CHAPTER 2
STUDY OF SEARCH ENGINE REPOSITORY

2.1. INTRODUCTION

Certain web applications require local access to substantial portion of the web, which include text search engines, related page services and topic based search and categorization services. Such applications perform their operation on the local store because performing directly on the web would be too slow. In order to get relevant and quality results to the user query, the local repository or the search engine repository should be up to date. This chapter describes the structure of the search engine repository and analyses the edifice methods. This chapter also analyses the preferred edifice method of the web crawlers, and its pitfalls in building and maintaining the repository. Finally focus is narrowed down to the coherence problem exhibited by the search engines, which is the result of the updating procedure followed by them.

2.2 SEARCH ENGINE REPOSITORY

A web search engine can only retrieve the documents present in the search engine repository. The repository is the collective local knowledge about the web documents, which are ephemeral and with the property of frequent modification. The properties of web search engine repository that differentiate them from most databases include very large size, necessity of frequent
 updation, dynamic nature of the contents, high performance requirement. However the repository does not have to provide functionalities such as transactions, logging, and directory structure as other systems.

### 2.2.1 Structures

Most of the literatures on search engines analyse the drawbacks of the outcomes of the search engines. Very few literatures are available on the functioning of search engines. Therefore the repository structure explained in the thesis is specified by Brin and Page (1998). The repository contains the full HTML of every web page. Each page is compressed using zlib (Deutsch, 1996). The choice of compression technique is a tradeoff between speed and compression ratio. In the repository, the documents are stored one after the other and are prefixed by docID, length, and URL as shown in Figure 2.1. Every web page has an associated identification number called a docID, which is assigned whenever a new URL is parsed out of a web page.

The repository requires no other data structures to be used in order to access it. This helps with data consistency and makes development much easier and all the other data structures are rebuilt from the repository only. The basic functions performed by search engine repository using the data structure include providing an interface for the crawler to store updates, providing an efficient access API to the indexer and other modules to retrieve the pages.
The challenges that need to be addressed by search engine repository as discussed by Arasu et al (2001) and Hirai (2000) are complied as follows:

- **Scalability** - It must be possible to distribute the repository across a cluster of computers and disks, in order to cope with the size of the web.

- **Dual access modes** – The repository should support two different access modes equally efficiently. Random access mode is used to quickly retrieve a specific web page, given the page’s unique identifier. Streaming access mode is used to receive the entire collection, or some significant subset, as a stream of pages.

- **Expunging pages** - In most file or data systems, objects are explicitly deleted when no longer needed. However, when a web page is removed from web site, the repository is not notified. Thus the repository must have some mechanism for detecting and removing obsolete pages.

- **Large bulk updates** – Since the web changes rapidly, the repository needs to handle a high rate of modifications. As new versions of web pages are received from the crawler, the space occupied by older
versions must be reclaimed through space compaction and reorganization. In addition, excessive conflicts between the update process and the application accessing pages must be avoided.

The sizes of the search engine repository are often based on unaudited, self-reported information from each search engine. The search engine sizes should not count duplicate pages and spam pages. But it is believed that they are included in the reported size figures. The recent reported size of search engines as given by (Sullivan, 2005) are given in Table 2.1 Page depth indicates the size of the partial index of some pages.

Table 2.1 Size estimates of search engine repository (Danny Sullivan, 2005)

<table>
<thead>
<tr>
<th>SEARCH ENGINE</th>
<th>REPORTED SIZE</th>
<th>PAGE DEPTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>8.1 billion</td>
<td>100K</td>
</tr>
<tr>
<td>MSN</td>
<td>5.0 billion</td>
<td>150 K</td>
</tr>
<tr>
<td>Yahoo</td>
<td>4.2 billion</td>
<td>500 K</td>
</tr>
<tr>
<td>Ask Jeeves</td>
<td>2.5 billion</td>
<td>101 K</td>
</tr>
</tbody>
</table>

Some search engines extract index of some pages upto a depth of 800 KB. A search engine with many pages might actually be worse than one with fewer, if the index is not refreshed often or if the relevancy simply is not there. With search engines reporting increasing size of the repository, the repository building methods need to be analysed in order to understand the impact of the web page harvesting methods on the network and server resources.
2.2.2 Edifice methods

Every component of the product should perform efficiently in order to get the complete output of the product. When the search engine repository is well built, then the users have a chance for getting better results. Search engines compete against each other primarily based on the size and currency of their underlying database, in addition to the quality and response time of their ranking functions. Even the largest search engines, such as Google or Altavista, currently cover only limited parts of the web, and much of their data is several months out of date (Shkapenyuk, 2002).

