

Web Crawler Requirements

David Kellogg

November 5, 2004

Abstract

Criteria for the next generation web crawler are listed. The goal of the proposed crawler is to re-create the look and feel of a website as it existed on the crawl date. The tool should be extensible to adapt to future changes in web standards. This includes ever-changing standards of Javascript, HTTP and embedded material. A set of unit tests are suggested as goals for potential crawlers. Suggestions for presentation of dynamic (usually CGI) material are made. Currently, no commercial or free crawler-mirroring tool found fits these difficult criteria.

1 Introduction

Earlier, twenty open source web crawlers were evaluated.¹ HTTrack and cURL were proposed as extensible crawling tools. Lessons learned from this study are presented here. It was learned that present crawlers parse Javascript and handle non-HTML documents poorly. The crawler's interpretation of this material must improve to create an accurate archive.

2 General Requirements

Requirements for a mirror and crawling tool include comprehensive downloading, an intuitive and extensible interface, security of the server and browser, presentation of dynamic elements, and accurate look and feel. Advanced features also are suggested.

¹https://diva.cdlib.org/projects/crawling_harvesting/recall_crawl/spider_eval.pdf

2.1 Saving

The goal in mirroring a web page is that the look and feel of the page are mirrored exactly, down to every image, link, and dynamic element. This is a difficult task, mostly due to poor Javascript parsing abilities.

2.1.1 Scope

The scope of a crawl depends on the presentation of original material. For example, the document behind an external link should not be followed for use in the present archive. External images are different. All images on the web page must be saved, regardless of server origin. HTTP header redirects must be followed. These are shown as “Location: ...” in the header. Links should be followed even for different servers on the same domain. This includes `quote.yahoo.com` and `mail.yahoo.com`, for example.

2.1.2 Difficult Links and Depth

Some links and redirection statements are written in Javascript. These should be followed as well. A predefined maximum link depth should be obeyed.

2.2 Interface

The crawler should use an intuitive graphical user interface. A command line interface is helpful for extensibility and batch services.

2.2.1 Command Line Interface

A command line interface is important because batch services can include a number of projects running at once or at different times without user intervention. The user may stop the crawl at any time. The crawler should restart crawling at the last site visited, and continue with all formerly-cached links. Some crawls of individual sites may be large and take several hours, so this function is essential. A *cron* system is needed to schedule crawls over weeks.

2.2.2 Caching Pages and Rights

To detect redundancy, the crawler should create a data digest of a page (a short, unique signature), then compare it to the original signature for

each successive visit. Redundant pages do not need to be saved, but rather symbolically linked or included through server-side scripts. Crawls should be nice, meaning time (usually seconds) must elapse between page fetches. The robots exclusion protocol² must be honored. Rights to archive the material are not presumed. Web site owners must permit the presentation of the archive to the outside world. While a speed limit on sites must be observed, downloads must occur quickly enough to avoid inaccurate links.

2.3 Niceness

The robots exclusion protocol must be obeyed. This requires downloading the *robots.txt* file before crawling the rest of the website. Though a crawl should occur on a single day, a repetition rate of twelve or fewer pages per minute is responsible. The phone number and e-mail address of the crawlmaster should be listed in the request field. The crawler should be given a unique user-agent name. A web page should be written to explain the crawler's purpose.

2.4 Dynamic Pages and Images

Rollover images presented and loaded through Javascript must be saved. Both the primary and rollover images are needed. The crawler must download Shockwave-Flash files and other content listed in **EMBED** tags. Care must be taken not to load the same file multiple times, such as files ending in a unique timestamp, such as "...?TS=12345".

2.5 Presentation

Presentation is the most important aspect of archiving web sites. All links should be accurate. Almost always, re-crawling an archive is needed to rewrite all links to be internal.

2.5.1 Accurate Links

Anachronisms and broken images can cause distrust of the accuracy of the content as being from a certain date. Unsaved external images should not be used after the crawl. All data needed for the mirrored site should reside

²<http://www.robotstxt.org/wc/exclusion.html>

on the archive server. All links meant for the former site should be rewritten to be local. CGI links need to be accessible through PHP, JSP or another server-side technology. Javascript on the client side can accomplish the same task, though with mixed results due to browser incompatibility.

