 0.844)	
18. home.alltel.net/bsprowl	474	(0.004, 0.654, 0.006)		(0.013, 0.808, 0.000)		(0.055, 0.665, 0.006)	

TABLE 18: Results of crawling-policy website reconstructions (Part 2).

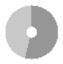
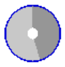
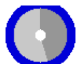
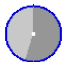
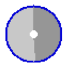
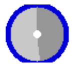
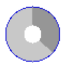
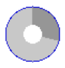
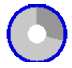
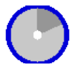
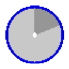
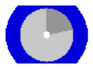
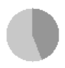
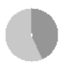
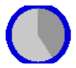
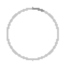
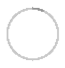
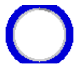
Website	URIs	Naïve		Knowledgeable		Exhaustive	
19. www.dpconline.org	580	(0.543, 0.209, 0.000)		(0.450, 0.217, 0.032)		(0.452, 0.214, 0.282)	
20. www.cs.odu.edu/˜pothen	610	(0.549, 0.067, 0.044)		(0.485, 0.146, 0.048)		(0.480, 0.152, 0.178)	
21. www.mypyramid.gov	646	(0.367, 0.327, 0.011)		(0.291, 0.345, 0.002)		(0.291, 0.344, 0.102)	
22. www.financeprofessor.com	673	(0.184, 0.165, 0.147)		(0.189, 0.080, 0.069)		(0.215, 0.120, 0.511)	
23. www.fishingcairns.com.au	1181	(0.439, 0.025, 0.000)		(0.434, 0.040, 0.000)		(0.411, 0.036, 0.197)	
24. www.kruderdorfmeister.com	2503	(0.068, 0.916, 0.000)		(0.068, 0.916, 0.000)		(0.069, 0.914, 0.243)	

TABLE 19: Statistics for crawling-policy website reconstructions.

Category	Policy	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

Table 19 contains the descriptive statistics of several important factors of the website reconstructions. The percentage of recovered resources are for those resources that share 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 expected, the Exhaustive policy recovered significantly more added resources because every resource stored in every web repository was recovered regardless if a link was found to the resource or not.

Table 19 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 the crawling policies is to examine each website reconstruction's **efficiency ratio**:

$$\text{efficiency ratio} = \frac{\text{total number of recovered resources}}{\text{total number of issued repository requests}} \tag{24}$$

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 42, and the distribution is plotted at the bottom. It is also useful to examine the efficiency ratios when added resources are not considered. Figure 43 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 19.

The efficiency ratios were grouped into pairs and a Wilcoxon Matched-Pairs Signed Ranks test was executed 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 expected, the Naïve policy was least efficient at recovering resources because the crawler did not know in advance which resources a web repository had stored. The Exhaustive policy was shown to be slightly more efficient than the Knowledgeable policy, likely because lister queries produce 'false positives' for the Knowledgeable policy that are not recovered.

When the efficiency ratios were considered without added resources, there continued to exist strong, statistically significant differences ($p < 0.001$) between the crawling policies with the exception of the (Exhaustive, Naïve) pair which still maintained a low p value ($p < 0.05$). When added resources were excluded, the Knowledgeable policy performed 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.

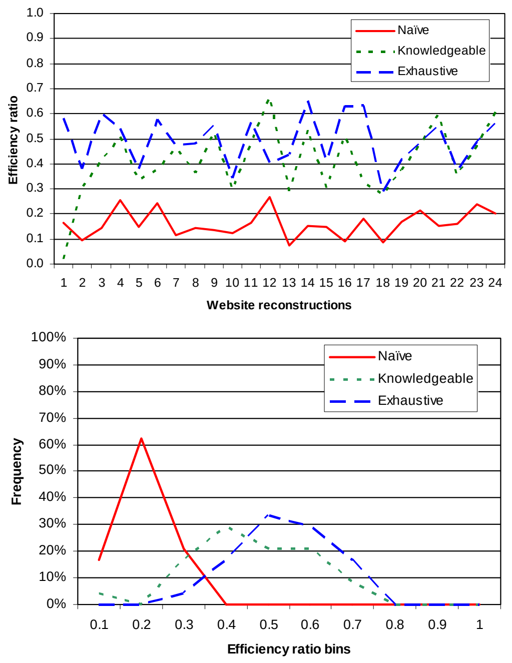


FIG. 42: Top: Efficiency ratio = total number of recovered resources divided by total number of repository requests. Bottom: Distribution of efficiency ratios where bin 0.1 corresponds to ratios with values between 0 and 0.1, etc.

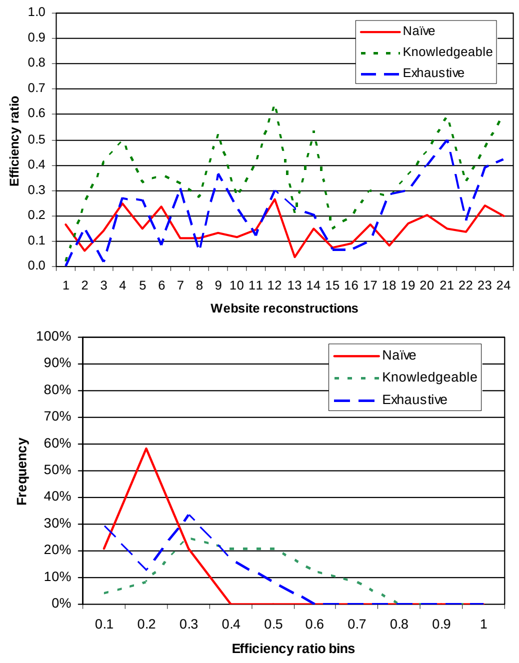


FIG. 43: Top: Efficiency ratio excluding 'added' resources. Bottom: Distribution of efficiency ratios excluding 'added' resources where bin 0.1 corresponds to ratios with values between 0 and 0.1, etc.

4.3 TechLocker Website

As noted earlier, the TechLocker website had apparently been closed down several months before this experiment was conducted. Using the Naïve and Knowledgeable policies, only a single HTML resource could be recovered. The Exhaustive policy recovered 2244 resources with 1952 of them 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 the cached date of the images from Google were not available, the HTML resources had been cached between January and September 2005, approximately five months to one year before the experiment had been conducted.

The original URLs of the recovered resources were accessed to see if they still resided on the TechLocker web server. Each request consistently resulted in an 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 the previous caching experiment from Chapter IV has demonstrated, Google may make resources available in their cache long after the resources have been removed from a website. This behavior has not been observed for MSN or Yahoo.

4.4 Discussion

The reconstruction experiments reveals 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. Therefore, the Naïve 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 reconstructing 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 as 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.

From the recovery of the lost TechLocker website, Google's cache has been shown to retain resources for months after they have disappeared from the Web. This suggests that re-running Warrick months after a website is lost may still uncover cached resources.

5 FACTORS AFFECTING WEBSITE RECONSTRUCTION

5.1 Methodology

Sampling Websites

A final experiment was conducted to determine which factors may contribute the most to the success of website reconstruction from the WI [112]. This required a random sample of websites in order not to bias the the findings. Since sampling uniformly from the Web is currently not possible [164], samples were obtained from the Open Directory Project (ODP) at `dmoz.org`. The ODP indexes a wide variety of websites in over 40 languages, and all search engines have an equal chance of indexing it.

URLs from the ODP were randomly selected that had a path depth of zero (`http://foo.org/`) or one (`http://foo.edu/~bar/`) in order to limit the selection to the root pages of websites. Each website was crawled starting from the selected seed URL, and all accessible resources were crawled, regardless of MIME type.

In the previous two experiments, Wget was used as the crawler. Wget lacks some needed options for a large-scale crawl, so Heritrix [120] was used instead. Heritrix is the Internet Archive's archival quality web crawler, and it is built for doing deep crawls of multiple websites at the same time. Heritrix was configured to respect the robots exclusion protocol and delay an appropriate amount of time per request in order to avoid over-burdening any particular site [165]. To avoid common crawler traps, the maximum path depth was limited to 15 and maximum hop from the root page to 15. To avoid re-crawling the same resource multiple times, URLs were normalized to lowercase and stripped of common session IDs. And for simplicity, the download was restricted to URLs using port 80 and the same host and domain name.

Sampling from the ODP data and crawling websites continued until 300 accessible websites had been found that matched a minimum set of qualifications (see Appendix B for a listing of all websites). First, any website that was entirely blocked by robots.txt or contained noindex/nofollow meta tags in the root page was rejected (only eight sites fit this description). Second, the websites had to contain valid content; websites with expired domains (two sites) or under reconstruction (one site) were rejected. And in order to ensure that the selected websites could be completely reconstructed within a one week time period, any websites that contained more than 10K resources when crawled were rejected (26 websites). Although Warrick is capable of reconstructing websites of any size, websites with more than 10K resources typically take more than a week to reconstruct due to the limited number of daily queries imposed by the web repositories. In terms of size, the sampled websites exhibited the power-law distribution that has been previously measured on the Web [2] where most sites had few resources and few sites had many resources.

Data Collection

For 14 weeks (late August to late November in 2006), the 300 websites were crawled using the same crawling policy as described previously. Crawls were performed on weekends when traffic is typically low on most web servers. All 300 websites were reconstructed weekly by running two Warrick processes on each of five servers. Warrick used the Knowledgeable crawling policy since it was shown in the previous experiment (Section 4) to be the most efficient policy in terms of number of repository queries and recovered resources. The weekly crawls and reconstructions produced approximately 5 GB and 500 MB of compressed data, respectively.

Over the course of the experiment, several of the websites became inaccessible. Three websites reported their bandwidth had been exceeded for a couple of weeks, and a few others appeared to be off-line or misconfigured for a few weeks. Two websites were inaccessible when they did not renew

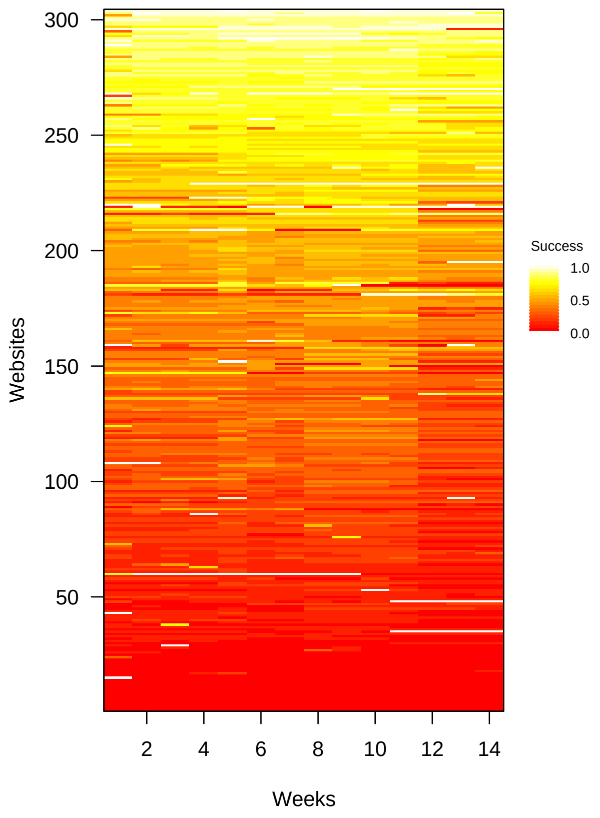


FIG. 44: Success of reconstructions by week.

their domain name, but both re-appeared intact as the same site a few weeks later. One website's domain name quit resolving on week 10 and never became accessible again. A couple of websites changed domain names. When this happened, the new domain name was added to the list of sites to crawl and reconstruct. Statistics have only been computed for successfully crawled websites.

5.2 Results

Recovery Success

Figure 44 shows an overall picture of how successful the reconstructions were over time. Each website's weekly success (using s5) is plotted with the most successful reconstructions at the bottom (graphs with other previously defined penalty adjustments looked similar). Each horizontal line marks the reconstruction success rate for the same website each week. The figure is not intended to give detailed information about any one website; instead, it shows that most websites were reconstructed to the same degree each week since the colors vary vertically but to a much lesser degree horizontally. But there are some exceptions. For example, site number 2 was successfully reconstructed every week (red all the way across), but site 148 experienced a huge increase in success on week 6 when it went from yellow to red (upon manual observation, site 148 changed the dynamic

TABLE 20: Descriptive statistics for reconstruction success levels.

	Mean	Median	Std	Min	Max	Websites with s* = 0
s1	0.7761	0.8164	0.3266	0	2	3%
s2	0.7137	0.7817	0.2606	0	1	3%
s3	0.6250	0.6796	0.2726	0	1	5%
s4	0.4567	0.4278	0.2867	0	1	6%
s5	0.3901	0.3477	0.2764	0	1	10%

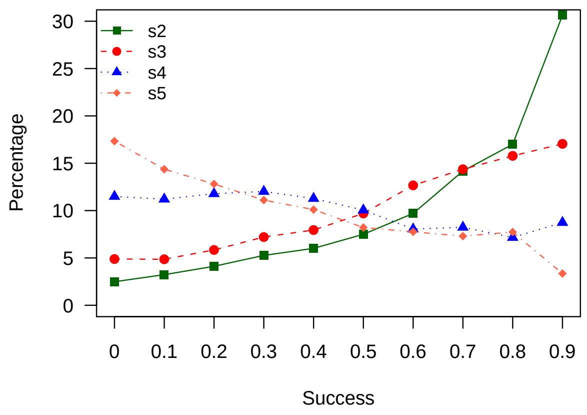


FIG. 45: Distribution of s2 to s5 success.

portion of their site which accounts for the increase in success).

The descriptive statistics for the reconstructions are given in Table 20 along with the percent of websites that experienced at least one reconstruction where the measured success was perfect (zero). As expected, the success values dropped closer to zero as the penalty adjustments were relaxed; where only 3% of the websites ever had a perfect reconstruction under the s1 level, 10% did under the s5 level.

When the distribution of the success levels are examined (Figure 45), the s2 and s3 levels are skewed to the right– about 16% of the reconstructions resulted in a 0.9 score or worse for s3, and 31% resulted in a 0.9 or worse for s2. The s4 and s5 levels (which do not penalize for changed resources) favor scores much closer to zero. Under the s5 measure, almost 17% of all reconstructions resulted in better than a 0.1 score. Note that s1 is not included in the figure since it is distributed over the interval [0 to 2]; its distribution was skewed to the right similar to s2 and s3.

Content Type

The two most common types of content found in the 300 websites were HTML and images, accounting on average for 40% and 53% of all content, respectively. Other textual resources like PDF, PostScript

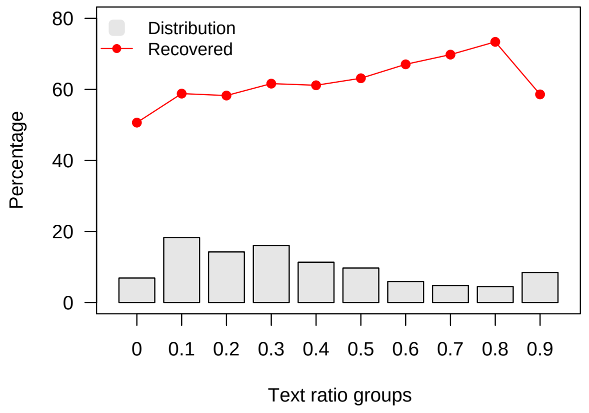


FIG. 46: Distribution and recovery of websites based on ratio of textual resources.

and Microsoft Office made up only a small fraction (2%) of all resources. All other resources combined made up 5% of the content on average. HTML and textual resources proved to be the most recoverable. On average, 77% of the HTML resources and 75% of the textual resources were recovered. Only 42% of the images and 32% of resources with some other MIME type were recovered.

To see how recovery affects the amount of textual resources in a website, the ratio of textual resources (HTML, PDF, MS Office, etc.) to other resource types were calculated for each website. Each site was placed in groups where the text ratio ranged from $[r,\ r + 0.1)$. Figure 46 shows the distribution of websites (bars) based on the ratio of textual resources making up the site. The average percentage of recovered resources for the sites is shown as a line (this is equivalent to 1 - s5, as discussed in Section 5.2). The figure shows that a majority of sites had text ratios between 0.1 and 0.6. Although the recovery line grows higher for each group, there is a significant drop from 73% for group 0.8 to 59% for group 0.9. The percentage of textual resources in a website is thus not the only factor dictating its recoverability from the WI.

Top-Level Domain

The 300 websites represented a variety of top-level domains (TLDs). As shown in Figure 47, almost half of the sites were from the .com domain, and almost 40% were from a country code (cc) domain (there were 25 distinct ccTLDs). Only four sites were from .edu, two from .tv and only one from .info. From Figure 47, most TLDs had a recovery rate around 60% with the exception of the four .edu sites which performed remarkably better.