2.2.2.1 Methods

By and large, search engines build their repository by using ★ paid listings ★ free add URL and ★ Link crawling. They use all the three techniques individually or in an association with other companies. Every major search engine company with significant traffic accepts paid listings as a form of advertising, guaranteeing the web page to appear in the top results for the interested terms.

Free Add URL technique allows direct submission of key pages to its crawler for free. Home page and one or two inside pages are submitted in case of any problem for a crawler to reach the home page. The search engine will crawl links from these pages to find other pages in the web site. Table 2.2 summarises key details about submitting to major search engines.
Table 2.2 Summaries key details about submitting to major search engine
(www.searchenginewarch.com)

<table>
<thead>
<tr>
<th>SEARCH ENGINE</th>
<th>ALLTHEWEB</th>
<th>ALTAVISTA</th>
<th>GOOGLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Listing tied to submit</td>
<td>No</td>
<td>Yes</td>
<td>No, but may help</td>
</tr>
<tr>
<td>Submit Limit</td>
<td>No limit</td>
<td>No limit</td>
<td>No limit</td>
</tr>
<tr>
<td>Pages appear</td>
<td>2-4 weeks</td>
<td>4-6 weeks</td>
<td>About 4 weeks</td>
</tr>
<tr>
<td>Overall freshness</td>
<td>1 day to 4 weeks</td>
<td>1 day to 6 weeks+</td>
<td>1 day to 4 weeks</td>
</tr>
<tr>
<td>Paid inclusion</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Inclusion / refresh</td>
<td>3 days 2 days</td>
<td>2 days Daily</td>
<td>N/A</td>
</tr>
<tr>
<td>Cost per year</td>
<td>$20 first ~$15 add</td>
<td>$78 first $38 - $58 add</td>
<td>N/A</td>
</tr>
<tr>
<td>Cost for 1, 10, 100 URLs</td>
<td>$20 $148 $1431</td>
<td>$78 $600 $3982</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Paid listings appear under the sponsored or featured tag of search results that will appear next to each title. Though the pages are paid for their listings, all search engine companies do not guarantee that the pages will rank well for particular terms but assures that they appear in the index. The subject matter of coverage may be more towards mercantile websites. Therefore the magnitude of coverage will be oriented towards business excluding the research, scientific and other category of sites. And also on the downside it should be paid.

In case of free add URL technique the search engines will not guarantee that any of the pages will be added through the use of the technique. Some search engines add URL submissions only as a backup to its link crawling and some consider them as junk pages due to excessive spamming compared to the three methods.
The magnitude of coverage in link crawling depends on the number of links that point to the web page. More the number of links, more the crawler is likely to index the page. Link crawling covers all pages irrespective of their category. Therefore the magnitude of coverage will be more compared to the paid listings. But, in this case, the crawler will index dynamic pages only to some extent and there are reasons for search engines to miss the pages because crawlers may not be able to locate during the normal crawls if the site has no link. Link crawling performed by the web crawlers are considered to be the reliable method of building search engine repository.

Many major search engines get their results by turning to third-party search providers to power their listings. To make matters more confusing, these search providers may run their own search engine sites, as well. Search engine repository built using any or combination of the above methods is refreshed using the web crawlers by revisiting the web pages checking for updates. The operation of web crawlers and their drawbacks in building and maintaining process is discussed in the following section.

2.2.3 Web Crawler

Crawler is a program that obtains some or all of the resources on a large number of sites primarily for the purpose of generating an inverted index to be used later in a search application. Search engines use separate crawlers to build their databases. Though the behaviour of crawlers in their scope, depth and frequency vary, the basic algorithm remains the same. The
search providers chart (Table 2.3) will provide an idea of how many web crawlers compete to harvest a single resource.

**Table 2.3 Search providers chart (Sullivan, 2004)**

<table>
<thead>
<tr>
<th>SEARCH ENGINE (Read Down)</th>
<th>PROVIDER GOOGLE</th>
<th>PROVIDER YAHOO</th>
<th>PROVIDER OPEN DIRECTORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Google</td>
<td>Main &amp; Paid</td>
<td>Paid (Overture)</td>
<td>Option</td>
</tr>
<tr>
<td>Yahoo</td>
<td>Main (Est. 10/04)</td>
<td>Main from Inktomi likely by 4/04</td>
<td></td>
</tr>
<tr>
<td>AOL</td>
<td>Main &amp; Paid (Est 10/05)</td>
<td>Main (Inktomi 12/05) Paid (Overture 6/05)</td>
<td>Option</td>
</tr>
<tr>
<td>MSN</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ask Jeeves (Main from Teoma owned by Ask)</td>
<td>Paid (9/05)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Overture (owned by Yahoo)</td>
<td></td>
<td>Paid (Overture) Backup (Inktomi)</td>
<td></td>
</tr>
</tbody>
</table>

