Evaluation of Open Source Spidering Technology

David Kellogg

June 14, 2004

Abstract

Twenty spidering programs were evaluated for ease of use, technical ability and extensibility. Some lacked relative link rewriting and image saving ability. None could break through to the deep web without extension. Two tools, cURL and HTTrack, can be used and extended to meet the needs of future spidering projects. cURL is a command line tool for fetching data over the internet through common and difficult obstacles. HTTrack shows promise as a command-line and GUI tool, due to its many options and ease of use. With several problems, Stanford's WebBase crawler showed that it was able to handle multiple crawls. This study aims to find mirroring tools that re-create sites accurately.

1 Introduction

Most spidering tools lacked good basic mirroring ability. Very few searched *.js* or *.css* files for links,[1] parsed those pages and followed what browsers and web users see as obvious links. Many did not download images,[2] but relied on live original sites to feed these images to the archive viewer. This is clearly unacceptable. Many evaluated open source tools propagated the same bugs.[3] Two of the twenty evaluated tools are worth a closer look.

[1]See Stanford WebBase at http://www-diglib.stanford.edu/~testbed/doc2/WebBase.

[2]See Websphynx at http://www-2.cs.cmu.edu/~rcm/websphinx/#download.

[3]Weblech, like several other Java-based mirroring tools, did not rewrite link or image tags, so many images appear broken. See Weblech at http://weblech.sourceforge.net

cURL and HTTrack are mirroring tools that can be extended to make good archive-quality crawlers. The characteristic that sets these two tools apart from the rest is the solid downloading abilities at their core. Both cURL and HTTrack can download encrypted documents. Both handle cookies. Wrapper code to manage spidering and rewrite links of mirrored sites is necessary to extend these tools for archiving use.

Objective unit tests show a crawler's ability to navigate and mirror a difficult site. Unit tests are listed in Section 7.

2 Basic Needs

2.1 Mirroring vs. spidering

Effective crawling for archive purposes requires both good mirroring[4] and spidering[5]. Complete mirroring of a complex website is very difficult, and entails the same logic (reading HTML, Javascript and Flash) that browsers use. Spidering and collecting links and text is simple by comparison. Accurate archiving imposes unique needs, which requires spidering to be comprehensive and mirroring highly accurate.

2.2 Scheduling and niceness

Robots should rotate through a set of links politely, without grabbing data faster than the server administrator intends. A schedule is required to return to a site on a periodic (daily or longer) basis.

2.3 Starting and stopping

A database aids in starting and stopping the spider and pruning links without losing its place in a search. Without a database the code may incorrectly delete frontier links.[6]

[4]In this context, a *mirror* tool makes a nearly perfect replica of a web site.

[5]Any tool that traverses the web and gathers links and information is a *spider*. Here, it means a tool that mainly gathers links and text.

[6]*Frontier* links are found but not yet searched.

2.4 Extensible code

A command line interface is helpful to extend code. Instantiating objects works well within the same coding language, but this method cannot be extended easily to multiple languages needed by the spider.

2.5 Potential Pitfalls

Code may not apply widely-used standards. Warning flags include non-port-80 connections and use of no standard protocols. Poorly documented code can lead to problems. Using proprietary code that cannot be extended locks CDL into the solutions of a single vendor.

3 Advanced Needs

3.1 Deep web

The deep web consists of pages not typically found in search engine databases. These include sites without external links to their pages, Javascript-obfuscated links, form-generated pages, Flash pages and password-protected pages.

3.2 Storage needs reduction

Using MD5 to create a fingerprint of page versions prevents redundant page saves. The crawler should not resave old pages.

3.3 XML

XML files, XMP-RPC and SOAP can assist coding in several languages. Both XML-RPC and SOAP send data between modules of the spidering program.

3.4 HTML interface

An HTML interface eases access to lists of searched sites and errors produced. Control of spider and viewing of data and mirrored sites can be done entirely in a browser.

3.5 External links warning

Archive visitors should be warned that some external links might no longer exist. Also, cross-links between *joinarnold.com* and *meetup.com*, for instance, might not rewrite themselves properly in the mirrored site.

3.6 Cross-site JS scripting

Cross-site scripting[7] occurs when Javascript attempts to find information about web users, such as passwords. CDL, the potential host of archived pages, is most vulnerable to this attack.

4 cURL

4.1 Benefits of cURL

cURL is fast, extensible, and fully featured as a web-downloading tool. It can use telnet, ftp and http protocols. It has build-in encryption for SSL connections. It fills in forms, providing passwords to websites if requested. It follows all redirects and collects cookies automatically. cURL has a command line interface. Hooks to cURL exist in PHP and Perl. A Perl mirror tool that leverages cURL to download web site files is presented in Table 1.

4.2 Drawbacks of cURL

cURL lacks spidering ability. By itself, it cannot mirror whole websites. A wrapper must extend cURL to spider sites and rotate through links. A Perl wrapper was written in this case.

5 HTTrack

5.1 Benefits of HTTrack

HTTrack employs the most professional interface of the evaluated crawlers. It actually has two interfaces, one GUI, one command line. It has many options and accurately downloads images and rewrites links. It has SSL

[7]http://httpd.apache.org/info/css-security/encoding_examples.html

capability. It spiders a prescribed depth into the site. It can spider several sites in series.

5.2 Drawbacks of HTTrack

HTTrack lacks the extensibility of cURL. Difficulties in the core code can lead to awkward workarounds. CDL, for instance, might not be able to build form guessing into HTTrack, which is required to reach much of the deep web.