Birth and Decay

The 300 websites exhibited little growth during the experiment. Half of the websites did not add any new resources during the 14 weeks. The weekly birth rate of new resources was calculated (as performed in [128]) by examining the fraction of new URLs that were crawled each week that were not seen in any of the previous crawls. The average birth rate was a relatively stable 0.049. Only

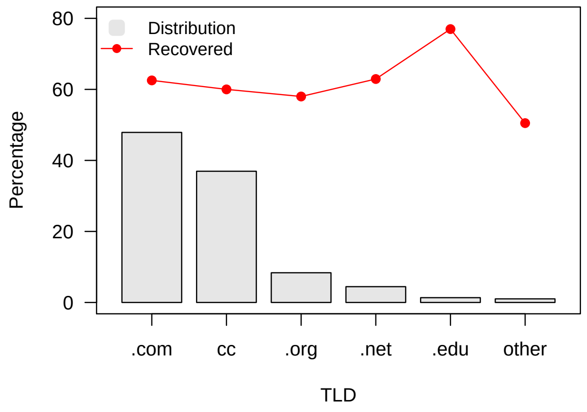


FIG. 47: Distribution and recovery by TLD.

on week 9 did the average birth rate increase substantially, and that was due to a single website that added 10K new URLs that week (and dropped almost all of them the following week). The URLs appeared to be dynamically generated, likely due to a configuration error on their web server. Discounting this website drops the average birth rate to a mere 0.014.

Although the websites did not grow much during the experiment, they did decay. A majority of the websites (61%) lost at least one resource during the 14 weeks. Figure 48 shows the fraction of new resources crawled each week (light bars) and the fraction of resources from week one that were also crawled on week n (dark bars). The bars are normalized so the number of resources in the first week is one. The figure illustrates that resources from week one slowly decayed (the dark bars gradually get smaller each week) and were usually replaced by new resources at different URLs (since the light bars hovered around 1.0). By week 14, the websites had lost about 13% of their resources on average.

Change Rate

The sampled websites exhibited a broad range of dynamism. Some websites remained unaltered for long periods of time, and others underwent numerous changes each day. To measure the rate of change from week to week, resources crawled from week n were compared with week $n - 1$. The change rate was calculated by taking the number of times a change was observed divided by the number of times the resource was downloaded minus one [50]. So a resource with a change rate of one means the resource changed every time it was crawled.

Most of the sampled resources (76%) did not change once during the 14 week period, and only 8% of the resources registered a change every time. Over a third of the websites (37%) did not have any resources that changed. The resource type that exhibited the most amount of change were HTML resources. Most images, PDFs, style sheets, etc. remained relatively static. Whereas 44% of the HTML resources changed at least once, and 15% of them changed every time, only 0.8% of the images changed more than once during the experiment.

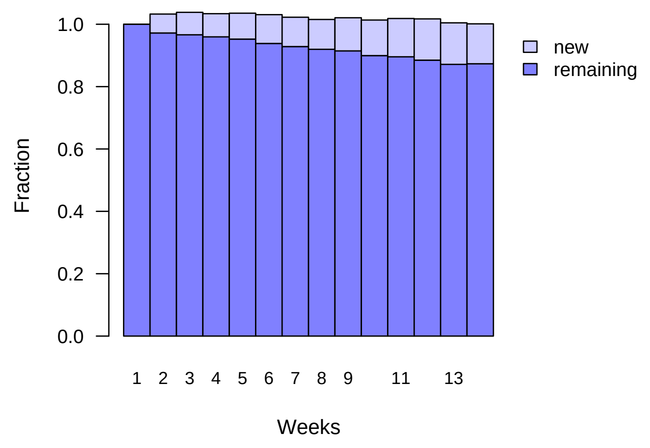


FIG. 48: Fraction of resources from the first crawl still existing after n weeks (dark bars) and new resources (light bars).

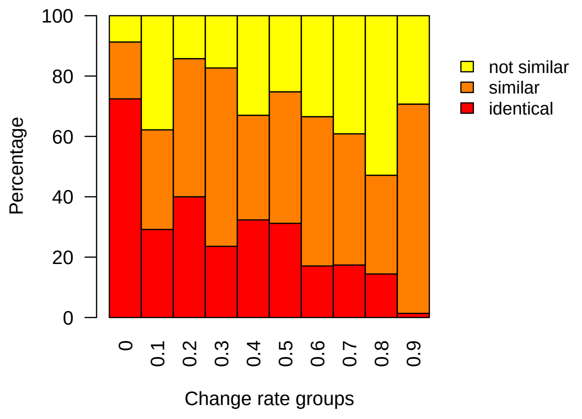


FIG. 49: Recovery category of HTML resources grouped by change rate.

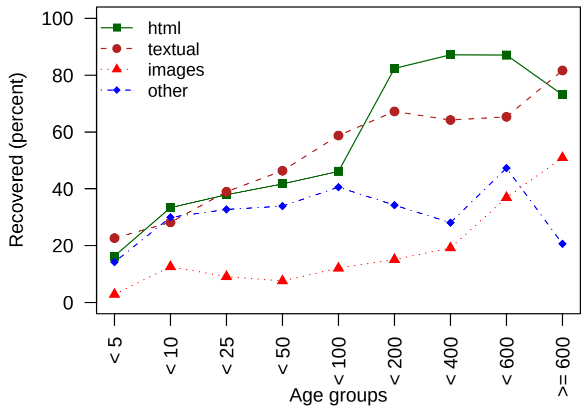


FIG. 50: Percentage of resources recovered by age.

To determine if the change rates would affect the recovery status (identical vs. changed) of recovered resources, the final week's reconstructions were examined (since the change rates are most accurate by the final week), and all recovered resources were grouped by their change rates varying from $[r$ to $r + 0.1)$. For each group, resources were categorized as identical, similar (shared 75% of their shingles) or not similar to their crawled counter-parts.

Figure 49 plots the change rates for HTML resources (since non-HTML resources exhibited little change). According to the figure, HTML resources that exhibited less than a 0.1 change rate had the highest percentage of identical recovered resources (72%). HTML resources with a change rate above 0.9 were rarely recovered in an identical state, but most (69%) were similar to their recovered counterparts.

The unusually sharp drop in identical resources for group 0.1 was accounted for by a single website; several hundred of its pages contained a MySQL error message embedded in them for two weeks in a row. If the dynamically generated pages had not been misconfigured when crawled, they would likely have been identical to the pages recovered from the WI.

The HTML resources with changed rates less than 0.1 that were 'not similar' to their recovered counterparts were manually examined, and most of the resources actually appeared to be similar to the recovered pages. Sometimes non-English pages were transformed when cached, and the comparison function did not account for all transformations.

Age

Determining the age of resources on the Web is difficult and imprecise. When a resource is downloaded, the only indication about its age can be derived from the Last-Modified timestamp. The Last-Modified date is when the file was last modified on the filesystem, not when it was created, so it is a lower bound on the resource's age. Additionally, web servers sometimes report incorrect timestamps [40], and they do not report timestamps for dynamic pages. The only resources for which the true age can be known (with an error of a few days) are those that appear for the first

TABLE 21: Reconstruction performance of web repositories.

Repository	Weekly contrib	Weekly requests per site	Efficiency ratio
IA	22.9%	127.2	0.38
Google	27.4%	152.6	0.38
MSN	32.4%	101.3	0.62
Yahoo	17.1%	232.2	0.24

time in a subsequent crawl. Even then it is possible for the resource to have been accessible at the same URL for a long time, but only before the crawl was a link to the resource added to the main website graph. Despite these limitations, a resource's age is defined as the number of days between the current access time and the first access time or Last-Modified timestamp, which ever is oldest.

Only 36% of the HTML resources in the crawls had a Last-Modified timestamp, but more than 99% of textual and image resources had them. On the final round of crawling, 59% of HTML resources were less than one year in age. If it is assumed that HTML resources missing a Last-Modified date that were crawled on the first week were also created that week, the percentage jumps to 85%. Images and textual resources were significantly older: 53% of images and 59% of textual resources were at least one year old.

To understand the relationship between age and recoverability, all crawled resources were grouped into 10 bins based on age. The bin breaks can be seen on the x-axis of Figure 50 (the first bin are resources less than 5 days old, the second less than 10 days old, etc.). Each resource type was graphed in Figure 50 based on the percentage of resources that were recovered in that age group. The figure shows a general increase in recovery success for all four types of resources as they age. As expected, the newest resources were generally the least recoverable. But the drop for HTML and 'other' resource types in the final age category indicate that age may be not be the single most significant predictor of recoverability.

Repository Contributions

Table 21 shows the percentage of resources that each repository contributed to the reconstructions. The table also lists the average number of weekly requests issued to each repository per website and the repository's efficiency ratio.

In the initial reconstruction experiment, Google was the largest provider of resources with MSN in second place. In this experiment, MSN was the largest contributor. One likely reason could be that in the initial study Warrick used Google's WUI to obtain cached results, but in this study Warrick used the Google API. As mentioned in Chapter III, the Google API may be serving from a smaller index than its web user interface.

The significantly higher requests per website and lower efficiency ratio for Yahoo is likely due to the fact that Yahoo must often be asked twice if it has a particular URL stored, one request with a 'www.' prefix and another without as discussed in Chapter V.

Crawler Directives

All four web repositories honor the robots exclusion protocol which protects certain URL paths from being crawled. There were 63 websites in the sample (21%) that had a valid robots.txt file, and 14 of the files did not block any URL paths. Two sites (one selling sporting goods and another on-line video games) specifically denied the IA crawler (ia_archiver) access to their entire website but only blocked a handful of URL paths for other crawlers. One website placed a robots.txt file on their site on week 4 that gave explicit permission for most search engines to crawl their entire site but blocked access to all other crawlers. The file was removed on subsequent weeks, possibly because the webmaster discovered that the rogue crawlers s/he wanted to block typically ignore robots.txt anyway. There were two sites that gave specific directives to googlebot, but none for msnbot or slurp (Yahoo).

By far the most popular URL paths being blocked were cgi-bin, images and administrative paths. This implies that many potentially valuable resources are not being preserved in the WI because of the high resource demands the WI places on some websites or the perceived danger of having administrative content replicated in the WI. Crawler burdens will likely continue to be a problem until more efficient Web discovery methods are adopted [126].

Some webmasters like their websites being indexed by search engines but would prefer they not be cached. Reasons may include the loss of potential website traffic and the lack of control to quickly remove embarrassing or false content from the Web [133]. All four web repositories will refrain from caching or archiving an HTML page if it contains a `noarchive` meta tag.

Examining all HTML pages, only two websites in the sample used `noarchive` meta tags, both from the `.de` ccTLD. The first site was protecting a personal blog from being cached, and the other was protecting all the PHP content from the commercial site. Interestingly, the second site only targeted Google; all other robots were allowed to cache the site. In Chapter IV, the use of `noarchive` meta tags only affected 2% of pages indexed by Ask, Google, MSN and Yahoo. The low usage of `noarchive` meta tags suggests that few webmasters of typical sites want their pages kept out of search engine caches and web archives. It may also be that few webmasters are even aware of the existence of, or reasons for, using `noarchive` meta tags. Whatever the reasons, the current low adoption of opt-out caching and archiving mechanisms is encouraging from a web preservation standpoint.

5.3 Reconstruction Model

Factors for Successful Reconstruction

There are many factors which may contribute to the success of website reconstruction from the WI. For example, a website composed mostly of textual resources would likely be more successfully reconstructed than a site of mainly binary zip files since all four repositories show a preference for textual resources over other types. It is also suspected that a website that is strongly connected to the web graph would be more recoverable than one with few inlinks since having greater inlinks increase the chance of a crawler finding the site. Older websites and sites that are more static in nature are also likely to be more recoverable.

In order to determine which factors contributed the most to reconstruction success of the 300 websites, several statistical tests were ran on the recovered resources, examining several variables:

- **External backlinks** – Websites with more inlinks (also called backlinks) to their root pages from other websites are more likely to be discovered by other crawlers and could possibly be crawled more frequently due to their importance. Lacking a large crawl of the entire Web, the backlink facility of Google, MSN and Yahoo was used to determine the known backlinks to the root page of each website every week. This measure is not always precise since Google does not reveal all known backlinks [45], and IA does not have a mechanism to reveal backlinks.
- **Internal backlinks** – Web crawlers can more easily find resources that contain a large number of backlinks within the site. Resources with few links may also be new additions to the website which a crawler has yet to find.
- **Google's PageRank** – Google likely re-visits a website frequently if it has a high PageRank, and therefore a website is more likely to have a larger footprint within Google than a site with a low PageRank. As mentioned previously, Google is the only search engine that publicly reports its 'importance' measure for a website, and the value published on the Google Toolbar may be out of date [62].
- **Hops from root page** – Crawlers often place hop count limits when crawling websites, so it is expected that websites with pages closer to the root page will be better reconstructed than sites with pages far from the root.
- **Path depth** – Like hops, crawlers may reject URLs with long path depths.
- **MIME type** – Search engines prefer textual resources over other types like images, zipped files, etc.
- **Query string parameters** – Crawlers may reject dynamic pages with many query string parameters.
- **Age -** Websites that have very old resources are more likely to be stored in the WI than websites with new resources. This is especially true since at the time of the experiment, only resources that were at least 6-12 months old were accessible from IA [80].
- **Resource birth rate** – Websites that are producing new content at new URIs are less likely to be reconstructed successfully than websites that are not increasing in URIs since it takes time for new resources to be discovered.
- **TLD** – It is possible that a bias exists for the web repositories for particular TLDs [166, 170].
- **Website size** – It is possible that very large websites (in terms of number of resources) may have fewer of their resources cached/archived than smaller sites.
- **Resource size** – Perhaps the size of a resource influences wether it will be stored in the WI. Larger resources may be more content-rich and therefore preferred over small resources. But perhaps large resources are not cached and archived as often because they take longer to download.

Factors like the use of Flash, JavaScript, etc. by the websites were not considered since the Heritrix crawler is likely of equal or lesser technical capability when compared to the crawlers used by the web repositories. Heritrix and WI crawlers are likely to discover the same resources on the same website.

Analysis

Several statistical tests were applied using SAS software (version 9.1) to the recovered and missing resources (143,001 observations) from the final week of reconstructions when the age variable was most accurate. The Pearson's correlation coefficient was first examined to see if there was a correlation between any of the above mentioned interval variables. Birth rate, website size and resource size were first transformed using log to approximate the normal distribution, an assumption of this test. The highest correlation (0.428, $p < 0.0001$ where p is the p-value of the test of zero correlation) was between hops and the website's size. The positive correlation matches the intuition that it takes more hops to reach resources from the root page in larger websites. There was also a mild correlation between hops and path depth (0.388, $p < 0.0001$) which is expected since URLs with greater path depth are often located further down the web graph from the root page.

There was a mild negative correlation between age and number of query parameters (-0.318, $p < 0.0001$). This may be because dynamically produced pages are easier to add to a website (for example, by adding more records to a database) and because determining the age of dynamic pages is problematic as discussed in Section 5.2.

Finally, there was also a mild positive correlation between external links and PageRank (0.339, $p < 0.0001$), website size (0.301, $p < 0.0001$), and hops (0.320, $p < 0.0001$). The correlation between external links and PageRank is expected since Google's PageRank is at least partially influenced by external backlinks. The correlation between external links and website size may be explained by reasoning that larger websites tend to attract more links, either because the effort to create a larger website may imply the website is more important or of higher quality or because websites with many pages are more easily found by search engines and therefore will garner more links over time [39].

Since none of the correlations were above 0.5, none of the variables were removed from the model. Next a generalized linear model analysis was applied to determine which of the variables were most important in explaining the model. The website's host name was added to the analysis since it is possible that, all things being equal from the twelve parameters, two websites may still experience different levels of recovery. The resulting analysis had an R-square value of 0.468041 (DF = 322, $p < 0.0001$), meaning that the model explains about half of the variations observed. According to the type III sum of squares analysis, all thirteen variables were significant at the $p < 0.0001$ level except website size which was significant at the $p < 0.05$ level.

A multiple regression analysis was then performed with the ten continuous variables (ignoring the categorical variables of host, TLD and MIME type since categorical variables are not appropriate to this analysis) to determine how the variables impact the model. The analysis had an R-squared of 0.1943 (DF = 10, $p < .0001$), and the parameter estimates are shown in Table 22. An analysis showing the three most significant variables (if none of the others were available) produced PageRank, hops and age (R-squared = 0.1496).

TABLE 22: Regression parameter estimates.

Variable	Param Est	$P_r > \|t\|$
Intercept	0.76071	$< .0001$
External backlinks	-3.96E-7	$< .0001$
Internal backlinks	0.00004	$< .0001$
Birthrate	-0.13361	$< .0001$
PageRank	0.08162	$< .0001$
Website size	-0.04074	$< .0001$
Hops	-0.04184	$< .0001$
Path depth	-0.06044	$< .0001$
Query params	-0.04342	$< .0001$
Resource size	0.00248	.0018
Age	0.00014	$< .0001$

The parameter estimates confirm the initial hypotheses on the effect of each variable in the overall success of website reconstruction. The only parameter which did not fit the initial hypothesis was resource size. According to the analysis, resources have a slightly better chance of being recovered as their size increases. This may be because very small resources are not indexed by some search engines or have a higher chance of being dropped during the de-duping processes. A caveat to resource size is that search engines often limit the amount of data they will cache from any particular resource. For example, Yahoo will not cache more than 215 KB from a textual resource as was seen in Chapter IV.

