The RBSE Spider - Balancing Effective Search Against Web Load[1]

David Eichmann

Repository Based Software Engineering Program
Research Institute for Computing and Information Systems
University of Houston – Clear Lake

Abstract

The design of a Web spider entails many things, including a concern for reasonable behavior, as well as more technical concerns. The RBSE Spider is a mechanism for exploring World Wide Web structure and indexing useful material thereby discovered. We relate our experience in constructing and operating this spider.

1 – Introduction

As the World Wide Web [3] increases in complexity and number of users, it will be increasingly difficult for users to find information. Recent statistics posted by Fletcher [6], McBryan [12] and others indicate that there are more than 100,000 artifacts now Web-accessible. Relying solely upon browsing of hyperlinks or hand-crafted indices to gain access to specific topics is intractable.

This paper describes our experience with constructing a spider as part of our work on the Repository Based Software Engineering (RBSE) project. Web spiders are programs that traverse the Web, acting in some manner upon the information thereby uncovered. The RBSE spider discovers Web structure, indexes the full-text of hypertext markup language (HTML) [1, 2] documents that are part of the discovered web and provides relevance feedback-based search of the index with HTML-formatted display of the results. The index is accessible at

`http://rbse.jsc.nasa.gov/eichmann/urlsearch.html`

2 – Background

As the repository development group moved increasingly large portions of our user interface into the Mosaic/WEB domain [4], it became increasingly obvious that one area of dramatic

1. This work supported in part by NASA cooperative agreement NCC-9-16, RICIS research activity number RB02.

potential for growth for the repository was in the discovery of resources from other Internet sites and integration of those resources into our production classification scheme. URLs provided the means to reference remote assets, and modifications to the repository layers to support URLs as well as local file references was straight-forward. What was needed was a means to discover ‘interesting’ assets.

A number of Web indexing projects have begun, typically falling into one of two groups, meta-indexes or spider-generated index pages. *Galaxy* [5] is an example of the meta-index class of system, where the index administrator manually constructs a web of documents containing pointers to Web resources. Web spiders (sometimes referred to as robots, walkers, wanderers or worms [11]) are programs that inspect Web documents and take some action upon them, usually adding them to a pool of searchable documents or constructing a graph from hyperlinks that it identifies within the document. RBSE’s *URLSearch* and *JumpStation* [6] at Stirling University are a good examples of this more automated class of index, where the spider indexes documents that it encounters. JumpStation indexes only the title and first level headers of documents. URLSearch indexes the entire document. There are now a number of spiders in existence, varying in sophistication and mission. Koster maintains a list of known robots and what is publicly known about them [9].

3 – Design and Implementation

Early experimentation with JumpStation yielded less than satisfactory results for domains where resource providers were known to exist on the Web (environmental and software engineering information). We decided that identification of high relevance documents on the web was going to require full-text indexing, using a search mechanism similar to that provided by WAIS. A web spider was created that walks the Web, exploring the graph of documents and retrieving the full text of the documents encountered. Our requirements included:

- separation of structure discovery and indexing, so that indices could be updated without needing to rediscover known structure;
- ability to restart the spider on a known subgraph, and not rediscover already known structure;
- clear identification of the spider as such, rather than masquerading as a human operator of a Web client (URLSearch runs from rbse.jsc.nasa.gov (192.88.42.10), requests “GET <path> RBSE-Spider/0.1”, and uses “RBSE-Spider/0.1” in the User-Agent field);
- low Web impact – retrieval should be limited to only HMTL documents; and
- limited effort in obtaining preliminary results, so that we could assess the spider’s utility.

The first requirement drove the creation of an architecture that consists of two parts: a spider and an indexer, each described in more detail below. The separation of concerns allows the spider to be a lightweight assessor of Web state, while still providing the value to the general community through the indexer's activity. The drawback to this architecture is that a given HTML document will be touched once by the spider and once by the indexer, which will run sometime after the spider. We have debated a few alternatives to this approach, but none has the flexibility and robustness of our current architecture.

The second requirement reinforced the need for the spider/indexer split, for as we built our first index, we discovered that coding a robust (read core-dump-free) spider was a challenge, given the broad interpretation that authors were giving the HTML definition. The third and fourth requirements derived from our preliminary sense of spider ethics – these issues are discussed in more detail in [8]. The last requirement is rather obvious – this was something that had potential pay-back to the central goals of our project, but its utility was as yet unproven.

