Appendix A

Practical Web Crawling Issues

When we tested our implementation, which is described in Chapter **??**, we found that there were several problems of Web crawling that did not become evident until a large crawl was executed. Our experiences arise from several crawls of the Greek, Spanish and Chilean Web carried out during the this thesis.

We are interested in documenting these problems for two reasons:

- To help other crawler designers, because most of the problems we found are related to the characteristics of the Web, independent of the Web crawler architecture chosen.
- To encourage Web application developers to check their software and configurations for compliance to standards, as this can improve their visibility on search engine's results and attract more traffic to their Web sites.

The rest of this chapter is organized as follows: Section A.1 deals with network problems in general. Section A.2 deals with more specific problems with massive DNS resolving. Section A.3 presents the problems of dealing with wrong implementations of HTTP. Regarding the server-side, Section A.4 deals with bad HTML coding, Section A.5 with problems in the contents of the pages, and Section A.6 with difficulties arising from the programming logic of some Web sites.

A.1 Networking in general

An estimation for the cost of an entire crawl of the World Wide Web is about US $1.5 Million [CCHM04], considering just the network bandwidth necessary to download the pages, so it is very important to use the network resources efficiently to maximize the crawler throughput and avoid wasting the allocated bandwidth.

A.1.1 Variable quality of service

One of the most challenging aspects of Web crawling is how to download pages from multiple sources in a stream of data that is as uniform as possible, considering that Web server response times are very variable.

Web server up-time cannot be taken for granted, and it is usual to find Web servers that are down for a long time, even days or weeks, and re-appear later. This is why sometimes "dead pages" are called "comatose pages" [Koe04]. If the Web crawler aims for a comprehensive coverage of the Web, it should consider that some of the pages which are not available now, could become available in the future.

Recommendation: the crawler should re-try each Web page a number of times if the page is down; the interval should be several hours. We used 12 hours as the default).

A.1.2 Web server administrators concerns

Web crawlers prompted suspicion from Web site administrators when they first appeared, mostly because of concerns about bandwidth usage and security, and some of those concerns are still in place today. In our experience, repeated access to a Web page can trigger some alarms on the Web server, and complaints from its administrator.

We consider that the two most important guidelines given by Koster [Kos93] are:

- A crawler must identify itself, including an e-mail address for contact, or some Web site administrators will send complaints to the listed owner of the entire originating network segment.
- A crawler must wait between repeated accesses to the same Web site.

These guidelines are even more important if we consider that many host names point to the same IP, usually belonging to a Web hosting provider, and in general several Web sites are hosted by a few physical servers. Being unpolite with a Web site can result in being banned from all the Web sites hosted by the same ISP.

Recommendation: the crawler should avoid overloading Web sites, and it must provide an e-mail address in the `From` HTTP header, and/or a Web site address as a comment in the `User-Agent` HTTP header.

A.1.3 Inconsistent firewall configurations

Some Web servers are behind a firewall, and we have found firewall configurations that we did not expect. We detected cases when the `connect()` call succeeds, i.e. a TCP connection is established with port `80` of the Web server, then the `write()` call succeeds, but there is no answer from the Web server.

This appears to be a problem with data packets to port `80` being dropped, but connections accepted, which is not a consistent configuration. This caused some threads of the harvester to hang up indefinitely in one of our early versions.

Recommendation: all network operations should have a timeout. The crawler must be prepared, because at any point of the download operation it could stop receiving data.

A.2 Massive DNS resolving

A.2.1 Crashing your local DNS servers

We found that some of our local DNS servers crash under heavy loads, instead of just queuing or denying connections. From the Web crawler's point of view, a DNS failure of the local servers is a critical situation because, if after repeated attempts it cannot get an IP address for connecting, it has to assume the Web site does not exist, and if all DNS lookups are failing, this can make an entire crawl useless.

Recommendation: local DNS servers should be tested for their response to high work loads. The Web crawler should detect a condition in which, e.g.: 90% of DNS lookups failed during one cycle, and stop under this condition. The Web crawler also could avoid resolving more than a fixed number of domain names at the same time and with the same DNS server.

A.2.2 Temporary DNS failures

This is related to the quality of service of Web servers themselves, as for small organizations typically the Web server and the DNS server are both under the same administration and even in the same physical computer. A DNS failure (e.g.: a DNS server crash) is very likely to go unnoticed, because of the default caching policy: one week. People who visit the Web site often will not notice that something is wrong until several days have passed.