The results of the multiple regression analysis can help predict how much of a website can be recovered if it were to be lost today. The model has a rather low R-squared value which indicates there are other parameters affecting website reconstruction which were not measured. One reason the model does not have a higher R-squared value is because IA and the three search engines have very different crawling and caching priorities. Had the reconstructions been performed with only IA or only the search engines, the analysis would possibly have been different. And since measuring age, external backlinks and PageRank is problematic, perhaps more accurate values would have increased the R-squared value. Finally, there are numerous hidden factors which cannot be measured which may account for some of the unexplained portions of the model. For example, webmasters submitting URLs directly to search engines and website discovery methods like the Sitemap Protocol may make a website more recoverable from the WI.

5.4 Discussion

This final experiment tried to measure the dynamism and “reconstructability” for the “typical” website. From the three month snapshots, it was shown that most of the sampled websites were relatively stable; over a third of the websites never lost a single resource over the entire experiment, and half of the websites never added any new resources. Thirty-seven percent of the websites did not have a single resource that registered a change during the 14 weeks. More than half of all images and textual resources were more than a year old, but at least 59% of the HTML resources were less than a year old (or had been modified within the year).

From the analysis, the typical website indexed by ODP can expect to get back 61% of its resources if it were lost today (77% textual, 42% images and 32% other). This finding is significantly higher than the 38% success rate witnessed in real Warrick usage data from the previous chapter (Section 6). Just the fact that the websites were indexed by ODP seems to have played a significant role in how well the websites were preserved.

The findings suggest the three most significant things a website can do to improve its chances of being successfully reconstructed are to improve its PageRank, decrease the number of hops a crawler must take to find all the website's resources, and create stable URLs for all resources. Google provides a number of tips for webmasters to improve their website PageRank scores, including admonitions to increase external backlinks, get listed in directories like the ODP, and use few query string parameters [68]. However, a website that does increase its PageRank may also hinder the ability to successfully reconstruct their website if it did become lost. Although a high PageRank will likely improve the amount of pages Google will crawl on a website, it also will likely increase the rate at which the search engine re-crawls the website. An experiment in Chapter IV demonstrated that Google purged their cache on the same day a re-crawl revealed a resource was no longer accessible on the Web. Therefore, resources may be evicted from cache more quickly for websites with high PageRank.

6 CONCLUSIONS

This first experiment with Warrick presented in this chapter demonstrated that reconstructing websites from the WI was feasible with typically a high level of success. It was also shown that the combination of multiple repositories was much more effective in reconstructing a website than if only a single repository from the WI was used. The second experiment showed how lister queries could be used to significantly improve the efficiency of website reconstructions. The final experiment attempted to capture the properties of a "typical" website and showed that those that were ranked as more popular by Google and were crawler friendly stood a better chance of being reconstructed than other websites.

All of these experiments have focused on recovering the rendered content of websites, but the generative functionality (the server components) have not been recoverable. Recovery of the server components would be very useful in restoring the functionality of a website that produced most of its content dynamically. The next chapter explores how the WI can be used to cache and recover both the rendered content and the generative functionality.

CHAPTER VIII

RECOVERING A WEBSITE'S SERVER COMPONENTS

Search engines and web archives do not have access to a website's server components (scripts, databases, etc.), so they only store the client-side representation of a website (see Figure 2 of Chapter III). In the event that a website is lost, only the client's view of the website can be recovered from the WI. While this may be useful for recovering much of the content of the website, it is not helpful for restoring the website's functionality. And because many dynamically produced pages reside in the deep web, a significant amount of content may not even be recoverable from the WI.

This chapter investigates how the WI can be used to store and recover a website's server components. Several techniques are explored to inject server components into crawlable portions of a website, thus using the WI as an off-site backup system. One of the implemented techniques is inspired by the use of steganography where information is hidden within a larger context [87]. This technique is implemented as a proof-of-concept, using open source software for a digital library.

1 GENERATING DYNAMIC WEB CONTENT

There are four different methods by which dynamic Web content can be produced. This summary is restated from [141] with some modification:

Server-side programs – A program or script executes on the server and generates a complete HTML (or other format) page which is then transmitted to the client. Common Gateway Interface (CGI), ASP.NET and Java servlets are popular technologies for server-side programs.

Embedded code with server-side execution – Code is inserted into static HTML files, and when the files are requested, the code is executed on the server, and the output replaces the code from the page. The newly created HTML page is then transmitted to the client. PHP: Hypertext Preprocessor (PHP), Active Server Pages (ASP) and SHTML are popular examples.

Client-side programs – A program which is downloaded from the server and executes on the client. It may need to be granted special privileges by the user in order to execute. The code is triggered by a web page and typically appears in a rectangular portion of the browser. Java applets, Flash and ActiveX controls are popular examples.

Embedded code with client-side execution – Code is inserted into static HTML files, and the code is executed by the client's browser. Parts of the code may optionally be placed in a separate file that is downloaded by the browser before execution. JavaScript and VBScript are popular examples.

Server-side programs and embedded code are completely hidden to web crawlers. Crawlers are exposed only to the output of the programs when links exist to generate such content are placed on the surface web. Content that is produced by server-side programs via a form submission is frequently hidden in the deep web (see Section 3.2 of Chapter II).

Although client-side programs are not hidden from web crawlers, search engines do not usually crawl them because they do not contain useful, indexable content. Embedded client code in a web page will remain cached by search engines if they are inserted into HTML files. If the code resides in a separate file, the search engine will frequently not cache it.

2 WHAT TO PROTECT

When considering which server components to protect, two approaches can be used. Using a *pessimistic approach*, the worst possible scenario is planned for, and as many server components are protected as possible: scripts, database contents, and crawlable content like images, style sheets, and PDFs that may not be stored in a canonical format by all members of the WI. Other resources could be included like the script interpreter, third party libraries, database software, and even the operating system if there is a possibility that any of these service components are in danger of being lost.

Using a more *optimistic approach*, protection is given to as few resources as possible without losing the functionality of the website. In this case only the customized scripts and database contents are protected, and it is assumed the WI would store any crawlable content in its canonical format, and all third-party supporting software would be readily available in an emergency. The optimistic approach puts a much lower storage burden on the WI although the chance of losing certain components is higher.

3 INJECTION MECHANICS

3.1 Techniques

There are three methods that could be used to inject server components into the WI:

1. **Exposing the raw components** - The server components of a website can be combined into a single compressed file (or multiple files) which is then placed on the web server and exposed to the WI.

2. **Robot vaults** - The server components are encoded into special crawlable pages which are created solely for WI crawlers.

3. **Dispersion through preexisting content** - The server components are injected into preexisting crawlable pages in an unobtrusive manner.

The first method is the simplest, but because search engines do not prefer compressed binary content, it is likely to only be captured by web archives[1]. This leaves the components particularly vulnerable to loss.

The second method has been explored by Traeger et al. [168]. They demonstrated how search engine caches could be used for backing-up a filesystem by creating special HTML pages designed only for storage; the pages did not contain readable information that would be useful to the public

[1]An exception to this is Google Code Search (`http://www.google.com/codesearch`) which makes source code available from compressed binary content found crawling the Web.

```
<html>
  <head>...</head>
  <body>...</body>

<!-- BEGIN_FILE_RECOVERY

U2FsdGVkX1/782IwTsCCkyNyw78O ...

END_FILE_RECOVERY -->
</html>
```

FIG. 51: Encoding of a file in HTML comments.

at large. While all members of the WI are likely to crawl these pages, search engines might consider them spam and would therefore not store them in their caches. Additionally, if a single page was not cached, the embedded file was lost.

The final method uses previously existing pages of a website for storing the server components which therefore does not run the risk of creating spam. Taking advantage of the fact that HTML allows comments of any size to be inserted within the page without affecting the appearance of the page, this approach can hide the encoding from the viewer of the page. Like steganography, the encodings are hidden from view. Additionally, erasure codes can be used to spread the encodings to multiple pages where recovery of only a subset of the pages allows complete recovery of the server component. This mitigates the risk incurred by the robot vaults method of requiring a single page to be stored by the WI in order to recover a single server component.

3.2 Dispersing Encoded Components

As performed in [168], the server components of a website can be injected into crawlable pages by base64 encoding the components and inserting the encodings into HTML comments as illustrated in Figure 51. The comments can be injected into any pre-existing page on the target website.

An erasure code transforms a message of b blocks into a message with n blocks (where $b < n$) where the original message can be recovered from a subset of those blocks [86]. Erasure codes have been used in a variety of applications like RAID systems [139], secret-sharing [86] and information dispersal [140]. Figure 52 illustrates how erasure codes can be used to inject a server file into crawlable content:

a) Use erasure codes to break an optionally encrypted server file into n blocks where recovery of any r blocks allows for complete recovery of the encrypted file.

b) Insert each block (base64 encoded) and metadata into n HTML files.

c) Wait for search engines (or other web repositories) to crawl and cache the HTML files.

Once a sufficient number of pages have been cached or archived, the server file can be recovered like so:

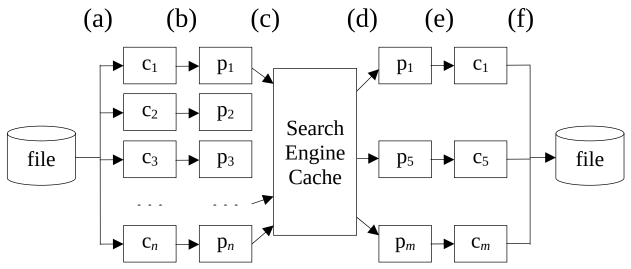


FIG. 52: Injecting and recovering a server file into and from the WI.

d) Recover as many HTML files from search engine caches as possible (m).

e) Extract available blocks and metadata from the HTML files.

f) If at least r blocks have been recovered (where $r \leq m$), reconstruct the server file (and optionally decrypt) using the erasure codes.

3.3 Segmenting Approaches

Since dynamically generated websites normally contain a large number of server files (sometimes much larger than the number of indexable pages), it is best to compress the server files together into one or more zip files. This reduces the amount of data injected into the WI and is practically easier to manage. When choosing how to segment the server files into zip files, two different approaches can be taken:

1) **All-or-nothing approach** – Compress all server files together into a single file and inject it into all available pages.

2) **Segmented approach** – Compress groups of server files into single files based on some sort of grouping mechanism (e.g., by directory, change rate, last modified, etc.) and inject them into subsets of available pages.

The first approach is the simplest to implement, but if the minimum number of blocks are not recovered, then nothing is recovered. Also, if one server file is changed, the single compressed file must be recreated and re-inserted into all the HTML pages. A web repository would then need to re-crawl and store a large number of pages it had already crawled in order to have stored all the blocks of the same version.

Although more complicated, the second approach allows recovery of individual groups of server components when a minimum number of blocks are recovered. Also, if one server file is changed, only the compressed file containing the changed server file must be recreated and re-inserted, thus isolating the changes to a subset of HTML files.

Figure 53 illustrates these two approaches. In the all-or-nothing approach, all the server files are compressed together into a single zipped file, and the file's blocks are then distributed to all

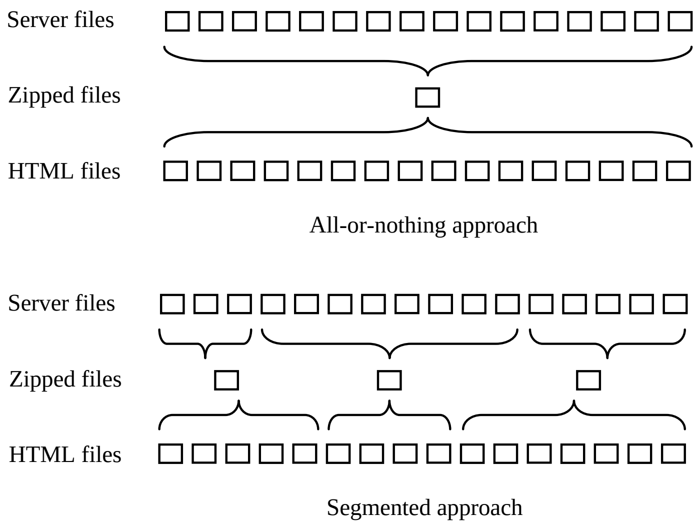


FIG. 53: Approaches to injecting server components into web pages.

the available pages. With the segmented approach, groups of server files are grouped together into separate zip files which are then distributed to subsets of pages. Larger zip files (in bytes) are allocated a larger number of pages. For each subset, r pages would need to be recovered (where r is specific to the subset) to reconstruct the zipped server file.

When a website is lost, complete knowledge of where each zipped file was injected and which zipped files are missing is ideal. For example, if none of the pages could be found in the WI that house the first zip file, knowledge that the file is missing (and knowledge of its contents) can help the recoverer when restoring the website's functionality. A **manifest** listing the filenames, paths, timestamp, sizes and details about where the components were injected can be produced and distributed along with the zipped server components. Inserting the manifest into every recoverable page would ensure its recoverability at the cost of re-building and re-injecting the manifest every time a file changed. To minimize these costs, the manifest could be inserted into one or more pages that have a high likelihood of being recovered, like the root page of the site or the pages a single hop away from the root page.

3.4 Choosing Block Sizes

When choosing values of n and r, care must be taken to ensure the block sizes are not too large since search engines will not cache the text of any HTML page beyond a certain threshold. Google's limit is 977 KB, Yahoo's limit is 214 KB, and Live's limit is 1 MB (see Section 3.5 of Chapter IV).

The block size can be calculated like so:

$$\text{block size} = \text{file size}/r \tag{25}$$

TABLE 23: Various r values (bold).

Zipped file size	Block size (before base64 encoding)		
	50 KB	100 KB	150 KB
1 KB	**1**	**1**	**1**
1 MB	**21**	**11**	**7**
100 MB	**2,048**	**1,024**	**683**
1 GB	**20,972**	**10,486**	**6,991**

As r increases, the size of the blocks will decrease, but the minimum number of HTML files to store r blocks increases. Therefore, a minimum value for r should be used which takes into account the search engine's cache limit t, the maximum block size or load z that is acceptable for adding to the website's pages, the total number of HTML resources h whose size is $< t - z$, and the size of the zipped server file s. The minimum value of r for a zipped file is calculated like so:

$$r = \lceil s/z \rceil \tag{26}$$

The z parameter is the only one which can be easily manipulated. Ideally a value of z should be picked that is as small as possible so the resulting pages are not overly large, causing them to download slower and creating more of a burden for the repository. If $r > h$ then there are not enough HTML pages to store the minimum number of blocks r. In this case the acceptable block size (z) would need to be increased, or if possible, more pages could be added to the website (thus increasing h) or the size of the server file (s) should be decreased by removing less essential server components. Once r has been calculated, n blocks can be created where $n = h$.

To illustrate how the size of the zipped file s and the block size limit z affects the number of HTML pages required to store the blocks, the value of r has been calculated for various values of s and z in Table 23. Since the average size of HTML files varies between 10-40 KB [13, 134], choosing block sizes above 170 KB is infeasible if wanting the repository with the least amount of available space (Yahoo) to store the blocks in their entirety.

3.5 Versioning

When web server components change or new ones are added, the injected pages must be updated with newly computed blocks. Additionally, metadata about each block needs to be kept along with the blocks for version control since only blocks that contain the same version of a server file can be used together to reconstruct the server file.

Since search engines often crawl the same pages only once a day or less often to be polite to the web server, blocks can be re-computed on a daily basis, ideally at times when server activity is low. Alternately, a web server module could be used to inject pages on-the-fly with encoded blocks depending on the identity (IP address or user agent) of the requester. This would allow smaller pages to be served to regular users and larger pages with the encoded blocks to be served to search engines. This technique of serving clients different pages based on their identity is called *cloaking*, and it must be used with caution since most search engines disapprove of it [176].

3.6 Security Considerations

The WI is accessible to everyone; this has its benefits and its drawbacks. If a website owner was to die and their site became lost over time, the site's users may want to recover the site so it would remain accessible to the site's user community. Having the server components readily accessible from the WI would dramatically decrease the effort involved in making the site functional once again.

On the other hand, the site may contain sensitive data like passwords, account IDs, credit card numbers, etc. which the site owner may *never* want accessible to a third party. In this case the server components could be encrypted with a private key, and death of the site owner (and knowledge of the key) would likely prevent the functionality of the site from ever being recovered (at least until the encryption was broken). The site owner should also consider what would happen if the key were ever forgotten or compromised.