3.1 – Spider

The spider is a actually a pair of programs. Spider itself is a program that creates and manipulates an Oracle database of the Web graph, traversing links having patterns of "*.html" or "http:*/". The restart mechanism mentioned earlier can be invoked in a number of modes:

- breadth first search from a given URL passed as an argument;
- limited depth first search from a given URL passed as an argument;
- breadth first search from already-known, unvisited URLs present in the database; and
- limited depth first search from already-known, unvisited URLs present in the database.

Spider commits transactions following every URL visited to ensure that a minimum of documents are revisited in the case of an interrupted scan.

The second program, mite, is actually a modified version of the *www* ASCII browser. Mite, when given a URL as a command line argument, retrieves the document into a local file and prints any URLs found in the document on standard output. Spider spawns mite, reads the list of URLs and stores URL source–target pairs in the primary database table.

Unsuccessful retrieval attempts and leaves are logged into a separate table to prevent revisiting. A third table, added more recently following the derivation of the robot exclusion protocol [10], is a permanent list of patterns to match against for suppression. This is effectively then, a limited-depth breadth-first traversal of only HTML portions of the Web. We err on the side of missing non-obvious HTML documents in order to avoid links we're not interested in.

3.2 – Indexer

The indexer consists of a script that retrieves HTML URLs out of the database (using the same patterns employed by spider) and feeds them to a modified *waisindex*, which first uses

Table 1: A Sequence of Subgraph Snapshots

Date	Time	Distinct Sources	Distinct Targets	Total Edges
2/19/94	1:00PM	0	0	0
2/20/94	9:50AM	13,082	24,421	103,417
	5:00PM	13,789	33,715	118,930
	9:30PM	14,490	37,981	128,541
2/21/94	8:30AM	16,690	48,341	162,226
	11:15AM	17,278	53,803	171,957
	12:15PM	17,617	62,397	182,880

mite to retrieve the document and then indexes it. The modifications to waisindex involve the acceptance of a URL as an argument, the subsequent invocation of mite, and recognition of <TITLE></TITLE> strings as the document's headline.

Retrieval support is provided by a front page and a cgi script driving a modified *waissearch*, altered to generate HTML output of the entire result headline list down to a command line specified cut-off point for the number of relevance feedback hits.

4 – Building a Subgraph of the Web

Preliminary results for a 4000 document subgraph of the Web run in early January and started at the RBSE home page yielded 50,000 edges to 24,000 distinct target URLs and a full-text index of 30 megabytes. JumpStation's index at the time comprised titles and first level headers for roughly 40,000 documents, implying that a complete full-text index of the Web was currently feasible, but given the current growth rate, such an index will become increasingly infeasible. As an indication of growth, consider that Gray [7] identified 130 Web servers in June 1993, 204 in September, 228 in October, 272 in November and 623 in December. March 19, 1994 saw 1265 sites on the web, and this likely does not include a great many sites operating without public visibility or connectivity. Web traffic has grown to place sixth in the ranking of packet traffic on the Internet.

Table 1 contains some statistics concerning construction of the spider's most recent graph (constructed in late February). The full-text index for this graph is 100 megabytes. Note that this is only a snapshot of a portion of the web - effectively a five level breadth-first probe from our home page. (It's not a complete probe because I ran out of tablespace in Oracle...) The size of the index is also skewed by the inclusion of a large archive of abstracts from the Journal of Geophysical Research stored at NASA/Goddard and a large NASA thesaurus of terms, both of which are relatively self-contained and made up of small documents.