Recommendation: if high coverage is desired, at least one attempt to resolve a DNS record should be done one week after a DNS failure. However, it can also be argued that a Web site with DNS problems has a lower quality than other Web sites and should not be added to the collection: in our model, Web server response quality can be used as a component of the intrinsic quality of a Web page.

A.2.3 Malformed DNS records

DNS report [Per04] is a tool for analyzing DNS records. Its author reports that a significant fraction of DNS records present some problems, ranging from inconsistencies in the serial numbers to misspelling errors or malformed responses.

Recommendation: DNS resolvers should be tolerant to errors in DNS records, and try to retrieve the IP address for a host name even if other portions of the record seems malformed.

A.2.4 Wrong DNS records

Consider the scenario depicted in Figure A.1:

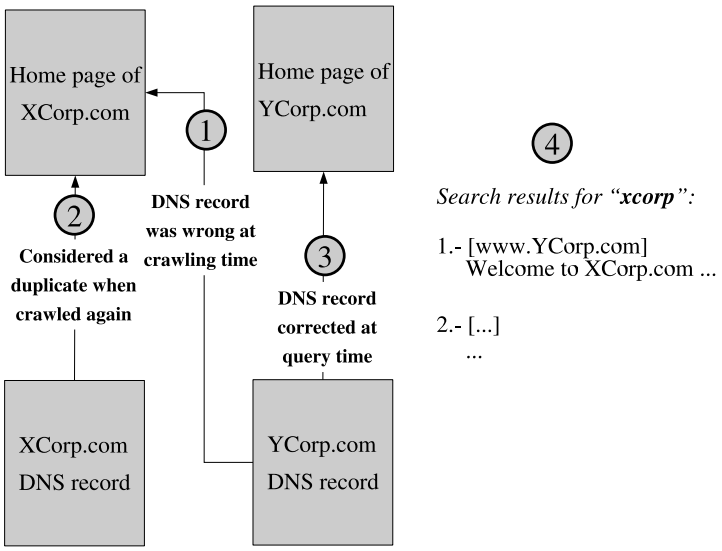

Figure A.1: A misconfiguration in the DNS record for "Ycorp.com" resulted in the wrong contents being assigned to its URL.

1. At crawling time, the DNS record for `Ycorp.com` pointed to the Website of `Xcorp.com`, so the contents of the later were indexed as if their URL was `Ycorp.com`. This DNS misconfiguration can be accidental, or malicious.

2. When the home page of `Xcorp.com` was downloaded, its contents were found to be a duplicate of `Ycorp.com`, so the pages of `Xcorp.com` were not downloaded again.

3. The wrong DNS record of `Ycorp.com` was fixed later.

4. In the search results, when users search for "Xcorp", they can be mistakenly redirected to the Web site of "Ycorp".

Recommendation: it is possible for Web site administrators to avoid these kind of problems by a careful configuration of virtual hosts. Any access to the IP address of the Web server that does not contain a known `Host` field in the request, should be redirected to the default virtual host, referencing the later by its canonical name.

A.2.5 Use of the "www" prefix

Due to the usage of the `www` prefix for the host part of the URLs, in most Websites both "www.example.com" and "example.com" names resolve to the same IP address, and have the same contents. Indeed, we have found that for many Web site administrators, this is the expected behavior, as some users do not type the full address when browsing.

However, if the Web site is built using some application that includes small changes in the pages (e.g.: the current date and time, or a poll question, or advertising, etc.), the Web crawler may not be able to detect that both Web sites are duplicates, and crawl the same contents twice.

Recommendation: we considered that `http://www.example.com/` and `http://example.com/` are the same URL.

A.3 HTTP implementations

A.3.1 Accept headers not honored

In some cases, it is impossible to tell the type of a file just by looking at its URL. Some URLs have no extensions, and some URL have extensions that are ambiguous, e.g.: links to files ending in `.exe` could be either links to dynamically generated HTML pages in the server side, or links to programs that should be downloaded.

A user agent, such as a Web browser or a Web crawler, can have limited capabilities and only be able to handle some data types. If it cannot handle a file (e.g.: an image), then it should not download it. For instance, a Web crawler searching only for plain text and HTML pages, should issue a request of the form:

```
GET /page.html HTTP/1.1
Accept: text/plain, text/html
...
```

This indicates that the Web crawler can only handle plain text or HTML documents. According to the HTTP specification, "the server SHOULD send a 406 (not acceptable) response code" [FGM+99] when a valid object of the desired type is not present at the given URL.