4 EXPERIMENTS

To validate the feasibility of recovering a website's server components from the WI, two experiments were conducted using websites that dynamically produce a majority of their content. The experiments were limited to digital libraries (DLs) which were running GNU EPrints [53], a popular open source DL package that is composed of Perl scripts, configuration files and MySQL database. Ten randomly selected DLs running EPrints were crawled and reconstructed to see how much content might be recovered for a typical EPrints DL. The server components for these DLs were not recoverable since they did not implement any injection methods. In the second experiment, a test DL was created, the Monarch Repository, using EPrints software. It contained 100 PDF resources and embedded server encodings. The DL was reconstructed on a weekly basis for 24 weeks.

4.1 Reconstructing 10 Digital Libraries

The first experiment examined how much of an EPrints DL could be recovered from the WI if the DL was to suddenly disappear. Ten randomly selected DLs running EPrints were selected from the Registry of Open Access Repositories (ROAR) [143]. Using the same methodology from the reconstruction experiments of Chapter VII, the DLs were crawled and then reconstructed with Warrick, and their reconstructions were compared to the crawled sites. Of course the reconstructions were only able to recover the client-side representation of the sites and none of the server components.

A distinguishing characteristic about DLs is that they typically curate a large number of non-HTML resources like historical images, academic documents, and the like. They maintain a variety of metadata about each resource for purposes of provenance. For many DLs, PDF documents are the primary resources being curated, and much of the metadata can be re-generated as long as the PDF remains available. The ten DLs selected in this experiment contained a variety of resource formats, but they all contained a number of PDFs.

Table 24 lists the ten DLs in order of total resources (HTML, images, PDFs, etc.) along with

TABLE 24: Ten reconstructed digital libraries.

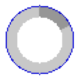
Digital Library	All resources		PDFs				Diff	Recon
	Total	Recov	Total	Recov	as PDF	as HTML	vector	diag
1. eprints.libr.port.ac.uk	176	97.2%	36	94.4%	2.9%	97.1%	(0.722, 0.028, 0.018)	
2. archiviomarini.sp.unipi.it	222	86.0%	47	46.8%	13.6%	86.4%	(0.658, 0.131, 0.000)	
3. eprints.erpanet.org	272	82.0%	66	60.6%	75.0%	25.0%	(0.467, 0.062, 0.229)	
4. open.ekduniya.net	452	95.4%	87	81.6%	4.2%	95.8%	(0.699, 0.046, 0.005)	
5. brief.weburb.dk	458	88.4%	143	82.5%	72.0%	28.0%	(0.286, 0.109, 0.022)	
6. eprints.bbk.ac.uk	771	93.4%	331	91.8%	1.3%	98.7%	(0.720, 0.065, 0.004)	
7. eprints.vu.edu.au	1192	98.1%	314	96.5%	84.2%	15.8%	(0.316, 0.017, 0.004)	
8. eprints.lse.ac.uk	1336	95.0%	513	89.5%	2.8%	97.2%	(0.821, 0.046, 0.003)	
9. www.cbmh.ca	1695	99.6%	673	99.6%	12.2%	87.8%	(0.602, 0.004, 0.003)	
10. bnarchives.yorku.ca	2130	30.3%	272	84.6%	47.4%	52.6%	(0.173, 0.690, 0.012)	
Average	870.4	86.5%	248.2	82.8%	31.6%	68.4%	(0.546, 0.120, 0.030)	

TABLE 25: Composition of Monarch DL.

Type	Total	Distribution
HTML	123	34.0%
Images	110	30.4%
PDF	100	27.6%
Style sheet	26	7.2%
Other	3	0.8%
Total	362	100%

the difference vectors and reconstruction diagrams. On average, 87% of the DL's resources were recovered and 83% of the PDFs. Unfortunately, the PDFs recovered were much more likely to be in HTML format rather than their native format (68% vs. 32%). This may be acceptable for PDFs that are purely textual, but loss of figures and other images in the PDF-to-HTML conversion process are usually problematic in the face of complete loss (see Section 2.2 in the previous chapter for a complete discussion of this issue).

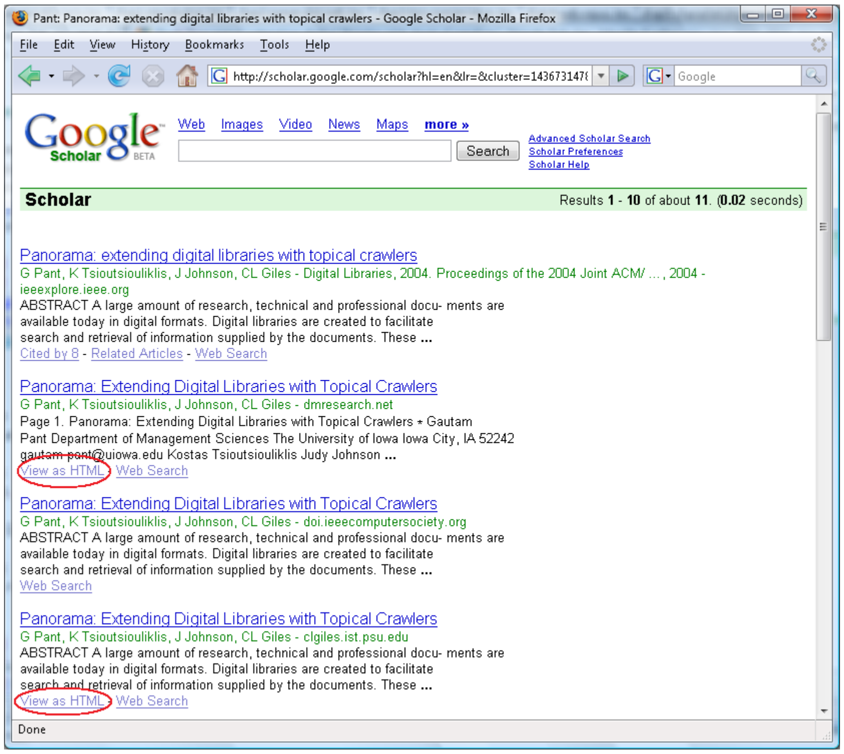
4.2 Recovering Server Components From a DL

Setup

The second experiment was a proof-of-concept, demonstrating the server injection techniques presented in the previous sections could be used to recover the server components of a website. A DL was created using EPrints and populated with 100 academic papers in PDF format and metadata from the fields of Web technologies and information retrieval. The PDFs were all previously accessible on the Web. Since the PDFs were accessible to a web crawler, it is possible that a majority of them would duplicate PDFs already cached by the search engines. However, Google Scholar frequently provides many alternate locations of the same PDF and maintains multiple cached copies as evidenced by Figure 54. Although Warrick does not pull directly from Google Scholar, an investigation by the author revealed that all the papers cached from the example in Figure 54 were also accessible from the Google cache used by Warrick.

An example web page from the Monarch DL (as it was called) is shown in Figure 55. If the user places the cursor over the PDF icon, a preview image of the document appears. This PNG image can easily be found by a web crawler. Although each page notified the reader that the DL was a "test repository," it was decided during the design of the experiment that a search engine would be unlikely to refuse caching the page because of the presence of that phrase. The composition of the DL as seen by a typical web crawler is shown in Table 25.

The optimistic approach was adopted to preserve the server components: only the Eprints software (Perl scripts), configuration files and database contents (extracted with the mysqldump tool) were preserved. The size of the software was approximately 3.3 MB (uncompressed), 2 MB for config and other miscellaneous files and 650 KB for the database contents. Tarring and compressing all these files together with gzip produced a 1 MB file. The PDFs occupied 41 MB of space and would


Pant: Panorama: extending digital libraries with topical crawlers - Google Scholar - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://scholar.google.com/scholar?hl=en&lr=&cluster=143673147
Web Images Video News Maps more »
Search
Advanced Scholar Search
Scholar Preferences
Scholar Help
Done

Scholar Results 1 - 10 of about 11. (0.02 seconds)

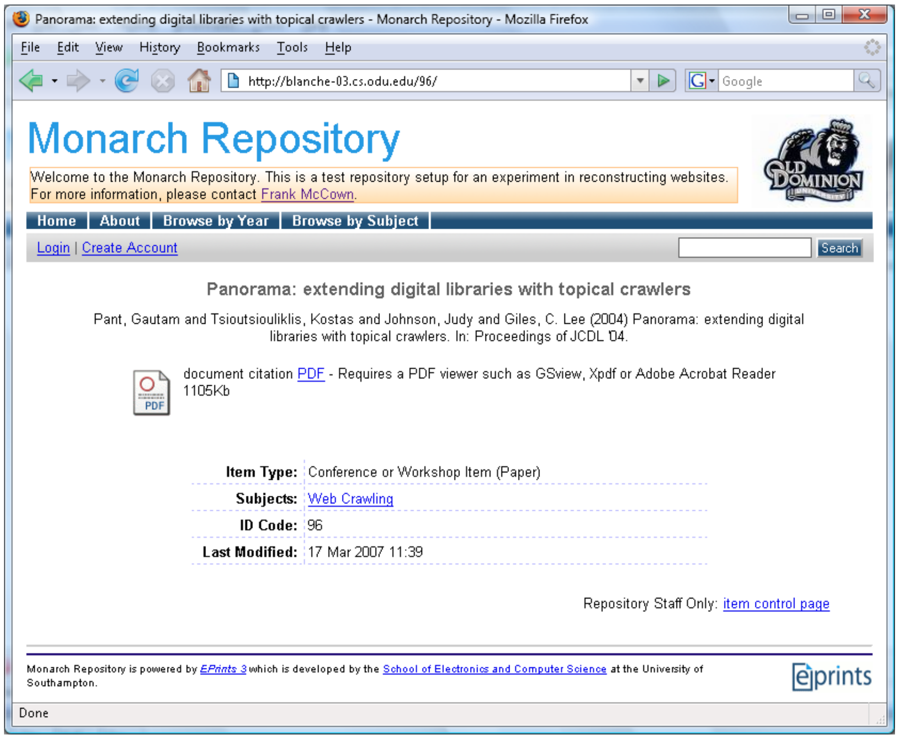
Panorama: extending digital libraries with topical crawlers
G Pant, K Tsioutsiouliklis, J Johnson, CL Giles - Digital Libraries, 2004. Proceedings of the 2004 Joint ACM/ ..., 2004 - ieeexplore.ieee.org
ABSTRACT A large amount of research, technical and professional docu- ments are available today in digital formats. Digital libraries are created to facilitate search and retrieval of information supplied by the documents. These ...
Cited by 8 - Related Articles - Web Search

Panorama: Extending Digital Libraries with Topical Crawlers
G Pant, K Tsioutsiouliklis, J Johnson, CL Giles - dmresearch.net
Page 1. Panorama: Extending Digital Libraries with Topical Crawlers ∗ Gautam Pant Department of Management Sciences The University of Iowa Iowa City, IA 52242 gautam-pant@uiowa.edu Kostas Tsioutsiouliklis Judy Johnson ...
View as HTML - Web Search

Panorama: Extending Digital Libraries with Topical Crawlers
G Pant, K Tsioutsiouliklis, J Johnson, CL Giles - doi.ieeecomputersociety.org
ABSTRACT A large amount of research, technical and professional docu- ments are available today in digital formats. Digital libraries are created to facilitate search and retrieval of information supplied by the documents. These ...
Web Search

Panorama: Extending Digital Libraries with Topical Crawlers
G Pant, K Tsioutsiouliklis, J Johnson, CL Giles - clgiles.ist.psu.edu
ABSTRACT A large amount of research, technical and professional docu- ments are available today in digital formats. Digital libraries are created to facilitate search and retrieval of information supplied by the documents. These ...
View as HTML - Web Search

FIG. 54: Google Scholar maintains 11 versions of the same paper, some of which are cached.


Panorama: extending digital libraries with topical crawlers - Monarch Repository - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://blanche-03.cs.odu.edu/96/
Home About Browse by Year Browse by Subject
Login | Create Account
Search
Done

Monarch Repository

Welcome to the Monarch Repository. This is a test repository setup for an experiment in reconstructing websites. For more information, please contact Frank McCown.

Panorama: extending digital libraries with topical crawlers

Pant, Gautam and Tsioutsiouliklis, Kostas and Johnson, Judy and Giles, C. Lee (2004) Panorama: extending digital libraries with topical crawlers. In: Proceedings of JCDL '04.

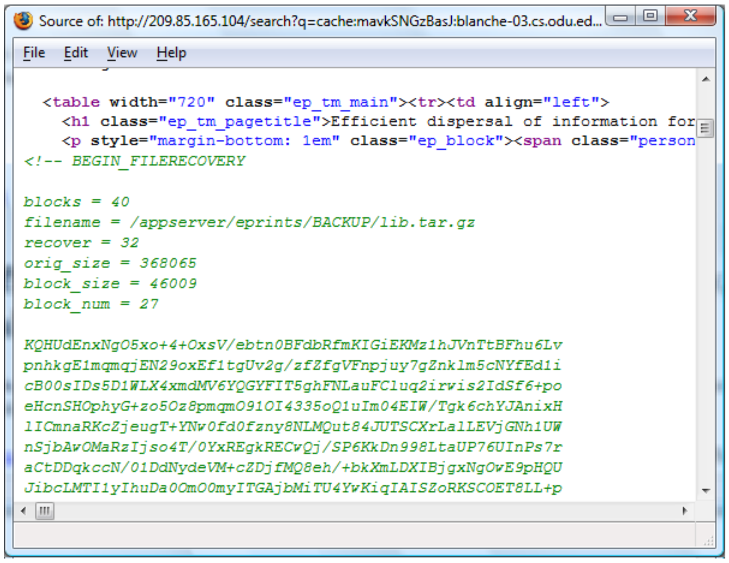
document citation PDF - Requires a PDF viewer such as GSview, Xpdf or Adobe Acrobat Reader 1105Kb

Item Type:	Conference or Workshop Item (Paper)
Subjects:	Web Crawling
ID Code:	96
Last Modified:	17 Mar 2007 11:39

Repository Staff Only: item control page

Monarch Repository is powered by EPrints 3 which is developed by the School of Electronics and Computer Science at the University of Southampton.

FIG. 55: Monarch DL screen shot.


Source of: http://209.85.165.104/search?q=cache:mavkSNGzBasJ:blanche-03.cs.odu.ed...
File Edit View Help

```
  <table width="720" class="ep_tm_main"><tr><td align="left">
    <h1 class="ep_tm_pagetitle">Efficient dispersal of information for
    <p style="margin-bottom: 1em" class="ep_block"><span class="person
<!-- BEGIN_FILERECOVERY

blocks = 40
filename = /appserver/eprints/BACKUP/lib.tar.gz
recover = 32
orig_size = 368065
block_size = 46009
block_num = 27

KQHUdEnxNgO5xo+4+OxsV/ebtn0BFdbRfmKIGiEKMz1hJVnTtBFhu6Lv
pnhkgE1mqmqjEN29oxEf1tgUv2g/zfZfgVFnpjuy7gZnklm5cNYfEd1i
cB00sIDs5D1WLX4xmdMV6YQGYFIT5ghFNLauFCluq2irwis2IdSf6+po
eHcnSHOphyG+zo5Oz8pmqm091OI4335oQ1uIm04EIW/Tgk6chYJAnixH
lICmnaRKcZjeugT+YNw0fd0fzny8NLMQut84JUTSCXrLalLEVjGNh1UW
nSjbAvOMaRzIjso4T/0YxREgkRECvQj/SP6KkDn998LtaUP76UInPs7r
aCtDDqkccN/01DdNydeVM+cZDjfMQ8eh/+bkXmLDXIBjgxNgOvE9pHQU
JibcLMTI1yIhuDa0Om0OmyITGAjbMiTU4YwKiqIAISZoRKSCOET8LL+p
```

FIG. 56: Encoding of a server file in HTML comments from the Monarch DL.

have added 30 MB to the compressed tar file. An example of the injected server components from the Monarch DL is shown in Figure 56.

For 19 weeks, the Monarch DL was crawled (with Heritrix) and reconstructed (with Warrick using the Comprehensive policy) at the end of the week as was performed in the reconstruction experiments from the previous chapter. The crawls were matched with the reconstructions to produce an accurate assessment as to how much of the website was being successfully reconstructed each week.

Several techniques were used throughout the experiment to test the all-or-nothing and segmented approaches and website updates. Initially, the segmented approach was used to create ten compressed tar files, one for each directory of the Eprints software and the database. The files were dispersed among the 123 HTML pages according to size, so larger files were allocated more pages than smaller files which kept the block sizes from becoming too large. The manifest was encoded and placed in the root page of the DL. The resulting pages averaged approximately 60 MB in size, far below Yahoo's 214 KB size limit. On May 19, 2007, links pointing to the Monarch DL were placed on three previously indexed pages on the `www.cs.odu.edu` website in order to advertise the existence of the DL to the WI.

After nine weeks, a single link was added to the root page which pointed to a single web page which had previously been unlinked (accidentally by the author). This unlinked page contained one of the encoded blocks which was needed to recover one of the ten server files. Three weeks later, several of the encoded blocks were re-arranged and re-allocated to eight web pages to simulate an update to the website. And three weeks later, the all-or-nothing approach was tested by creating a single gzipped tar file for all the DL server components and distributing it among the entire website. A "This page was modified on *date and time*" notice was also added to each page in this last phase to encourage the search engines to re-cache all the pages.