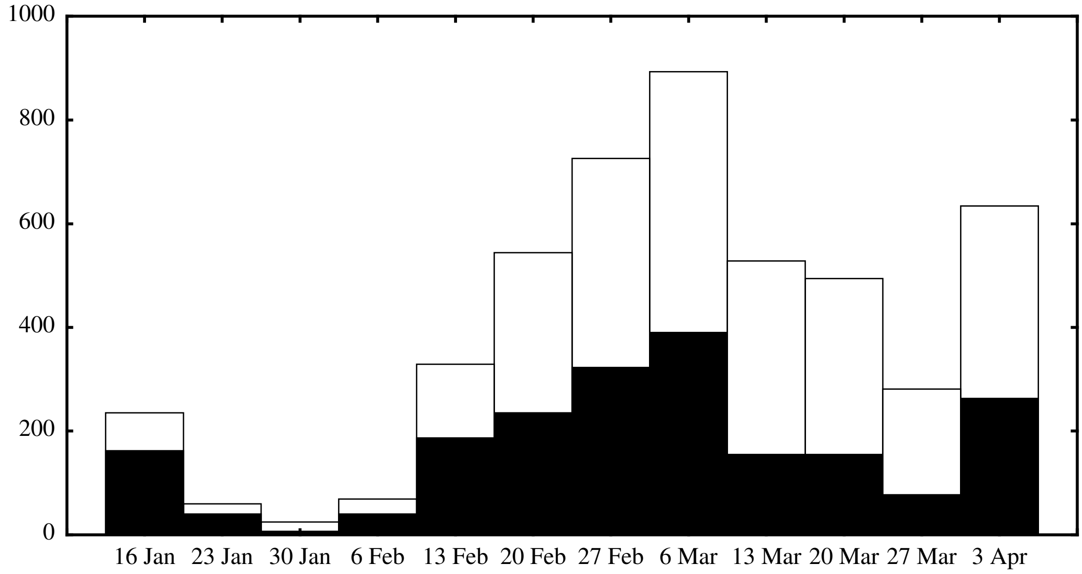

Figure 1: Number of Accesses by Week
(white: accesses, black: queries)

5 – Access Patterns Against URLSearch

Our original intention in creating the spider was for internal activities relating to resource discovery for inclusion into our repository [4]. As the Koster initiative developed, we developed a corresponding sense of needing to return something to the Web community in return for the intermittent load that we have placed upon their servers. At the same time, our experimental probes of the Web were relatively limited, and so we weren't prepared to announce the URLSearch facility *too* broadly. We struck a middle ground of including a link to the index on our project's home page. NASA (our sponsor) has subsequently developed an increasing attention to the Web, and a link to our index appears on the NASA home page (http://hypatia.gsfc.-nasa.gov/NASA_homepage.html), but still marked as experimental.

Figure 1 shows the number of accesses our server has received with this limited form of announcement. The quantity itself is not particularly remarkable, given the load on other publicly announced indices, but remember that we didn't start out with an intention to become a public index either. The numbers *are* interesting in that they demonstrate the effects of individuals probing paths of hyperlinks, and both coming back to those that are interesting and spreading the word about interesting links to others. An excellent example of this is our placement on the NASA home page, which we found out (to our surprise and pleasure) by browsing the Web based upon results of a search on URLSearch for project related words. Figure 2 shows the same data broken out by hour of the day. Note that even with the low advertising, our server is receiving a request every 7-12 minutes during peak hours (9AM to 5PM, CST).

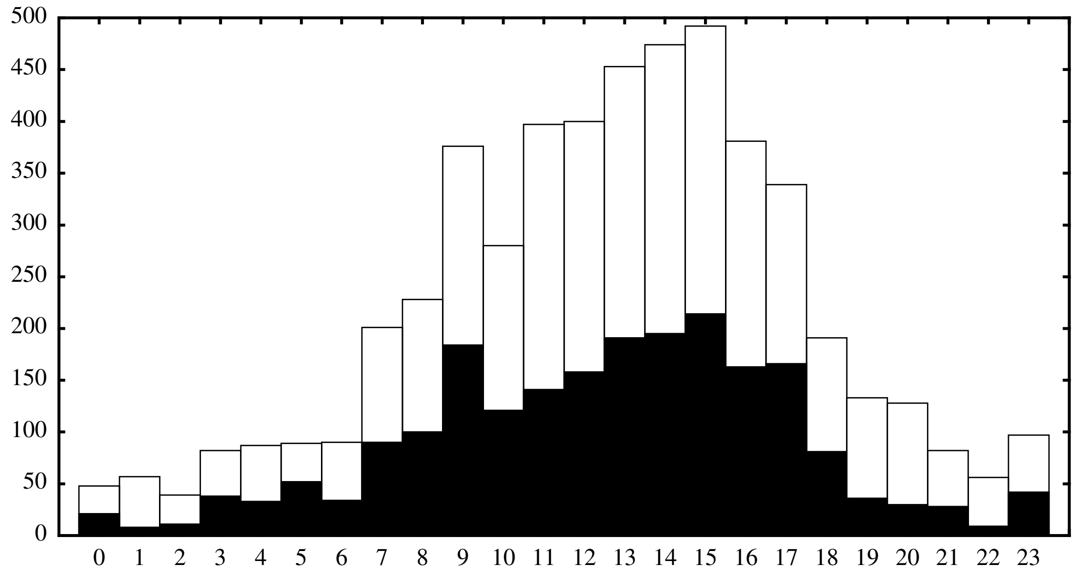

Figure 2: Number of Accesses by Hour of Day (white: accesses, black: queries)