Several Web browsers simply issue a header of the form `Accept: */*`, so some Web server implementations do not check the "accept" header at all. It has somehow lost relevance, and today a Web server can send a response with almost any data type.

A related concern is that some Web sites return a header indicating content-type HTML, but the information returned is a large binary file (such as a ZIP archive, etc.). The crawler can waste bandwidth downloading such a file.

Recommendation: the returned `Content-type` header should always be checked in the downloaded pages, as it might not be a data type that the Web crawler can handle. A download limit is necessary because potentially any file type can be returned by the Web server, even when it is indicating HTML content type.

A.3.2 Range errors

To ensure a good coverage of the Web, we must limit the amount of data that is downloaded from every Web server. This can be done by limiting both the maximum page size, and the number of Web pages that are downloaded from a single Web site.

We limit page size usualy to a default of 300-400 Kb per Web page. We consider that this should capture enough keywords to index each document. To inform the Web servers of the download limit, we use the HTTP `Range` header:

```
GET /page.html HTTP/1.1
Range: 0-400000
...
```

However, a few Web sites return a response code 416 (range error). We have found that these responses correspond to files that are smaller than the desired size. This is not correct, because the HTTP specification indicates that "if the [second] value is greater than or equal to the current length of the entity-body, last-byte-pos is taken to be equal to one less than the current length of the entity- body in bytes" [FGM+99].

Recommendation: in the case of range errors, a second attempt for download could be made without the `Range` header. In all cases, the Web server may ignore the range, so the Web crawler must be prepared to disconnect from the Web server, or discard part of the contents, if the server sends a long document.

A.3.3 Response lacking headers

We have found that most Web browsers are very tolerant to strange behavior from the Web servers. For instance, we have tested Opera and Internet Explorer against a "dummy" Web server that only sends the contents of the Web page requested, with *no status line and no headers*. Both browsers displayed the downloaded pages. The browser Mozilla shows an error message.

Some real Web sites exhibit the same misbehavior as our dummy Web servers, probably because of misconfigured software, or misconfigured firewalls. The Web crawler should be prepared to receive content without headers.

A related problem is that of imcomplete headers. We have found, for instance, responses indicating a redirection, but lacking the destination URL.

Recommendation: from the point of view of Web crawlers or other automated user agents, pages from Web sites that fail to comply with basic standards should be considered of lower quality, and consistently, should not be downloaded. We consider a lack of response headers a protocol error and we close the connection.

A.3.4 Found where you mean error

It is hard to build a Web site without internal broken links, and the message shown by Web servers when a page is not found, i.e.: when the Web server returns a 404 (not found) response, is considered by many Web site administrators as too annoying for users.

Indeed, the default message looses the context of the Web site, so the Web site administrators of some Web sites prefer to build error pages that maintain visual and navigational consistency with the rest of their Web sites.

The problem is that in many cases the response for a page that does not exists is just a normal redirect to a custom-built error page, without the response header signaling the error condition. Bar-Yosef *et al.* [**?**], refer to these error pages as "soft-404", and observe that about 29% of dead links point to them.

The indexing process could consider a redirect to a "soft-404" error page as a link between the URL in which the page was not found and the error page, and this can increase the score of the later.

Recommendation: Web site administrators should configure their Web servers in such a way that the error messages have the correct response codes signaling the error conditions. Servers can be tested by Web crawlers issuing a request for a known non-existent page (e.g.: an existent URL concatenated with a random string [**?**]) and checking the result code.

A.3.5 Wrong dates in headers

A significant fraction of computers are configured to a wrong date, wrong time, or wrong time zone. These configurations are sometimes done on purpose, e.g.: to extend the trial period of shareware software.

During Web crawling, we found that 17% of Web servers returned no last-modification data, dates in the future, or a date prior to the invention of the Web, as shown in Section **??** (page **??**). These wrong dates affect the `Last-Modified` field in the Web server response, which in these cases cannot be used for estimating freshness.

However, not all wrong dates should be discarded: if a Web server replies with a time stamp in the future, but just a few minutes or a few hours, we can consider that it is likely that the Web server clock is just skewed with respect to ours (e.g.: it has the wrong time zone, or it is wrongly set).

Recommendation: we consider that a last modification date for a Web page older than the year 1993 is

wrong and should be ignored. For dates in the future, we consider that up to 24 hours can be considered just a small problem, so those time stamps are changed into the current date. If the date is more than 24 hours ahead it is ignored. This is depicted in Figure A.2.