6 Stanford WebBase

6.1 Benefits of WebBase

Stanford's WebBase mirrors sites with almost perfect recall of mirrored html. Simple sites with little dynamic content can be mirrored with ease. After relinking and grabbing extra files from the current site, *joinarnold.com* as mirrored by WebBase is available.[8] A raw version from WebBase without some anachronistic content from the live site also is available.[9]

6.2 Drawbacks of WebBase

WebBase does not rewrite links within pages. It does not download files required in an `EMBED` tag as used by Flash and other programs. It does not parse or even scan Javascript for links or images. WebBase may be extended only with great difficulty. The C++ code is not ANSI compliant, and cannot be compiled *with* a modern compiler.

7 Unit Tests

Unit tests were constructed prior to coding.[10] Ten unit tests of medium difficulty were constructed to test the best performing mirror tools. These

[8] http://srb.cdlib.org/recall/unittest/mirror/www.joinarnold.com

[9] http://srb.cdlib.org/recall/unittest/raw_mirror/www.joinarnold.com

[10] For an interesting take on unit tests, see Extreme Programming at http://www.extremeprogramming.org/rules/unittests.html

Unit Test	**cURL**	**HTTrack**	**Stanford**
JS Link	N	Y	N
Infinite Loop Honeypot	N	Y	?
Commented link	Y	N	?
onMouseOver img	Y	Y	N
Img in .js	N	N	N
Simple Flash	Y	Y	N
CSS	Y	Y	N
Flash imgs	N	N	N
Malicious JS	N	N	Y
Malicious PHP	Y	Y	N
Total	+5	+6	+1

Table 1: Unit Tests

were HTTrack[11], Stanford[12], and cURL[13] (see section 4). The tests follow.

1. *JavaScript Link.*[14] A Link is broken into two strings, echoed by Javascript.

2. *Infinite Loop Honeypot.*[15] Linked pages are the same, except for a '`?ts=`*`TIMESTAMP`*' query string.

3. *Commented Link.*[16] Link is preceded by '`<!--`' without a closing '`-->`', which is rendered incorrectly by most browsers.

4. *onMouseOver Image.*[17] Rollover image may not work.

5. **.js Dynamic Image.*[18] Dynamically loaded image may not show on rollover.

6. *Simple Flash.*[19] Monolithic Flash documents are the easiest to save, and require no externally images.

[11]http://www.httrack.com
[12]http://www-diglib.stanford.edu/~testbed/doc2/WebBase
[13]http://ucop200171.ucop.edu/urbana
[14]http://srb.cdlib.org/recall/unittest/unit_1.php
[15]http://srb.cdlib.org/recall/unittest/unit_2.php
[16]http://srb.cdlib.org/recall/unittest/unit_3.php
[17]http://srb.cdlib.org/recall/unittest/unit_4.php
[18]http://srb.cdlib.org/recall/unittest/unit_5.php
[19]http://srb.cdlib.org/recall/unittest/unit_6.php

7. *CSS Files.*[20] CSS files are necessary to render pages in a usable way.

8. *Flash with Images.*[21] Spider must parse **.swf* documents and load proper images.

9. *Malicious Javascript.*[22] Javascript can grab form data from users (especially CDL staff) and send it to any server.

10. *Malicious PHP.*[23] Smart web site developers can read CDLIB databases.

8 Conclusion

Two possible tracks lead from this study. With little coding, HTTrack can be extended to meet immediate mirroring needs. A Perl script can be used to find extra links to feed deep web links to HTTrack. Alternatively, given funding and time, CDL's programming team can extend cURL to meet many higher-order tasks to archive 100% site mirroring.

The goal for archiving purposes is more difficult than simple spidering or mirror tools. A nearly perfect replication is necessary to give future users the look and feel of the website. Either HTTrack or cURL can be extended to overcome their present limitations.

A Other Crawlers

Eighteen other crawler tools were investigated. These are shown in Table 2.

B Location

The location of this document is
`diva.cdlib.org/projects/crawling_harvesting/`

[20]http://srb.cdlib.org/recall/unittest/unit_7.php
[21]http://srb.cdlib.org/recall/unittest/unit_8.php
[22]http://srb.cdlib.org/recall/unittest/unit_9.php
[23]http://srb.cdlib.org/recall/unittest/unit_10.php

Other Crawlers		
Name	**Benefits**	**Drawbacks**
Heritrix	Planned fully-featured bulk crawler.	Not a finished tool. Versions 0.4 and 0.6 were evaluated.
WWW::Spyder	Command line interface. Extensible.	Limited capability.
Websphinx	Has a nice visual tool. saves all html	No images.
Weblech	Command line. Saves to text & images.	Internal links are not re-written.
JSpider	Based on Java book.	Did not save files.
spider.pl	Perl spider.	Failed on 3 systems.
Arale	Good. Posts to "action=" links.	ENV variables must be set.
Lucene	Indexes site	Difficult compile.
infocopter.com	One-line spider interesting.	Not usable.
javaworld.com	Lightweight. Multi-threaded	Rapid downloading of sites impolite.
LWP	Solid Perl libraries for web.	No complete spidering tools.
Panoptic	Uses HTTPS & MD5.	$(AUS)15k per annum.
indiana.edu	Uses Google API. Subject is seed.	Applet appears to be security hole.
hcat	Good HTTP 1.0 downloading tool.	Old, not HTTP 1.1 compliant.
wget	Good HTTP 1.0 tool.	Not HTTP 1.1 compliant.
SimpleSiteMap-Gen	Java generates site map.	Not a mirroring tool.
Nutch	Java search engine.	Difficult to compile.
Swish-e	Good indexing tool for post-spidering.	None.

Table 2: Other Crawlers

`recall_crawl/spider_eval.pdf`

The location of the notes concerning all twenty spidering tools is `diva.cdlib.org/projects/crawling_harvesting/` `recall_crawl/spider_eval_notes.txt`

C Revisions

This document was revised July 6, 2004.

This document is written in LaTeX.