6 – A Rationale for the Existence of Spiders

Koster has raised reasonable concerns about Web robots generating significant demands upon Web servers [8]. Authors of such systems must exercise reasonable restraint in the scope and frequency of execution. There are an increasing number of servers that are generating HTML dynamically from large databases of information, and it is not reasonable to index this type of information.

An important consideration, however, is the value added in what spiders can provide in avoided network traffic. Good indices can provide direct and focused interaction with the web, rather than frantic traversals in search of potential information. No one would suggest that the best mechanism for establishing the location of a destination city on a map is to start at your current location and follow the roads leading from it to see if they lead to the desired goal. Much of the Web is currently accessed in this manner, however, or though the use of hand constructed lists of interesting links - akin to the mariner's log of old.

The true value in spiders will be the construction of up-to-date virtual neighborhoods of documents. Our spider constructs one such form of local neighborhood, generated by simple connectivity. Other forms include host domain (e.g., *.nasa.gov) or semantic similarity (e.g., relevance greater than 0.5 on 'software reuse' and 'domain engineering'). Note however, that both of these require awareness of information concerning the contents of the Web, existence of a URL in the first case, and of its contents in the second. Any virtual neighborhood built upon anything more sophisticated than simple connectivity and involving a single constructing agent requires discovery of a more general neighborhood that includes that target neighborhood, which leads back to the simplistic discovery techniques of spider and mite.

7 – Conclusions and Future Work

Building a first cut implementation of a spider is rather easy. Building a spider that is sufficiently robust and well-behaved as to operate as a good Web denizen requires more careful design. We have described our experience with building what we hope is one such system.

The growth of the Web is beginning to outstrip the ability of any but a single-purpose server to construct a complete full-text index. RBSE development plans for the spider therefore involve domain-specific Web index construction, using domain profiles to identify documents and indexing their local neighborhoods. RBSE will be using spider results to provide two important functions for on-line information services. The first is the identification of new sources of information for inclusion in the general interfaces for the repository. The second is support for customized notification of new resources to clients based upon profiles placed on file.

References

[1] Andreessen, M., "A Beginner's Guide to HTML," National Center for Supercomputer Applications, Univ. of Illinois, http://www.ncsa.uiuc.edu/demoweb/html-primer.html.

[2] Berners-Lee, T. (ed.), "HyperText Mark-up Language," CERN, http://info.cern.ch/hypertext/WWW/MarkUp/MarkUp.html.

[3] Berners-Lee, T., "The World Wide Web Initiative," CERN, http://info.cern.ch/hypertext/WWW/TheProject.html.

[4] Eichmann, D., T. McGregor and D. Danley, "Integrating Structured Databases Into the Web: The MORE System," *First International Conference on the World Wide Web,* CERN, Geneva, Switzerland, May 25-27, 1994. Also available as http://rbse.jsc.nasa.gov/eichmann/www94/MORE/MORE.html.

[5] EIT, Inc., "EINet Galaxy," http://www.eit.com/, January 1994.

[6] Fletcher, J., "JumpStation Front Page," http://www.stir.ac.uk/jf1bin/js, January 1994.

[7] Gray, M., "Growth of the World Wide Web," http://www.mit.edu:8001/aft/sipb/user/mkgray/ht/web-growth.html, Dec. 1993.

[8] Koster, M., "Guidelines for Robot Writers," Nexor Corp., http://web.nexor.co.uk/mak/doc/robots/guidelines.html.

[9] Koster, M., "List of Robots," Nexor Corp., http://web.nexor.co.uk/mak/doc/robots/active.html.

[10] Koster, M., "A Proposed Standard for Robot Exclusion," Nexor Corp., http://web.nexor.co.uk/mak/doc/robots/norobots.html.

[11] Koster, M., “World Wide Web Wanderers, Spiders and Robots,” Nexor Corp., http://web.nexor.co.uk/mak/doc/robots/robots.html.

[12] McBryan, O., “WWWW - the World Wide Web Worm,” U. Colorado, http://www.cs.colorado.edu/home/mcbryan/WWWW.html.