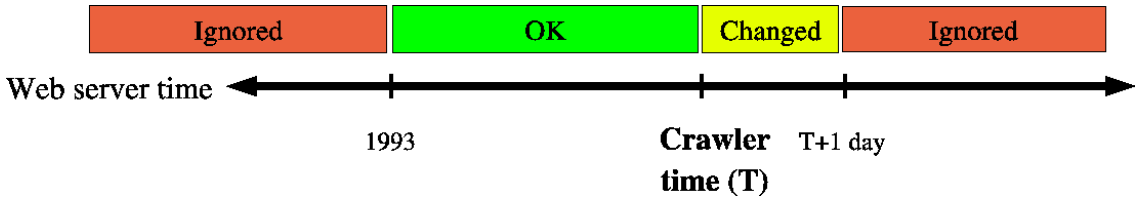

Figure A.2: Diagram showing how we deal with last-modification dates in the responses.

A.4 HTML coding

A.4.1 Malformed markup

HTML coding, when done by hand, tends to be syntactically very relaxed. Most HTML coders only check if the page can be seen in their browsers, without further checking for compliance. We have found several errors in HTML coding, and we have adapted our parser accordingly. These errors include:

- Mixing single quotes, double quotes, and no quotes in attributes, e.g.: `<IMG ALT="This is a photo" SRC='photo.jpg' border=1>`.
- Mixing empty tags in HTML form (such as `<BR>`) and in XHTML form (such as `<BR/>`).
- Unbalanced tags, e.g.: `<SMALL><B>...</SMALL></B>`.
- Mixed case in tags and attributes, e.g.: `<IMG src="...">`. For HTML, the tags should be written in uppercase, and for XHTML, in lowercase.
- Unterminated strings, e.g.: `<IMG ALT="...>`. This can be very problematic, because it will cause a buffer overflow if the parser is not properly written. These unterminated or long strings can also appear in HTTP response codes.

Recommendation: as described in Section **??** (page **??**), we use an events-oriented parser for the HTML pages, as in many cases it is very difficult to map the Web page to a tree. For Web site administrators, the usage of a tool for cleaning markup such as “HTML tidy” [Rag04] is encouraged.

A.4.2 Physical over logical content representation

The search engine must build an index of the Web pages. This index may consider only the text, or it may also consider the HTML tags by, e.g.: assigning a higher weight to terms appearing in section headers.

However, HTML markup is usually oriented to the visual characteristics of the documents; consider this HTML fragment:

```
<div align="center"><font size="+1" color="red">Important facts</font></div>
<p>Read this ...</p>
```

The visual characteristics of the phrase “Important facts” are: larger font size, red color, aligned to the center of the page. These visual aspects indicate an important block of text, but they are not visible by most search engines.

Recommendation: the following markup should be preferred:

```
<style>
h1 {
font-size: larger;
color: red;
text-align: center
}
</style>
<h1>Important facts</h1>
<p>Read this ...</p>
```

This markup separates content from representation, and visually produces the same results. For improved maintainability, the style rules can be provided in a separate file.

A.5 Web content characteristics

A.5.1 Blogging, mailing lists, forums

Blogs, Web forums and mailing list archives are large repositories of information, comprised of many small postings by individual users. They can be a useful source of information when the topic is not covered somewhere else; typical examples are technical support messages, usually describing solutions to problems with very specific software or hardware configurations.

However, sometimes individual postings are not as valuable as other pages, as they are very short, or lack clarity or factual information. Also, there is a problem with the granularity of the data, i.e.: a single posting contains little information, but the complete conversation can be valuable.

There is no easy solution for this problem, but as Web sites archiving user discussions can have hundreds of thousands of pages, they make even more important the use of a good scheduling order to try to download important pages first.

A.5.2 Duplicate detection

The prevalence of mirrored content on the web is very high. For exact duplicates, it is estimated in over 30% [CSGM99].

We calculate a hash function of the contents of the pages to avoid storing the same content twice. To account for minor variations in the Web pages, this hash function is calculated *after the page have been parsed*, so two pages with identical content but different colors or formatting will still be detected as duplicates.

Note that this method only avoids storing the duplicate Web pages, it does not prevent downloading the page, and duplicate content can generate a waste of valuable network resources.

Recommendation: in our case, the crawler does not follow links from a Web page if the Web page is found to be a duplicate; applying this heuristic, we downloaded just about 6% of duplicates.