One interesting anachronism is a Javascript code that displays the date above the weather conditions on a California county government page. The script is not controlled by a server, but lists the current date through the Javascript `Date` object, confusing the user.

2.5.2 Form-Generated Links

For a CGI or inline-script generated page, where `search.cgi` is the root page, `search.cgi?s=apple` should also deliver a page. Unusable filename characters, such as “?” and “&” should be mapped to readable ASCII strings. Long filenames must be digested before saving, since operating systems limit filename length. This is especially a problem with query strings, which can be longer than 1000 characters.

2.6 Advanced Features

There are many advanced features that may never be perfected. These include security and reading pre-compiled code. Javascript should be rewritten to avoid security threats. Similarly, original website writers can craft PHP to cause a serious security threat on the archive server. ActionScript in Shockwave-Flash should be parsed for links and dynamic content, such as sound and images. The archive server should warn users that external links may not be accurate.

3 Unit Tests

Several unit tests were proposed for future crawlers. These and other unit tests should be constructed prior to coding.³ A crawler is qualified when it successfully follows these links and downloads required files without creating an error. As an example in Table 1, the crawler HTTrack receives a +6 score out of ten possible.

³For an interesting take on unit tests, see Extreme Programming at <http://www.extremeprogramming.org/rules/unittests.html>

Unit Test	HTTrack
JS Link	Y
Infinite Loop Honeypot	Y
Commented link	N
onMouseOver img	Y
Img in .js	N
Simple Flash	Y
CSS	Y
Flash imgs	N
Malicious JS	N
Malicious PHP	Y
Total	+6

Table 1: Unit Tests

1. *JavaScript Link*.⁴ A Link is broken into two strings, echoed by Javascript.
2. *Infinite Loop Honeypot*.⁵ Linked pages are the same, except for a ‘*?ts=TIMESTAMP*’ query string.
3. *Commented Link*.⁶ Link is preceded by ‘*<!--*’ without a closing ‘*-->*’, which is rendered incorrectly by most browsers.
4. *onMouseOver Image*.⁷ Rollover image may not work.
5. **.js Dynamic Image*.⁸ Dynamically loaded image may not show on rollover.
6. *Simple Flash*.⁹ Monolithic Flash documents are the easiest to save, and require no externally images.
7. *CSS Files*.¹⁰ CSS files are necessary to render pages in a usable way.

⁴<http://srb.cdlib.org/recall/unittest/unit.1.php>

⁵<http://srb.cdlib.org/recall/unittest/unit.2.php>

⁶<http://srb.cdlib.org/recall/unittest/unit.3.php>

⁷<http://srb.cdlib.org/recall/unittest/unit.4.php>

⁸<http://srb.cdlib.org/recall/unittest/unit.5.php>

⁹<http://srb.cdlib.org/recall/unittest/unit.6.php>

¹⁰<http://srb.cdlib.org/recall/unittest/unit.7.php>

8. *Flash with Images*.¹¹ Spider must parse **.swf* documents and load proper images.
9. *Malicious Javascript*.¹² Javascript can grab form data from users (especially CDL staff) and send it to any server.
10. *Malicious PHP*.¹³ Smart web site developers can read CDLIB databases.

4 Open Source Tools

There are many open source tools that can help to create a modular design and shorten the development cycle. Open source spiders and Javascript interpreters exist. These can be combined with tools to rewrite links to create a small suite of tools for creating accurate archives of web sites.

5 Conclusion

A comprehensive web archiving tool meeting the requirements outlined in this document is unavailable. A very good one can be produced using open source software with minimal expense and time. A good tool should be developed and updated using a short development cycle. Modular design can enable flexibility and speed for updates to evolving web standards.

This document is written in L^AT_EX.

¹¹http://srb.cdlib.org/recall/unittest/unit_8.php

¹²http://srb.cdlib.org/recall/unittest/unit_9.php

¹³http://srb.cdlib.org/recall/unittest/unit_10.php