TABLE 26: Updates and changes made throughout the experiment.

Week	Summary
1	Ten segments are distributed throughout the website.
10	Single link is added to unlinked web page.
13	Subset of blocks are reallocated to eight web pages.
16	All server components are reallocated to all web pages.
19	Website is taken off-line.

Finally, the DL was taken off-line on week 19 to simulate the loss of the website. The DL was configured to return an HTTP 404 (not found) response to every URL request except for the root page which contained a notice that the website had been taken off-line. These steps are summarized in Table 26.

Results

Figure 57 shows how much of the Monarch DL was recovered each week. The 362 resources are ordered on the y-axis by resource type. A square dot indicates that the resource was recovered that week. The percentage of resources recovered each week is plotted in Figure 58. Just a few days after a link was created to the Monarch DL, Google discovered and crawled the website, making a quarter of the discovered pages available from their cache. Each week the percentage of resources recovered from Google increased. Live and Yahoo had crawled a number of resources from the DL a few weeks after Google, but neither search engine made anything available from their cache until later in the experiment as will be discussed shortly. Several style sheets (categorized as "other") where recovered from IA beginning on week 17, only two months after they first crawled the website.

On week 7 and 8, 100% of the HTML resources were recovered (minus one unlinked page), all from Google. However, on week 9 several HTML resources which were once accessible from Google's cache were no longer cached, and a 100% recovery rate for HTML resources was not observed again. As illustrated by Figure 57, a small number of URLs tended to fluctuate in and out of Google's cache throughout the experiment. For example, the URL `http://blanche-03.cs.odu.edu/118/` was cached on weeks 1, 2, 4–9, 13–17.

Although Google crawled a large number of images on week 7, only a handful of images were recovered from Google by week 15; interestingly, one of them appeared to be a blank image (Figure 59). Live had cached 18 images by week 15, but Warrick was not able to recover them due to a bug in the Live API [109].

Unlike the high percentage of recovered PDFs from the first experiment (Table 24), Figure 58 shows only a small percentage of the Monarch's PDFs were made available from Google's cache; none were accessible from Live and Yahoo. As mentioned earlier, a possible explanation for the low percentage of cached PDFs may be that a majority of the PDFs were recognized to be duplicates of PDFs already cached.

Figure 60 shows the distribution of the Monarch DL resources that were vulnerable, replicated, endangered and unrecoverable each week. The DL was taken off-line at week 19 when all replicated

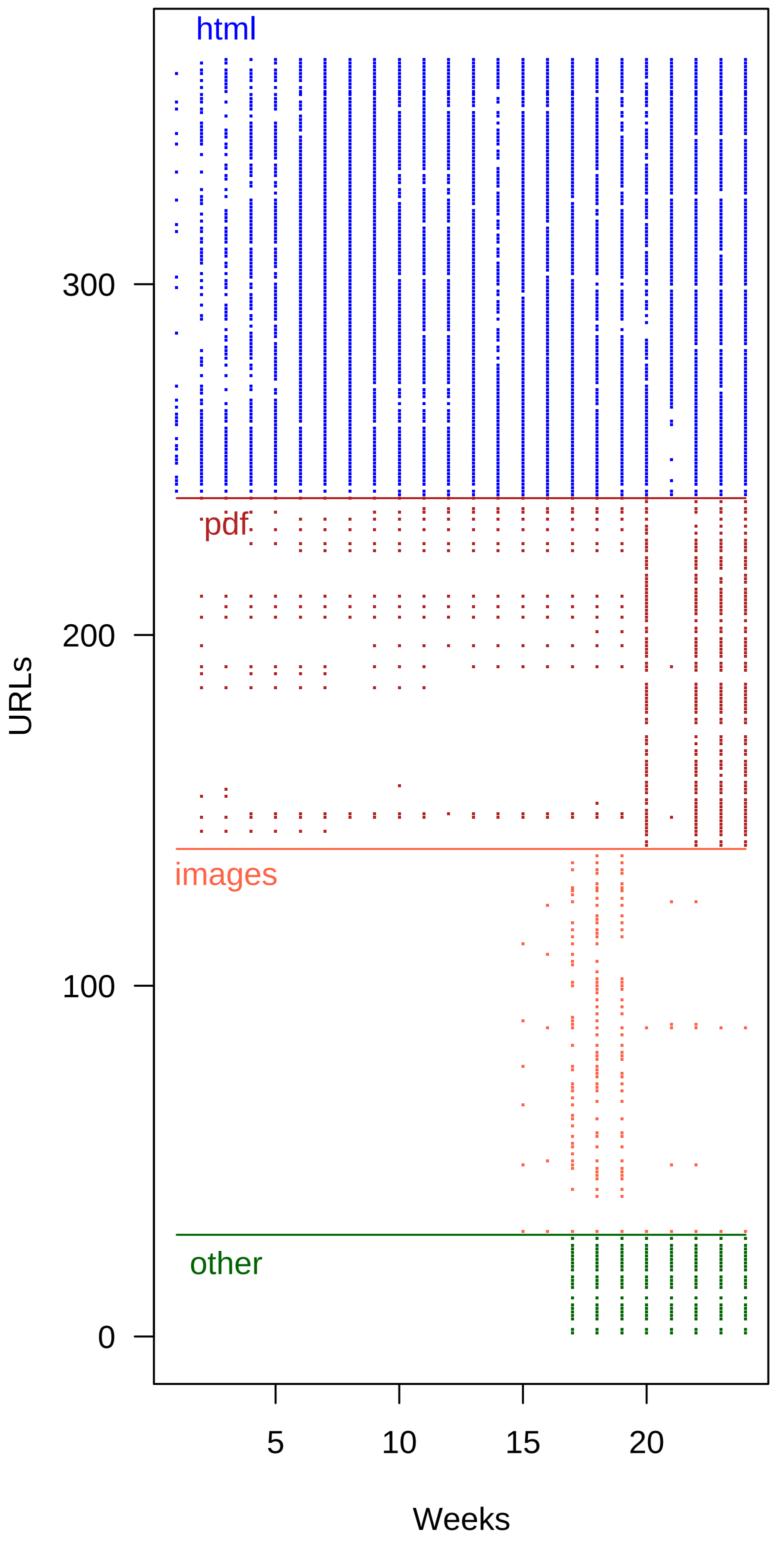


FIG. 57: Recovered DL resources each week.

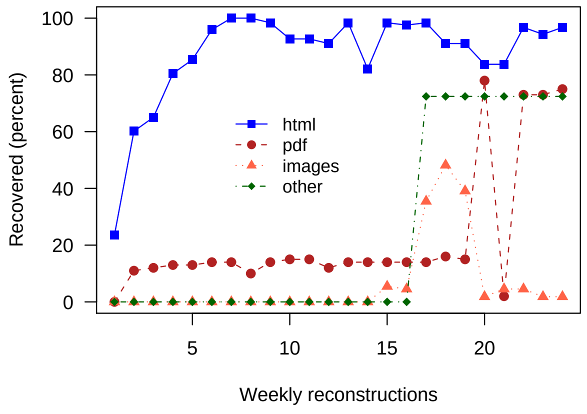


FIG. 58: Percent of recovered DL resources each week.

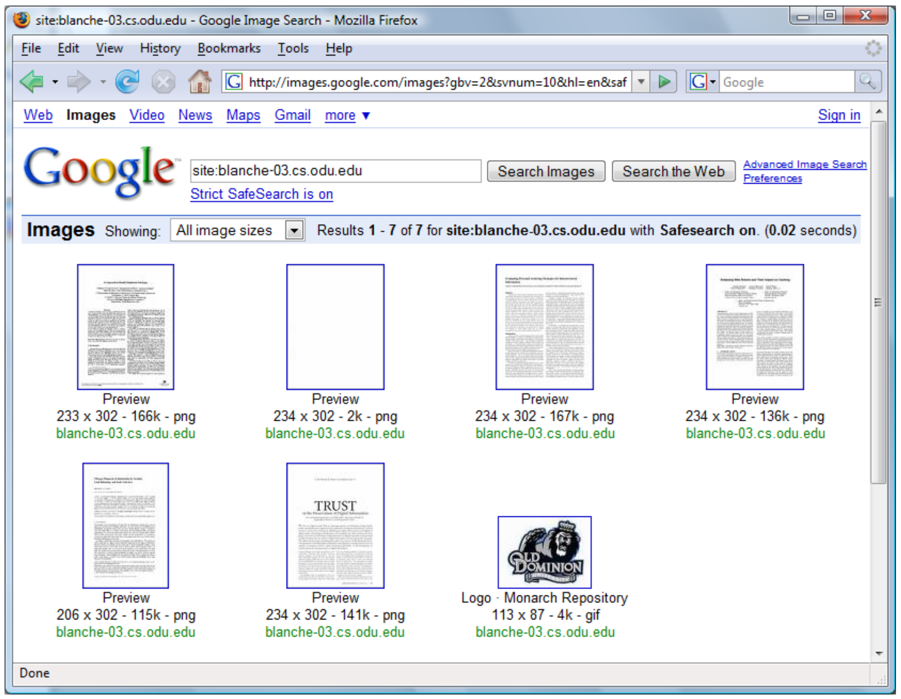


FIG. 59: Images from Monarch DL cached by Google.

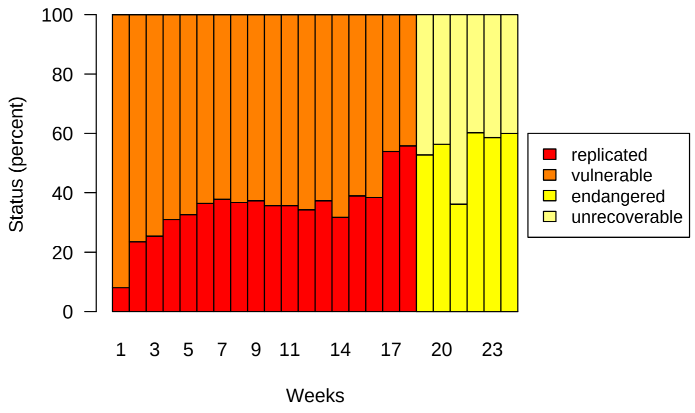


FIG. 60: Availability of Monarch DL resources.

resources became endangered and all vulnerable resources became unrecoverable. At this point, the percentage of unrecoverable resources continued to increase throughout the rest of the experiment.

The server components were far more recoverable than the resources making up the client view. Table 27 lists the percentage of recovered HTML pages and server files each week and the contribution rate of each repository throughout the experiment. Some of the weeks indicate an update to the website was performed prior to that week's reconstruction. Almost 100% of the server components were recoverable from the WI just two weeks after the experiment began, despite having only recovered 60% of the HTML pages. It was not until week 10, when a link was posted to an unlinked web page, that all 100% of the server files could be recovered.

When several encoded blocks were reallocated to eight web pages at the beginning of week 13, the percent of recoverable server files dropped accordingly. The web server logs indicated that Google, Live and Yahoo continued to crawl the affected pages after the changes were made, but the cached pages were not being updated. Since the vast majority of pages were dynamically generated and did not return a Last-Modified HTTP header, the search engines may have performed some type of processing on the crawled pages which ignored changes to HTML comments only. It took nearly three weeks before three of the eight pages were recoverable which allowed all 100% of the server files to be recovered once again.

The the end of week 16, none of the server components were recoverable. This was due to the complete reallocation of all the server components at the beginning of the week which required the altered pages to be re-crawled and cached. Since the WI seemed to mostly ignore changes to only HTML comments in the previous weeks, a single line of text was added to each page stating "This page was modified on *date and time*." This modest change seemed to encourage the WI to re-cache the altered pages because the following week 100% of the server files were once again recovered.

When the DL was taken off-line at the beginning of week 19, 100% of the server components were recoverable at the end of the week. All 100% continued to be recoverable through week 24 due, mainly because of Live's contribution. Google and Yahoo contributed far fewer resources after

TABLE 27: Recovered server files and repository contributions each week.

Week	Recovered HTML (%)	Recovered server files (%)	Contributors (%)			
			Google	Live	Yahoo	IA
1	23.6	59.4	100	0	0	0
2	60.2	99.6	100	0	0	0
3	65.0	94.6	98.9	1.1	0	0
4	80.5	99.7	100	0	0	0
5	85.4	99.7	99.2	.8	0	0
6	95.9	99.7	99.2	.8	0	0
7	100.0	99.7	97.1	2.9	0	0
8	100.0	99.7	98.5	1.5	0	0
9	98.4	99.7	98.5	1.5	0	0
10[1]	92.7	100	100	0	0	0
11	92.7	100	88.4	11.6	0	0
12	91.1	100	89.5	10.5	0	0
13[2]	98.4	92.1	85.2	14.8	0	0
14	82.1	92.1	69.6	17.4	13.0	0
15	98.4	100	81.6	14.2	4.2	0
16[3]	97.6	0	80.6	11.5	7.9	0
17	98.4	100	68.7	8.2	12.3	0.1
18	91.1	100	66.3	6.9	16.3	0.1
19[4]	91.1	100	47.1	5.8	36.1	0.1
20	83.7	100	2.5	66.2	21.17	0.1
21	83.7	100	9.9	52.7	21.4	0.2
22	96.7	100	3.7	79.4	7.3	0.1
23	94.3	100	3.8	83.5	2.8	0.1
24	96.7	100	4.1	82.9	3.2	0.1

[1]Single link is added to unlinked web page.
[2]Subset of blocks are reallocated to eight web pages.
[3]All server components are reallocated to all web pages.
[4]Website is taken off-line.

week 19 because most of the missing DL content was purged from their caches; IA remained steady, contributing mainly CSS files each week.

5 DISCUSSION

The injection technique using erasure codes seems to have been very effective for the proof-of-concept. Even though the Monarch DL had never been crawled before, nearly all of its server components were recoverable within two weeks of it going live. This means the Monarch DL would have lost *none* of its dynamically-produced web pages had it been lost whereas the DLs from Table 24 would have lost an average of 13% of their pages. A month after the DL was "lost," all the server components remained recoverable.

As mentioned earlier, injecting server components into the WI has some disadvantages. First, it puts an additional load, however small, onto the web repositories that they may not want to take-on. If the injection technique were adopted widely, web repositories may take steps to remove suspicious comments from crawled pages. Second, the additional payload to each page makes them download slower. This is certainly an issue where bandwidth is limited (e.g., developing countries and mobile clients). Third, some alteration of the visible contents of injected pages may be required to induce the WI to refresh its holdings. Forth, setting up such an injection mechanism requires a small amount of work by the webmaster before the website is lost; this is opposed to the lazy preservation approach of "no work required." And finally, it may not be wise to place private data into publicly accessible locations like the WI, even if encryption is used.

Despite these limitations, there is a need for a safety net like lazy preservation for preserving the server components of a lost website so re-enabling its functionality and accessing its deep web resources is possible. Perhaps a business model could be developed that provided an insurance policy for lost websites. An organization like Google with large amounts of disk space (and large amounts of public trust) could automatically back-up any number of web servers without cost to the webmasters. In the advent of a loss, the organization could recover the lost web server components for a fee. Considering the cost of re-building a dynamic website from scratch, a webmaster may be willing to pay a large amount of money to recover a lost website, enough to cover the collective cost of storing so much data. For such a mechanism to work, it would need to be easily enabled by the "laziest" of webmasters.

6 CONCLUSIONS

Several techniques for recovering the server components of a website from the WI were developed in this chapter. One promising technique was implemented in an EPrints digital library, and it was demonstrated that nearly all the DL's server components could be recovered from the WI just two weeks after the DL was made accessible on the surface web. Pages were refreshed in the WI a few weeks after they were modified, allowing the modified server components to be completely recoverable. This injection technique may not be ideal for every type of website, but it does demonstrate that the lazy preservation approach to preserving server components is at least feasible with a small amount of work on behalf of the webmaster.

CHAPTER IX

CONCLUSIONS AND FUTURE WORK

1 CONCLUSIONS

As the Web becomes a hub for our daily activities, curation and preservation of Web-based material imposes an increasing burden on individuals and institutions. It is likely that the Web will continue to suffer the effects of link rot for the foreseeable future, and content that is of significant importance to a variety of audiences will continue to be lost when they are unprotected, abandoned or destroyed by their creators. Lazy preservation is an alternate model of preservation which harnesses the collective refreshing and migration facilities of the Web Infrastructure. Rather than preserving a limited collection of known value, lazy preservation provides Web-scale preservation for items of unknown importance.