A.6 Server application programming

A.6.1 Embedded session ids

As a way of tracking users, some Web sites embed identifiers in the URLs (e.g. `/dir/page.html;-jsessid=09A89732`). These identifiers are later used for detecting logical sessions in log analysis. From the point of view of the Web crawler, these session ids are an important source of duplicates, because the crawler cannot accuratelly tell when two pages have semantically the same content.

A Web crawler must consider session-ids. “Unless prior knowledge of his fact is given to a crawler, it can find an essentialy unbounded number of URLs to crawl at this one site alone” [**?**] .

Typical variable names for storing session ids in the URLs include e.g.: `CFID`, `CFTOKEN`, `PHPSESSID`, `jsessionid`, etc. These variables are widely used in Web sites, and two pages that differ only in the session id are very likely to hold the same contents.

Recommendation: the crawler has a manually-built list of known session-id variables, and whenever it detects one, it changes the variable to a null value. We found that the Google crawler does not seem to download any page with a not-null value in the `PHPSESSID` variable (verified in June 2004), this can be checked by issuing a `allinurl:phpsessid` query.

A.6.2 Repeated path components

A common mistake when encoding links is to forget to include the root directory, e.g.: referencing `a/b/c` when we want to reference `/a/b/c`. This problem can accumulate, and it is common to find URLs with path components repeated several times, such as `a/b/c/c/c/c/`; this is due to dynamic pages in which the author has mistakenly created a relative link when it should be an absolute link.

These repeated path components usually refer to the same page, and the crawler downloads repeatedly the same information.

Recommendation: some crawler implementations, such as CobWeb [dSVG+99] discard repeated components in paths, as an heuristic to avoid this problem. Our heuristic of not following links from duplicate Web pages helps to avoid this problem, so we do not check explicitly for duplicate path components.

A.6.3 Slower or erroneous pages

Dynamically generated pages are, in general, slower to transfer than static pages, some times by a factor of 10 or 100, depending on the implementation and of caching issues. In some cases this is because building the page requires querying different sources of information, and in other cases this can be due to a programming error. A slow page can waste crawler resources by forcing it to keep a connection open for a long time.

Recommendation: besides a timeout, a lower speed limit should be enforced by the crawler.

A.7 Conclusions

The practical problems of Web crawling are mostly related to bad implementations of some Web servers and Web applications. These issues are not visible until a substantial amount of pages have been downloaded from the Web, and can affect the design of the Web crawler.

Implementing a Web crawler is, in a certain way, like building a vehicle for exploring the surface of mars: you need to build the vehicle to explore the terrain, but once you have tested it, you know more about the terrain and you have to modify your vehicle's design accordingly. In that sense, the wrong implementations we have presented in this chapter are just constraints that the Web crawler designer must consider: the Web crawler must accommodate to bad coding in the same way as Web browsers do.

However, the lack of good implementations imposes costs on the design of all kinds of applications. The idea of standards is to be able to interoperate. If the standars are not respected, then the only alternatives are either design for the smallest common denominator, or design for a proprietary, fixed platform. Both alternatives are detrimental to the quality of the Web.

Bibliography

[CCHM04] Nick Craswell, Francis Crimmins, David Hawking, and Alistair Moffat. Performance and cost tradeoffs in web search. In *Proceedings of the 15th Australasian Database Conference*, pages 161–169, Dunedin, New Zealand, January 2004.

[CSGM99] J. Cho, N. Shivakumar, and H. Garcia-Molina. Finding replicated web collections. In *ACM SIGMOD*, pages 355–366, 1999.

[dSVG+99] Altigran Soares da Silva, Eveline A. Veloso, Paulo Braz Golgher, Berthier A. Ribeiro-Neto, Alberto H. F. Laender, and Nivio Ziviani. Cobweb - a crawler for the brazilian web. In *Proceedings of String Processing and Information Retrieval (SPIRE)*, pages 184–191, Cancun, México, September 1999. IEEE Computer Society.

[FGM+99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and Tim Berners-Lee. RFC 2616 - HTTP/1.1, the hypertext transfer protocol. http://w3.org/Protocols/rfc2616/rfc2616.html, 1999.

[Koe04] Wallace Koehler. A longitudinal study of Web pages continued: a consideration of document persistence. *Information Research*, 9(2):(paper 174), January 2004.

[Kos93] Martijn Koster. Guidelines for robots writers. http://www.robotstxt.org/wc/guidelines.html, 1993.

[Per04] R. Scott Perry. DNS report. http://www.dnsreport.com/, 2004.

[Rag04] Dave Raggett. HTML tidy. http://tidy.sourceforge.net/, 2004. GPL software.