This dissertation has demonstrated how lazy preservation tools like Warrick can reconstruct lost websites from the WI with generally positive results. The experiments from Chapter IV characterized the types of resources found in the WI and the behavior the WI exhibits when ingesting new web content. Chapter V laid the groundwork for how the WI can be crawled, and Chapter VI showed how a web-repository crawler like Warrick can be implemented. The experiments of Chapter VII verified that using the WI as a collective whole was much more effective than using individual WI members alone and that a majority of a 'typical' website's resources can be recovered from the WI. This dissertation concluded with an experiment showing how the functionality of a dynamically produced website to be recovered from WI.

There are numerous up-front preservation mechanisms that individuals could use to protect their websites from loss (Chapter II), but none of them are foolproof, and all require the creator to do some amount of work *before* the loss has occurred. Usage data collected at ODU shows that Warrick is currently being used to reconstruct an average of 108 websites per month. Perhaps tools like Warrick (after-loss recovery) provide greater immediate gratification than up-front preservation applications. Until the fundamental qualities of the Web change that make publication permanent, it is likely that a safety net like lazy preservation will always be needed.

2 CONTRIBUTIONS

This dissertation makes eight significant contributions to the field of digital preservation:

1) This dissertation identifies the pervasive problem of website loss that has been previously addressed by *a priori* solutions. A novel solution called lazy preservation is offered which allows after-the-fact recovery for little to no work required for the content creator.

2) The WI is characterized in terms of its behavior to consume and retain new web content, and the types of resources it contains. The overlap between the search engine caches of Ask, Google, Live Search and Yahoo are compared with the Internet Archive.

3) A model for resource availability is developed which defines the stages a web resource goes through from its initial creation to its potential unavailability.

4) A new type of crawler is introduced: a web-repository crawler. The crawler's architecture is presented, interfaces for crawling web repositories are developed, rules for canonicalizing URLs between web repositories are supplied, and three crawling policies are evaluated experimentally.

5) A statistical model is developed which can be used to measure the characteristics of a reconstructed website along with a reconstruction diagram which can graphically summarize reconstruction success. Other formulas are also developed which can account for varying degrees of reconstruction success.

6) Experimental results confirm that websites that are "crawler friendly" are the most recoverable from the WI. The three most significant variables that determine how successfully a web resource will be reconstructed from the WI is Google's PageRank for the website's root page, the number of hops from the root page to the resource and how long the resource has been available from the same URL.

7) A novel solution to recovering a website's server components is proposed and experimentally validated.

8) A website reconstruction service was created which is currently being used by the public to reconstruct more than 100 lost websites a month.

3 FUTURE WORK

There are a number of directions the work presented in this dissertation may be expanded.

3.1 Warrick Improvements

There are several improvements which could be made to Warrick which would increase its effectiveness at recovering lost websites:

- Warrick currently uses only four web repositories for locating missing resources. Adding additional repositories like Furl or Hanzo:Web would likely improve coverage.
- Warrick's detection of URLs within HTML resources, JavaScript and Flash could be improved to find potentially more recoverable resources.
- Warrick is currently unable to detect soft 404s, web pages that are returned to a crawler with an HTTP 200 response instead of the proper 404 (not found) response [15]. These pages are sometimes stored in the WI when a website removes previously existing pages or when a website's domain name expires. Warrick could use an algorithm like the one presented in [15] to detect and reject these pages when reconstructing a website that has just been lost.

3.2 Determining Loss

A potential area for future work involves automatically determining when a website has been lost. If such a determination could be made, the website could be reconstructed immediately and saved by a memory institution. Instead of preemptively saving every website, only websites which have just been lost could be saved, thus decreasing the storage requirements needed for preserving a large number of websites. This would require monitoring websites on a periodic basis and applying trend analysis to detect significant shifts in the page "aboutness," similar to the techniques applied in the Walden's Path Project [47].

Such a technique could also be used when reconstructing a website when repository copies of a website are tainted. For example, when a website domain name is hijacked, its contents change significantly. If Warrick could detect these significant changes, it could reject tainted resources from deep repositories like IA.

3.3 Using Browser Caches

Browser caches are a significant source of potentially unrecoverable web resources. In fact, the author once used a client's browser cache to recover significant portions of the client's lost website when very little of the website was recoverable from the WI. There are several proposed methods of accessing browser caches in peer-to-peer systems which, if implemented widely, could be used as a starting point for enhancing Warrick's ability to recover lost web resources [102, 177, 178].

3.4 Overlap Studies

The search engine/Internet Archive overlap experiment of Chapter IV used URLs sampled from the search engines. Using a different set of URLs sampled from user requests would provide a better measure for determining what URLs are missing from search engine caches and IA.

BIBLIOGRAPHY

[1] Abiteboul, S., Cobena, G., Masanes, J., Sedrati, G.: A first experience in archiving the French Web. In: Proceedings of the 6th European Conference on Research and Advanced Technology for Digital Libraries, pp. 1–15. Springer (2002)

[2] Adamic, L.A., Huberman, B.A.: Zipf's law and the Internet. Glottometrics **3**, 143–150 (2002). `http://www.hpl.hp.com/research/idl/papers/ranking/adamicglottometrics.pdf`

[3] Adobe Flash. `http://www.adobe.com/products/flash/`

[4] von Ahn, L., Blum, M., Hopper, N., Langford, J.: CAPTCHA: Using hard AI problems for security. In: EUROCRPYT 2003: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (2003)

[5] Amazon Simple Storage Service (Amazon S3). `http://aws.amazon.com/s3`

[6] Apache Tomcat. `http://tomcat.apache.org/`

[7] Archive-It. `http://www.archive-it.org/`

[8] Arms, W., Huttenlocher, D., Kleinberg, J., Macy, M., Strang, D.: From Wayback Machine to Yesternet: New opportunities for social science. In: Proceedings of the 2nd International Conference on e-Social Science (2006)

[9] Arms, W.Y., Aya, S., Dmitriev, P., Kot, B.J., Mitchell, R., Walle, L.: Building a research library for the history of the web. In: JCDL '06: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 95–102 (2006). doi: 10.1145/1141753.1141771

[10] Arvidson, A., Persson, K., Mannerheim, J.: The Kulturarw3 Project - The Royal Swedish Web Archiw3e - An example of "complete" collection of web pages. In: Proceedings of the 66th IFLA Council and General Conference (2000). `http://www.ifla.org/IV/ifla66/papers/154-157e.htm`

[11] Backup.com. `http://www.backup.com/`

[12] Baeza-Yates, R., Castillo, C.: Crawling the infinite web: five levels are enough. In: Proceedings of the third Workshop on Web Graphs (WAW), vol. 3243, pp. 156–167 (2004)

[13] Baeza-Yates, R., Castillo, C., Efthimiadis, E.N.: Characterization of national web domains. ACM Trans. on Internet Technol. **7**(2), 9 (2007). doi: 10.1145/1239971.1239973

[14] Bailey, S., Thompson, D.: UKWAC: Building the UK's first public web archive. D-Lib Mag. **12**(1) (2006). doi: 10.1045/january2006-thompson

[15] Bar-Yossef, Z., Broder, A.Z., Kumar, R., Tomkins, A.: Sic transit gloria telae: towards an understanding of the web's decay. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web, pp. 328–337 (2004). doi: 10.1145/988672.988716

[16] Bar-Yossef, Z., Gurevich, M.: Random sampling from a search engine's index. In: WWW '06: Proceedings of the 15th International Conference on World Wide Web, pp. 367–376 (2006). doi: 10.1145/1135777.1135833

[17] Bar-Yossef, Z., Gurevich, M.: Efficient search engine measurements. In: WWW '07: Proceedings of the 16th International Conference on World Wide Web, pp. 401–410 (2007). doi: 10.1145/1242572.1242627

[18] Bar-Yossef, Z., Keidar, I., Schonfeld, U.: Do not crawl in the DUST: Different URLs with similar text. In: WWW '07: Proceedings of the 16th International Conference on World Wide Web, pp. 111–120 (2007). doi: 10.1145/1242572.1242588

[19] Bausch, S.: Nielsen//Netratings announces May U.S. search share rankings (2007). `http://www.nielsen-netratings.com/pr/pr_070620.pdf`

[20] Berghel, H.: Responsible web caching. Communications ACM **45**(9), 15–20 (2002). doi: 10.1145/567498.567514

[21] Bergman, M.K.: The deep web: Surfacing hidden value. J. Electron. Publishing (2001). `http://www.press.umich.edu/jep/07-01/bergman.html`

[22] Berners-Lee, T.: Cool URIs don't change (1998). `http://www.w3.org/Provider/Style/URI.html`

[23] Berners-Lee, T., Fielding, R., Masinter, L.: Uniform Resource Identifier (URI): Generic syntax. RFC 3986 (2005). `http://www.ietf.org/rfc/rfc3986.txt`

[24] The Beryl Project: Maybe Google cache can return some data (2006). `http://forum.beryl-project.org/viewtopic.php?t=29`

[25] Bibliotheca Alexandrina. `http://www.bibalex.org/ISIS/archive_web.htm`

[26] Bott, E.: Windows XP Backup made easy (2003). `http://www.microsoft.com/windowsxp/using/setup/learnmore/bott_03july14.mspx`

[27] Brandman, O., Cho, J., Garcia-Molina, H., Shivakumar, N.: Crawler-friendly web servers. SIGMETRICS Performance Evaluation Review **28**(2), 9–14 (2000). doi: 10.1145/362883.362894

[28] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., Wiener, J.: Graph structure in the Web. In: Proceedings of the 9th international World Wide Web conference on Computer Networks: The International Journal of Computer and Telecommunications Networking, pp. 309–320 (2000). doi: 10.1016/S1389-1286(00)00083-9

[29] Broder, A.Z., Glassman, S.C., Manasse, M.S., Zweig, G.: Syntactic clustering of the Web. Computer Networks & ISDN Systems **29**(8-13), 1157–1166 (1997). doi: 10.1016/S0169-7552(97)00031-7

[30] Burner, M.: Crawling towards eternity: Building an archive of the World Wide Web. Web Techniques Mag. **2**(5) (1997)

[31] Cantrell, A.: Data backup no big deal to many, until... CNNMoney.com (2006). `http://money.cnn.com/2006/06/07/technology/data_loss/index.htm`

[32] Cathro, W., Webb, C., Whiting, J.: Archiving the Web: The PANDORA Archive at the National Library of Australia. In: Proceedings of the Preserving the Present for the Future Web Archiving Conference (2001)

[33] Chakrabarti, S., Dom, B., Kumar, R.S., Raghavan, P., Rajagopalan, S., Tomkins, A., Kleinberg, J.M., Gibson, D.: Hypersearching the Web. Scientific Am. **280**(6), 54–60 (1999)

[34] Cheney, M., Perry, M.: A comparison of the size of the Yahoo! and Google indices (2005). `http://vburton.ncsa.uiuc.edu/indexsize.html`

[35] Cho, J., Garcia-Molina, H.: The evolution of the web and implications for an incremental crawler. In: VLDB '00: Proceedings of the 26th International Conference on Very Large Data Bases, pp. 200–209 (2000)

[36] Cho, J., Garcia-Molina, H.: Effective page refresh policies for web crawlers. ACM Trans. on Database Systems (TODS) **28**(4), 390–426 (2003). doi: 10.1145/958942.958945

[37] Cho, J., Garcia-Molina, H., Haveliwala, T., Lam, W., Paepcke, A., Raghavan, S., Wesley, G.: Stanford WebBase components and applications. ACM Trans. on Internet Technol. (TOIT) **6**(2), 153–186 (2006). doi: 10.1145/1149121.1149124

[38] Cho, J., Garcia-Molina, H., Page, L.: Efficient crawling through url ordering. Computer Networks ISDN Systems **30**(1-7), 161–172 (1998)

[39] Cho, J., Roy, S.: Impact of search engines on page popularity. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web, pp. 20–29 (2004). doi: 10.1145/988672.988676

[40] Clausen, L.: Concerning Etags and datestamps. In: IWAW '04: Proceedings of the 4th International Web Archiving Workshop (2004)

[41] Clinton, D.: Beyond the SOAP Search API (2006). `http://google-code-updates.blogspot.com/2006/12/beyond-soap-search-api.html`

[42] Cooper, B., Crespo, A., Garcia-Molina, H.: Implementing a reliable digital object archive. In: ECDL '00: Proceedings of the 4th European Conference on Research and Advanced Technology for Digital Libraries, pp. 128–143 (2000)

[43] Cooper, B.F., Garcia-Molina, H.: Infomonitor: Unobtrusively archiving a World Wide Web server. Int. J. on Digital Libraries **5**(2), 106–119 (2005)

[44] Cox, L.P., Murray, C.D., Noble, B.D.: Pastiche: Making backup cheap and easy. SIGOPS Operating Systems Review **36**(SI), 285–298 (2002). doi: 10.1145/844128.844155

[45] Cutts, M.: GoogleGuy's posts (2005). `http://www.webmasterworld.com/forum30/29720.htm`

[46] Cutts, M.: SEO advice: URL canonicalization (2006). http://www.mattcutts.com/blog/seo-advice-url-canonicalization

[47] Dalal, Z., Dash, S., Dave, P., Francisco-Revilla, L., Furuta, R., Karadkar, U., Shipman, F.: Managing distributed collections: Evaluating web page changes, movement, and replacement. In: JCDL '04: Proceedings of the 4th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 160–168 (2004). doi: 10.1145/996350.996387

[48] Day, M.: Collecting and preserving the World Wide Web (2003). http://library.wellcome.ac.uk/assets/WTL039229.pdf

[49] Day, M.: Preserving the fabric of our lives: A survey of web preservation initiatives. Research Advanced Technol. Digital Libraries pp. 461–472 (2003). doi: 10.1007/b11967

[50] Douglis, F., Feldmann, A., Krishnamurthy, B.: Rate of change and other metrics: a live study of the World Wide Web. In: Proceedings of the USENIX Symposium on Internet Technologies and Systems, pp. 147–158 (1997)

[51] Dyreson, C.E.: Towards a temporal World-Wide Web: A transaction-time server. In: ADC '01: Proceedings of the 12th Australasian conference on Database technologies, pp. 169–175 (2001)

[52] Dyreson, C.E., Lin, H., Wang, Y.: Managing versions of web documents in a transaction-time web server. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web, pp. 422–432 (2004). doi: 10.1145/988672.988730

[53] Eprints for digital repositories. http://www.eprints.org/

[54] Eysenbach, G., Trudel, M.: Going, going, still there: using the WebCite service to permanently archive cited web pages. J. Medical Internet Research **7**(5) (2005). doi: 10.2196/jmir.7.5.e60

[55] Feise, J.: An approach to persistence of web resources. In: HYPERTEXT '01: Proceedings of the 12th ACM conference on Hypertext and Hypermedia, pp. 215–216 (2001). doi: 10.1145/504216.504267

[56] Fetterly, D., Manasse, M., Najork, M.: Spam, damn spam, and statistics: using statistical analysis to locate spam web pages. In: WebDB '04: Proceedings of the 7th International Workshop on the Web and Databases, pp. 1–6 (2004). doi: 10.1145/1017074.1017077

[57] Fetterly, D., Manasse, M., Najork, M., Wiener, J.: A large-scale study of the evolution of web pages. In: WWW '03: Proceedings of the 12th International Conference on World Wide Web, pp. 669–678 (2003). doi: 10.1145/775152.775246

[58] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol – HTTP/1.1. RFC 2616 (Draft Standard) (1999). http://www.ietf.org/rfc/rfc2616.txt

[59] Fire destroys top research centre (2005). http://news.bbc.co.uk/2/hi/uk_news/england/hampshire/4390048.stm

[60] Florescu, D., Levy, A., Mendelzon, A.: Database techniques for the world-wide web: a survey. ACM SIGMOD Record **27**(3), 59–74 (1998). doi: 10.1145/290593.290605

[61] Furl. `http://www.furl.net/`

[62] Galt, J.: Google says: Toolbar PageRank is for entertainment purposes only (2004). `http://forums.searchenginewatch.com/showthread.php?t=3054`

[63] Giles, C.L., Bollacker, K.D., Lawrence, S.: Citeseer: an automatic citation indexing system. In: DL '98: Proceedings of the third ACM conference on Digital Libraries, pp. 89–98 (1998). doi: 10.1145/276675.276685

[64] Gladney, H.M.: Trustworthy 100-year digital objects: Evidence after every witness is dead. ACM Trans. on Inf. Systems **22**(3), 406–436 (2004). doi: 10.1145/1010614.1010617

[65] GNU Wget Version 1.10. `http://www.gnu.org/software/wget/wget.html`

[66] Google privacy center: Terms of service (2006). `http://www.google.com/accounts/TOS`

[67] Google Toolbar. `http://toolbar.google.com/`

[68] Google webmaster help center: Webmaster guidelines (2007). `http://www.google.com/support/webmasters/bin/answer.py?answer=35769`

[69] Gulli, A., Signorini, A.: The indexable web is more than 11.5 billion pages. In: WWW '05: Special interest tracks and posters of the 14th International Conference on World Wide Web, pp. 902–903 (2005). doi: 10.1145/1062745.1062789

[70] Gupta, V., Campbell, R.: Internet search engine freshness by web server help. In: SAINT '01: Proceedings of the 2001 Symposium on Applications and the Internet, p. 113 (2001)

[71] Hagedorn, K.: OAIster: A "no dead ends" OAI service provider. Library Hi Tech **12**(2), 170–181 (2003)

[72] Hallgrimsson, T., Bang, S.: Nordic Web Archive. In: Proceedings of the 3rd ECDL Workshop on Web Archives in conjunction with 7th European Conference on Research and Advanced Technologies in Digital Archives (2003)

[73] Hank, C., Choemprayong, S., Sheble, L.: Blogger perceptions on digital preservation. In: JCDL '07: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries, p. 477 (2007). doi: 10.1145/1255175.1255276

[74] Hanzo:Web. `http://www.hanzoweb.com/`

[75] Harrison, T.L.: Opal: In vivo based preservation framework for locating lost web pages. Master's thesis, Old Dominion University (2005)

[76] Harrison, T.L., Nelson, M.L.: Just-in-time recovery of missing web pages. In: HYPERTEXT '06: Proceedings of the 17th ACM conference on Hypertext and Hypermedia, pp. 145–156 (2006)

[77] Harter, S.P., Kim, H.J.: Electronic journals and scholarly communication: a citation and reference study. Inf. Research **2**(1) (1996)

[78] Heydon, A., Najork, M.: Mercator: A scalable, extensible web crawler. In: WWW '99: Proceeding of the 8th International Conference on World Wide Web, pp. 219–229 (1999). doi: 10.1023/A:1019213109274

[79] How do I keep my page from being cached in Yahoo! Search? `http://help.yahoo.com/help/us/ysearch/basics/basics-10.html`

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

[81] Internet Archive forums: Robots archive - noarchive meta tags (2005). `http://www.archive.org/iathreads/post-view.php?id=31555`

[82] Internet Archive Web Team: Around the world in 2 billion pages (2007). `http://wa.archive.org/aroundtheworld/`

[83] Internet Archive Web Team: Worldwide Wayback Machine updated: 25% larger (2007). `http://wa.archive.org/blog/2007/07/02/worldwide-wayback-machine-updated-25-larger/`

[84] Jantz, R., Giarlo, M.J.: Digital preservation: Architecture and technology for trusted digital repositories. D-Lib Mag. **11**(6) (2005). doi: 10.1045/june2005-jantz

[85] Kahle, B.: Preserving the Internet. Scientific Am. **273**(3), 82–83 (1997)

[86] Karnin, E.D., Greene, J.W., Hellman, M.E.: On secret sharing systems. IEEE Trans. on Inf. Theory **29**(1), 35–41 (1983)

[87] Katzenbeisser, S., Petitcolas, F.A. (eds.): Information Hiding Techniques for Steganography and Digital Watermarking. Artech House, Inc., Norwood, MA, USA (2000)

[88] Koehler, W.: An analysis of web page and web site constancy and permanence. J. Am. Soc. Inf. Sci. **50**(2), 162–180 (1999). doi: 10.1002/(SICI)1097-4571(1999)50:2⟨162::AID-ASI7⟩3.3.CO;2-2

[89] Koehler, W.: Web page change and persistence — a four-year longitudinal study. J. Am. Soc. Inf. Sci. Technol. **53**(2), 162–171 (2002). doi: 10.1002/asi.10018

[90] Koehler, W.: A longitudinal study of web pages continued: A consideration of document persistence. Inf. Research **9**(2) (2004)

[91] Koster, M.: A standard for robot exclusion (1994). `http://www.robotstxt.org/wc/norobots.html`

[92] Lamolinara, G.: Library of Congress announces awards of $13.9 million to begin building a network of partners for digital preservation (2004). `http://www.digitalpreservation.gov/index.php?nav=4&subnav=2`

[93] Lannom, L.: Handle system overview. In: 66th IFLA Council and General Conference (2000). http://www.ifla.org/IV/ifla66/papers/032-82e.htm

[94] Lavoie, B., Nielsen, H.F.: Web Characterization Terminology and Definition Sheet, W3C Working Draft (1999). http://www.w3.org/1999/05/WCA-terms/

[95] Lawrence, S., Giles, C.L.: Searching the world-wide web. Sci. **280**(4), 98–100 (1998)

[96] Lawrence, S., Pennock, D.M., Flake, G.W., Krovetz, R., Coetzee, F.M., Glover, E., Nielsen, F.A., Kruger, A., Giles, C.L.: Persistence of web references in scientific research. Computer **34**(2), 26–31 (2001)

[97] Lee, S.H., Kim, S.J., Hong, S.H.: On URL normalization. In: ICCSA '05: Proceedings of the International Conference on Computational Science and its Applications, pp. 1076–1085 (2005). doi: 10.1007/11424826_115

[98] Lewandowski, D., Wahlig, H., Meyer-Beautor, G.: The freshness of Web search engine databases. J. Inf. Sci. **32**(2), 131–148 (2006)

[99] Liddle, S.W., Embley, D.W., Scott, D.T., Yau, S.H.: Extracting data behind web forms. In: Workshop on Conceptual Modeling Approaches for e-Business, pp. 402–413 (2002)

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

[101] Liu, X., Maly, K., Zubair, M., Nelson, M.L.: DP9: an OAI gateway service for web crawlers. In: JCDL '02: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 283–284 (2002). doi: 10.1145/544220.544284

[102] Mao, Y., Zhu, Z., Shi, W.: Peer-to-peer web caching: hype or reality? In: ICPADS 2004: Proceedings of the 10th International Conference on Parallel and Distributed Systems, pp. 171–178 (2004)

[103] Markwell, J., Brooks, D.W.: Broken links: the ephemeral nature of educational WWW hyperlinks. J. Sci. Education Technol. **11**(2), 105–108 (2002)

[104] Marshall, C., Bly, S., Brun-Cottan, F.: The long term fate of our personal digital belongings: Toward a service model for personal archives. In: Proceedings of IS&T Archiving 2006, pp. 25–30 (2006)

[105] Marshall, C., McCown, F., Nelson, M.L.: Evaluating personal archiving strategies for Internet-based information. In: Proceedings of IS&T Archiving 2007, pp. 151–156 (2007)

[106] McCown, F.: Google is sorry (2006). http://frankmccown.blogspot.com/2006/01/google-is-sorry.html

[107] McCown, F.: Warrick reconstructs jaysromanhistory.com (2006). http://frankmccown.blogspot.com/2006/04/warrick-reconstructs.html

[108] McCown, F.: Yahoo - error 999 (2006). `http://frankmccown.blogspot.com/2006/06/yahoo-error-999.html`

[109] McCown, F.: Windows Live Search development forum: Image search with 'site:' operator (2007). `http://forums.microsoft.com/MSDN/ShowPost.aspx?PostID=1799762&SiteID=1`

[110] McCown, F., Benjelloun, A., Nelson, M.L.: Brass: A queueing manager for Warrick. In: IWAW '07: Proceedings of the 7th International Web Archiving Workshop (2007)

[111] McCown, F., Chan, S., Nelson, M.L., Bollen, J.: The availability and persistence of web references in D-Lib Magazine. In: IWAW '05: Proceedings of the 5th International Web Archiving Workshop (2005)

[112] McCown, F., Diawara, N., Nelson, M.L.: Factors affecting website reconstruction from the web infrastructure. In: JCDL '07: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 39–48 (2007). doi: 10.1145/1255175.1255182

[113] McCown, F., Liu, X., Nelson, M.L., Zubair, M.: Search engine coverage of the OAI-PMH corpus. IEEE Internet Comput. **10**(2), 66–73 (2006). doi: 10.1109/MIC.2006.41

[114] McCown, F., Marshall, C.C., Nelson, M.L.: Why websites are lost (and how they're sometimes found). Communications ACM (2007). Accepted for publication

[115] McCown, F., Nelson, M.L.: Evaluation of crawling policies for a web-repository crawler. In: HYPERTEXT '06: Proceedings of the 17th ACM conference on Hypertext and Hypermedia, pp. 145–156 (2006). doi: 10.1145/1149941.1149972

[116] McCown, F., Nelson, M.L.: Agreeing to disagree: Search engines and their public interfaces. In: JCDL '07: Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 309–318 (2007). doi: 10.1145/1255175.1255237

[117] McCown, F., Nelson, M.L.: Characterization of search engine caches. In: Proceedings of IS&T Archiving 2007, pp. 48–52 (2007)

[118] McCown, F., Nelson, M.L.: Search engines and their public interfaces: Which APIs are the most synchronized? In: WWW '07: Proceedings of the 16th International World Wide Web Conference, pp. 1197–1198 (2007). doi: 10.1145/1255175.1255237

[119] McCown, F., Smith, J.A., Nelson, M.L., Bollen, J.: Lazy preservation: Reconstructing websites by crawling the crawlers. In: WIDM '06: Proceedings from the 8th ACM International Workshop on Web Information and Data Management, pp. 67–74 (2006). doi: 10.1145/1183550.1183564

[120] Mohr, G., Kimpton, M., Stack, M., Ranitovic, I.: An introduction to Heritrix, an archival quality web crawler. In: IWAW '04: Proceedings of the 4th International Web Archiving Workshop (2004)

[121] Morcos, F., Chantem, T., Little, P., Gasiba, T., Thain, D.: iDIBS: An improved distributed backup system. In: ICPADS '06: Proceedings of the 12th International Conference on Parallel and Distributed Systems, pp. 58–67 (2006). doi: 10.1109/ICPADS.2006.52

[122] MSN site owner help. `http://search.msn.com/docs/siteowner.aspx?t=SEARCH_WEBMASTER_REF_RestrictAccessToSite.htm`

[123] MSN terms of service (2006). `http://tou.live.com/en-us/default.aspx`

[124] Najork, M., Wiener, J.L.: Breadth-first crawling yields high-quality pages. In: WWW '01: Proceedings of the 10th International Conference on World Wide Web, pp. 114–118 (2001). doi: 10.1145/371920.371965

[125] Nelson, M.L., Allen, B.D.: Object persistence and availability in digital libraries. D-Lib Mag. **8**(1) (2002). doi: 10.1045/january2002-nelson

[126] Nelson, M.L., Smith, J.A., Garcia del Campo, I., Van de Sompel, H., Liu, X.: Efficient, automatic web resource harvesting. In: WIDM '06: Proceedings from the 8th ACM International Workshop on Web Information and Data Management, pp. 43–50 (2006). doi: 10.1145/1183550.1183560

[127] Nelson, M.L., Van de Sompel, H., Liu, X., Harrison, T.L., McFarland, N.: mod_oai: An Apache module for metadata harvesting. In: ECDL '05: Proceedings of the 9th European Conference on Research and Advanced Technology for Digital Libraries, pp. 509–510 (2005)

[128] Ntoulas, A., Cho, J., Olston, C.: What's new on the Web? The evolution of the Web from a search engine perspective. In: WWW '04: Proceedings of the 13th International Conference on World Wide Web, pp. 1–12 (2004). doi: 10.1145/988672.988674

[129] Ntoulas, A., Najork, M., Manasse, M., Fetterly, D.: Detecting spam web pages through content analysis. In: WWW '06: Proceedings of the 15th International Conference on World Wide Web, pp. 83–92 (2006). doi: 10.1145/1135777.1135794

[130] Ntoulas, A., Zerfos, P., Cho, J.: Downloading textual hidden web content through keyword queries. In: JCDL '05: Proceedings of the 5th ACM/IEEE-CS Joint Conference on Digital Libraries, pp. 100–109 (2005). doi: 10.1145/1065385.1065407

[131] Olsen, S.: Developers dig in to Google's toolbox. CNET News.com (2002). `http://news.com.com/2100-1023-884546.html`

[132] Olsen, S.: Court backs thumbnail image linking. CNET News.com (2003). `http://news.com.com/2100-1025_3-1023629.html`

[133] Olsen, S.: Google cache raises copyright concerns. CNET News.com (2003). `http://news.com.com/2100-1038_3-1024234.html`

[134] O'Neill, E.T., Lavoie, B.F., Bennett, R.: Trends in the evolution of the public web. D-Lib Mag. **3**(4) (2003). doi: 10.1045/april2003-lavoie

[135] Pagefactor. `http://www.pagefactor.com/`

[136] Pant, G., Srinivasan, P., Menczer, F.: Crawling the Web. In: Web Dynamics: Adapting to Change in Content, Size, Topology and Use. Edited by M. Levene and A. Poulovassilis, pp. 153–178 (2004)

[137] Paskin, N.: E-citations: Actionable identifiers and scholarly referencing. Learned Publishing **13**(3), 159–168 (2002)

[138] Phelps, T.A., Wilensky, R.: Robust hyperlinks cost just five words each. Tech. Rep. UCB/CSD-00-1091, EECS Department, University of California, Berkeley (2000)

[139] Plank, J.S.: A tutorial on Reed-Solomon coding for fault-tolerance in RAID-like systems. Software: Practice Experience **27**(9), 995–1012 (1997). doi: 10.1002/(SICI)1097-024X(199709) 27:9⟨995::AID-SPE111⟩3.3.CO;2-Y

[140] Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. J. ACM **36**(2), 335–348 (1989). doi: 10.1145/62044.62050

[141] Raghavan, S., Garcia-Molina, H.: Crawling the hidden web. In: VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases, pp. 129–138 (2001)

[142] Rao, H.C., Chen, Y., Chen, M.: A proxy-based personal web archiving service. SIGOPS Operating Systems Review **35**(1), 61–72 (2001). doi: 10.1145/371455.371462

[143] Registry of Open Access Repositories (ROAR). `http://roar.eprints.org/`

[144] Reich, V., Rosenthal, D.S.: LOCKSS: A permanent web publishing and access system. D-Lib Mag. **7**(6) (2001). doi: 10.1045/june2001-reich

[145] Remove cached pages from Google. `http://www.google.com/support/webmasters/bin/answer.py?answer=35306`

[146] Rosenthal, D.S.H., Lipkis, T., Robertson, T.S., Morabito, S.: Transparent format migration of preserved web content. D-Lib Mag. **11**(1). doi: 10.1045/january2005-rosenthal

[147] Ross, A.: Internet Archive forums: Web forum posting (2004). `http://www.archive.org/iathreads/post-view.php?id=23121`

[148] Rothenberg, J.: Ensuring the longevity of digital documents. Scientific Am. **272**, 42–47 (1995)

[149] Rothenberg, J.: Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation. CLIR Report #77. Council on Library and Information Resources, Washington, DC, USA (1999)

[150] Rumsey, M.: Runaway train: Problems of permanence, accessibility, and stability in the use of web sources in law review citations. Law Library J. **94**(1), 27–39 (2002)

[151] Sellitto, C.: The impact of impermanent web-located citations: A study of 123 scholarly conference publications. J. Am. Soc. Inf. Sci. Technol. **56**(7), 695–703 (2005). doi: 10.1002/asi.v56:7

[152] Shafer, K., Weibel, S., Jul, E., Fausey, J.: Introduction to persistent uniform resource locators. In: INET '96: Proceedings of the sixth annual conference of the Internet Society (1996). `http://www.isoc.org/isoc/whatis/conferences/inet/96/proceedings/a4/a4_1.htm`

[153] Sherman, C.: Microsoft upgrades Live Search offerings. Search Engine Watch (2006). `http://searchenginewatch.com/showPage.html?page=3623401`

[154] Shivakumar, N., Garcia-Molina, H.: Finding near-replicas of documents and servers on the web. In: WebDB '98: Selected papers from the International Workshop on The World Wide Web and Databases, pp. 204–212 (1999)

[155] Sitemap Protocol. `http://www.sitemaps.org/protocol.php`

[156] Spinellis, D.: The decay and failures of web references. Communications ACM **46**(1), 71–77 (2003). doi: 10.1145/602421.602422

[157] Spurl.net. `http://www.spurl.net/`

[158] StayBoyStay. `http://www.stayboystay.com/`

[159] Swaney, D.S., McCown, F., Nelson, M.L.: Dynamic web file format transformations with Grace. In: IWAW '05: Proceedings of the 5th International Web Archiving Workshop (2005)

[160] Swartz, A.: arcget: Retrieve a site from the Internet Archive (2005). `http://www.aaronsw.com/2002/arcget/`

[161] Symons, J.: How the Google cache can save your a$$ (2005). `http://www.smartmoneydaily.com/Business/How-the-Google-Cache-can-Save-You.aspx`

[162] Tepper, M.: The rise of social software. netWorker **7**(3), 18–23 (2003). doi: 10.1145/940830.940831

[163] Thati, P., Chang, P., Agha, G.: Crawlets: Agents for high performance web search engines. In: MA 2001: Proceedings of the 5th International Conference on Mobile Agents, vol. 2240 (2001)

[164] Thelwall, M.: Methodologies for crawler based web surveys. Internet Research **12**(2), 124–138 (2002)

[165] Thelwall, M., Stuart, D.: Web crawling ethics revisited: Cost, privacy, and denial of service. J. Am. Soc. Inf. Sci. Technol. **57**(13), 1771–1779 (2006). doi: 10.1002/asi.v57:13

[166] Thelwall, M., Vaughan, L.: A fair history of the Web? Examining country balance in the Internet Archive. Library & Inf. Sci. Research **26**(2), 162–176 (2004)

[167] Thomas, S.E., Kroch, C.A.: Project Harvest: A report of the planning grant for the design of a subject-based electronic journal repository (2002). `http://diglib.org/preserve/cornellfinal.html`

[168] Traeger, A., Joukov, N., Sipek, J., Zadok, E.: Using free web storage for data backup. In: StorageSS '06: Proceedings of the second ACM workshop on Storage Security and Survivability, pp. 73–78 (2006)

[169] U-M expands access to hidden electronic resources with OAIster. University Michigan News Service (2004). `http://www.umich.edu/news/index.html?Releases/2004/Mar04/r031004`

[170] Vaughan, L., Thelwall, M.: Search engine coverage bias: Evidence and possible causes. Inf. Processing & Management **40**(4), 693–707 (2004)

[171] Waters, D., Garrett, J.: Preserving digital information. Tech. rep., Task Force on Archiving of Digital Information (1996). `http://www.rlg.org/ArchTF/`

[172] Google Webmaster Help Center: How do I submit my OAI-PMH path? `http://www.google.com/support/webmasters/bin/answer.py?hl=en&answer=34655`

[173] Weideman, M., Mgidana, M.: Website navigation architectures and their effect on website visibility: a literature survey. In: SAICSIT '04: Proceedings of the 2004 annual research conference of the South African institute of computer scientists and information technologists on IT research in developing countries, pp. 292–296 (2004)

[174] Weiss, R.: On the web, research work proves ephemeral: Electronic archivists are playing catch-up in trying to keep documents from landing in history's dustbin. Washington Post p. A08 (2003). `http://www.washingtonpost.com/ac2/wp-dyn/A8730-2003Nov23`

[175] Wren, J.D.: 404 not found: The stability and persistence of URLs published in MEDLINE. Bioinformatics **20**(5), 668–672 (2004). doi: 10.1093/bioinformatics/btg465

[176] Wu, B., Davison, B.: Cloaking and redirection: A preliminary study. In: AIRWeb '05: Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web (2005)

[177] Xiao, L., Zhang, X., Andrzejak, A., Chen, S.: Building a large and efficient hybrid peer-to-peer Internet caching system. IEEE Trans. on Knowledge Data Engineering **16**(6), 754–769 (2004)

[178] Xiao, L., Zhang, X., Xu, Z.: On reliable and scalable peer-to-peer web document sharing. In: IPDPS 2002: Proceedings of the International Parallel and Distributed Processing Symposium, pp. 23–30 (2002)

[179] Yahoo: Submit your site. `http://submit.search.yahoo.com/free/request`

[180] Young, J.: Extensible repository resource locators (Errols) for OAI identifiers. `http://www.oclc.org/research/projects/oairesolver/`

APPENDIX A

WARRICK COMMAND-LINE SWITCHES

This appendix lists all the command-line switches used by Warrick. Many of these were borrowed from Wget [65].

TABLE 28: Warrick command-line switches.

Switch	Description
Startup:	
-V, --version	Display the version of Warrick
-h, --help	Print this help
-ga, --google-api	Use Google API instead of screen scraping (requires API key)
-gk, --google-key=KEY	Specify the Google key to be used
-t, --terminate-file=FILE	Specify FILE to look for if wanting to terminate before reconstruction is complete
Queries:	
-ql, --query-limits=LIMITS	Specify the total number of queries that may be issued in a 24 hour period
-I, --initial-used-queries=Q	Specify the number of queries that have been previously used in the past 24 hours
-nl, --no-lister-queries	Do not issue lister queries
Logging:	
-o, --output-file=FILE	Log messages to FILE
-s, --summary-file=FILE	Path to summary file (default uses URL to name file)
-d, --debug	Print debugging information
-v, --verbose	Print verbose information (this is the default)
-nv, --no-verbose	Turn off verboseness
Download:	
-c, --complete-recovery	All resources found from lister queries are downloaded
-n, --number-download=N	Specify the number of items N to be downloaded before quitting
-nc, --no-clobber	Skip downloads that would download to existing files
-w, --wait=SECONDS	Wait 5 +-SECONDS (random) between retrievals
-ic, --ignore-case	Ignore the case of URLs
-D, --target-directory	Directory to download the files to
-Y, --proxy	Use a proxy server (uses environment variable HTTP_PROXY)
-r, --recursive	Specify recursive download
-k, --convert-links	Make links in downloaded HTML point to local files
-v, --view-local	Add .html extension to HTMLized Word, PDF, Excel, etc. files and make links in downloaded HTML point to local files
Resource accept/reject:	
-dr, --date-range=BEG:END	Begin and end dates (yyyy-mm-dd) or single year (yyyy) for resources in IA
-m, --most-recent	Select the most recent version of resource, not necessarily the canonical version

APPENDIX B

RECONSTRUCTED WEBSITES

This appendix lists the 309 websites that were used in the experiment from in Section 5.1, Chapter VII. The first 303 URLs were crawled and reconstructed the first week, and the final six URLs were added on subsequent weeks after the experiment had begun.

TABLE 29: Listing of 309 websites.

1. alaskarottweilerclub.tripod.com	41. nativenewsonline.org
2. areq.csq.qc.net	42. noleeo.com
3. around.lgs.free.fr	43. onepeoples.com
4. art.sdsu.edu	44. pacificcoastsoftball.com
5. bcsvaluations.com	45. pages.prodigy.net/robertmorgan
6. boyertile.com	46. people.freenet.de/cindytimmypage
7. capitolautoparts.com	47. promo-echecs.ifrance.com
8. cartsoftweb.com	48. royalrangers.ch
9. ci.pierre.sd.us	49. schmidtandbartelt.com
10. coders31.free.fr	50. shermanbuck.com
11. dshc43015.tripod.com	51. smackdabdesign.com/mednf
12. elbonetazo.iespana.es	52. spanish.correctauditing.org
13. espanol.geocities.com/ligatolima	53. spoilerman.free.fr
14. espot.ddl.net	54. sthafrica.lcmglobal.org
15. estradas.no.sapo.pt	55. stonemillmodels.com
16. fib.ma.cx	56. sundog.stsci.edu
17. filia.mazoku.org	57. talk-2me.com
18. guide.supereva.com/lingua_russa	58. tartans.byair.net
19. hjem.get2net.dk/ahome	59. timarre.50megs.com
20. hofmuseum.sinfree.net	60. torahacademy.us.nstempintl.com
21. home.earthlink.net/~d4s	61. users.actcom.co.il/~yeda
22. home.novoch.ru/~vlad	62. utenti.lycos.it/fratresvg
23. i.hsr.ch	63. www.24-7consulting.com
24. idp.bl.uk	64. www.2bone.com
25. jpimpressions.com	65. www.abyouthhockey.net
26. kageki.hankyu.co.jp/english	66. www.ac-s.co.uk
27. kambuworld.free.fr	67. www.acapulcorentacar.com
28. kaskader.keys.pl	68. www.agclinic.gr
29. kaybuena.com	69. www.agf-trier.de
30. komatsu-npocenter.or.jp	70. www.agls.uidaho.edu/ccc
31. laidmanproductions.com	71. www.al6xn.com
32. library.thinkquest.org/3903	72. www.albatrossgolf.se
33. lienrag.free.fr	73. www.alconix.com
34. marienhospital.de	74. www.alexneilmsp.net
35. mars.bw.qc.ca	75. www.alhabtoor-motors.com
36. meiseikan.com	76. www.alliancesolutions.co.uk
37. members.aol.com/garzarea	77. www.americanserviceco.com
38. members.tripod.com/~lvhs	78. www.amigorico.org
39. mm-world.gamesurf.tiscali.de/heroes3	79. www.amycarolwebb.com
40. myautobuyers.com	80. www.andreamagro.it

TABLE 30: Listing of 309 websites continued.

81. www.andrelczyk.pl	131. www.daum-eickhorn.de
82. www.angel-flight.org	132. www.davismclay.com
83. www.annangrove-p.schools.nsw.edu.au	133. www.deoestgloria.com
84. www.antique-chinese-furnitures.com	134. www.depechemode-mp3.de
85. www.aoce.com	135. www.derbyshiremason.org/glossop_henry_hall
86. www.apollocoms.co.uk	136. www.dewafelbakkers.com
87. www.applewave.co.jp	137. www.dghallortho.com
88. www.architektur-bauphysik.de	138. www.diedreifragezeichen.info
89. www.arts.adelaide.edu.au/historypolitics	139. www.djbarranch.com
90. www.audreystark.com	140. www.dumontbrothers.com
91. www.avrealty.com	141. www.duperemover.com
92. www.bacowka.piwniczna.iap.pl	142. www.duperon.com
93. www.balletnorth.com	143. www.e-signature.com
94. www.banderolieren.de	144. www.eastgoth.de
95. www.bankasia-bd.com	145. www.easybirthing.com
96. www.barnetconservatives.co.uk	146. www.edithsfloralshop.com
97. www.bearpaw.ab.ca	147. www.ekoparkpernat.org
98. www.bergsrockshop.com	148. www.elegantk.org
99. www.bjallen.com	149. www.elscingles.com
100. www.blessedjohnduckett.durham.sch.uk	150. www.elviras.dk
101. www.bluesource.at	151. www.emg-inge-werner.de
102. www.boatletteringshop.com	152. www.ep1.ruhr-uni-bochum.de
103. www.bowentherapytechnique.com	153. www.euroflexpad.com
104. www.bronx.go3.pl	154. www.f-center.net
105. www.brucebrownlee.com	155. www.farmerindia.com
106. www.call911.se	156. www.flytxt.com
107. www.calsakcolorants.com	157. www.foreclosuretrac.com
108. www.campistrouma.com	158. www.forestry.gov.gy
109. www.celtichorizon.com	159. www.forhiskids.org
110. www.ces-landtec.com	160. www.franzincarni.it
111. www.chichwys.com	161. www.freewebs.com/starvedofsense
112. www.chinon.co.jp/eng	162. www.frogislandbrewery.co.uk
113. www.chiponein.com	163. www.gaestehaus-siegfried.de
114. www.ci.barnegat.nj.us	164. www.genevacom.com
115. www.cigarettefilters.com	165. www.geocities.com/icebreakersteam
116. www.city.hamamatsu-szo.ed.jp/shinohara-e	166. www.gites-tarn.com
117. www.citycup.dk	167. www.gmg2005.com
118. www.climatecampaign.org	168. www.gotemba.ne.jp
119. www.cobobrothers.com	169. www.greensheetads.com
120. www.cofrentes.com	170. www.groovee.com.au
121. www.coll.mpg.de	171. www.groundround.com
122. www.connectem.uji.es	172. www.gzespace.com
123. www.contractmanager.biz	173. www.hausunterricht.org
124. www.cooperstownallstarvillage.com	174. www.hemtours.com
125. www.cramerairportparking.com	175. www.homeprorichmond.com
126. www.crawfordcountyil.com	176. www.hoosiervan.com
127. www.css-webdesign.co.uk	177. www.hymnia.dk
128. www.dartmoorbks.dabsol.co.uk	178. www.incoretec.com
129. www.dartmouth.edu/~riding	179. www.inspirationsdesigns.com.au
130. www.dataopen.com	180. www.ivs-freiburg.de

TABLE 31: Listing of 309 websites continued.

181. www.jflowers.com	231. www.panzer.punkt.pl
182. www.joerg-schmitz-online.de	232. www.parkklinik-meersburg.de
183. www.jostvandykeferry.com	233. www.perfect10wines.com
184. www.juttamagic-key.de	234. www.plastyk.kielce.pl
185. www.katterugclub.nl	235. www.playingweb.com
186. www.kikori.org	236. www.princetonhcs.org
187. www.langfordcanoe.com	237. www.pronetworkcenter.com
188. www.lascanitas.de	238. www.proxan.de
189. www.lazerxtx.com	239. www.rasalam.com
190. www.learnrugbylaws.com	240. www.readyquip.com
191. www.libertyhorsetrailer.com	241. www.realestatecopies.com
192. www.logicdata.com	242. www.revelateur.be
193. www.macropixel.com	243. www.ritmo.ch
194. www.madacademy.com.au	244. www.rls2000.com
195. www.manaties.com	245. www.rondjenuland.nl
196. www.manna-vandaag.nl	246. www.rpaltd.co.uk
197. www.mark-stein.com	247. www.schachtermine.de
198. www.markgoodge.iofm.net/ stevewilliamssoccer_v2	248. www.scottaircraft.com
199. www.matem.unam.mx/whapde	249. www.seark.net/~rays
200. www.mccoy.org	250. www.senaiairport.com
201. www.mcnaystreet.childrencentre.org	251. www.setex-germany.com
202. www.mcplumbing.com	252. www.shipping-cases-now.com
203. www.meade.k12.ky.us/B-Town	253. www.silnirazem.plocman.pl
204. www.mebb.de	254. www.silvabelle.com
205. www.mece.net	255. www.slouch.tv
206. www.mediares.fr/sto	256. www.smallmarco.com
207. www.medicalsociety.org	257. www.smgfan.com
208. www.meganet.net	258. www.softcanarias.com
209. www.middle-earth.ru	259. www.sorensen.com.au
210. www.minorisa.es/gastronomia	260. www.sozedde.com
211. www.mrbox.co.uk	261. www.spenceschool.org
212. www.nast.gr	262. www.stadtmuseum.de
213. www.naturephotographs.com	263. www.stiftung-drittes-millennium.com
214. www.nealaw.com	264. www.sto-pavucina.cz
215. www.neiltanner.com	265. www.strumien.pl
216. www.newartgallery.net	266. www.summitautomotivegroup.com
217. www.newportirish.com	267. www.sungraph.co.uk
218. www.nikkigrogan.com	268. www.swgreens.com
219. www.northcincy.org	269. www.swtimes.com
220. www.nwclydes.com	270. www.taomusic.tv
221. www.objectrelations.org	271. www.terracottawarriors.com
222. www.oceanopolis.com	272. www.thacherrealestate.com
223. www.oceanridgeflorida.com	273. www.thaipressasso.com
224. www.oleificiodellorto.it	274. www.thoratec.com
225. www.oln.org	275. www.thriftlodgepetawawa.com
226. www.oregonsignworks.com	276. www.tirolerhof.erpfendorf.com
227. www.otrhobbyist.com	277. www.torikyo.ed.jp/kawasaki-e
228. www.oursaviorshartland.org	278. www.torontocpr.com
229. www.palemoon.com/RobertPlant01	279. www.tqu.jp
230. www.pallolaw.com	280. www.trailerservice.be

TABLE 32: Listing of 309 websites continued.

281. www.travel-company.spb.ru	296. www.watda.org
282. www.trents.co.uk	297. www.weber.nl
283. www.triplemhighlandfarmllc.com	298. www.wertstoffzentrum-schwandorf.de
284. www.twiceasniceshop.nl	299. www.whirlywiryweb.com
285. www.twinklebulbs.com	300. www.williamstwp.org
286. www.uk.sage24.com	301. www.woodenclocks.co.uk
287. www.ultrafachschaft-biertechnologie.de	302. www4.tpgi.com.au/louise_a
288. www.ultrasoundadvice.co.uk	303. www5.ocn.ne.jp/~urkasuga
289. www.unitedskate.com	304. *users.tpg.com.au/louise_a
290. www.uv.es/cide	305. *www.angelflightmidatlantic.org
291. www.valvision.fr	306. *www.areq.csq.qc.net
292. www.veterinariomilano.it	307. *www.areq.qc.net
293. www.vetrotexeurope.com	308. *www.theresavilliers.co.uk
294. www.victoriahighlandgames.com	309. *www.thinkquest.org/library
295. www.vicyoungjr.com	

$*$ indicates URLs added after the beginning of the experiment.

VITA

Frank McCown
Department of Computer Science
Old Dominion University
Norfolk, VA 23529

CONTACT

Harding University
Box 10764
Searcy, AR, 72149-0764
501-279-4826
fmccown@harding.edu

EDUCATION

Ph.D. in Computer Science, Old Dominion University, 2007
M.S. in Computer Science, University of Arkansas at Little Rock, 2002
B.S. in Computer Science, Harding University, 1996

EMPLOYMENT

8/07 to Present	Assistant Professor of Computer Science at Harding University (Searcy, Arkansas)
8/97 to 8/07	Instructor of Computer Science at Harding University (Searcy, Arkansas)
6/96 to 8/97	Software Engineer for Lockheed Martin Astronautics (Denver, Colorado)
6/95 to 8/95	Software Engineer Intern for Auto-trol Technology (Denver, Colorado)

PUBLICATIONS AND PRESENTATIONS

A complete list is available at http://www.harding.edu/fmccown/mccown-CV.pdf

PROFESSIONAL SOCIETIES

Association for Computing Machinery (ACM)

Typeset using LaTeX.