Crawlers use breadth first and depth first policies to traverse the links. Though the coverage of the first method is wide and shallow, it causes the web server to have many rapid requests. The depth first traversal is deep and space complexity is cheaper but it is narrow. By a combination of breadth first and depth first sweep, the resources in a site are harvested. Table 2.4 shows a list of known crawlers used by some popular search engines. The agent name is the name assigned to the crawler tool and crawler host name is the machine from which the crawler originates. Many search engines other than one provided by the chart operates and causes excessive load to network bandwidth and web servers of the global database.
Table 2.4 Collection of search engines and crawlers
(Krishnamurthy, 2001)

<table>
<thead>
<tr>
<th>SEARCH ENGINE</th>
<th>AGENT NAME</th>
<th>CRAWLER HOSTNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Altavista</td>
<td>Scooter 2.0.G.R.A.B.x 20</td>
<td>Scooter.pa.xdec.com</td>
</tr>
<tr>
<td></td>
<td>Scooter/1.0</td>
<td></td>
</tr>
<tr>
<td>Google</td>
<td>Back Rub/2.1</td>
<td>Google.com</td>
</tr>
<tr>
<td>Inktomi</td>
<td>Slurp/2.0</td>
<td>*. inktomi.com</td>
</tr>
<tr>
<td>Infoseek</td>
<td>Infoseek Sidewinder/0.9</td>
<td>*. Infoseek.com</td>
</tr>
<tr>
<td>Lycos</td>
<td>Lycos spider (T.Rex)</td>
<td>Lycosidae.lycos.com</td>
</tr>
<tr>
<td>Northern Light</td>
<td>Gulliver/1.2</td>
<td>Taz.Northernight.com</td>
</tr>
</tbody>
</table>

Key differentiators between crawlers include the total number of sites visited, total number of resources fetched and frequency with which they span the web. Once a site has been indexed, the crawler has to revisit the site occasionally because the contents of the site may change. However, the contents of some sites don’t change as often as the contents of some other sites. The crawler should be intelligent enough to maintain some information on the rate of change of a site in order to reduce the amount of work it must do and to avoid contacting a site needlessly. Also some parts of a site may change more often than others.

Although the task of a web crawler is conceptually simple and deals with quantifiable electronic entities, in fact its practicalities make its algorithm a complex routine. The obstacles that have to be overcome in the engineering of crawler will be well known for designers, but are not well documented. Academic research is being carried out on web crawler, or its
unique features, rather than on the crawler itself. The situation is further complicated by the fact that the technology behind of many search engines is commercial secrets. With the increasing use of the results of web crawlers outside computer science, there is however a real need for an explicit discussion of the factors involved (Thewall, 2000).

The factors that affect the efficiency of web crawlers in updating the search engine repository include frequency of change, degree of change, and relevance of change. According to a recent study by Noutlas et al (2004), every week about 25% new links are created. After a year, about 80% of the links on the web are replaced by new ones. These results indicate that search engines need to update link based ranking metrics (such as Pagerank) very often. Given 25% changes every week, a week old ranking may not reflect the current ranking of the pages very well. Therefore metric used to rank important pages is not satisfying the requirement. The factors involved in crawler based search engine updation are discussed in detail in section 2.3 when discussing the coherence problem of web search engines. The following section discusses the problems created by the crawlers during the process of updation.

2.2.3.1 Problems

Given a responsive, finite and static web, it could be imagined that a simple crawler search could result in the discovery of all web pages (for which there exists a path from the start node). Unfortunately the web is large, effectively infinite, individual pages are dynamic and the bandwidth to any given server is
limited. It takes a nontrivial amount of time and network resources to download any given URL, and the contents of any given URL could change at any time (Glover, 2001). Over the years, more and more crawlers have been released on the web. Many web crawlers now make use of distributed or parallel technology in efforts to increase search engines web coverage. Web crawlers have had negative impacts on internet resources in the past and so while brute force crawling technique may help a crawler gain a small competitive edge, the impact on shared resources could be problematic (Gregory, 2002). Further sections describe the problems created by web crawlers to the network and web server to update the search engine repository.

Coherence Problem

In many applications it may be important to control how out-of-date information becomes, and to perform the refresh process so that data freshness is improved (Cho and Molina, 2000). Since web pages are modified at different frequencies, the crawler typically does not know when to revisit these pages. The crawler periodically polls the pages to check if they have changed. If the page has not changed since the last visit, the web server has wasted its resources. These resources could have been better-spent servicing users or sending crawler updates of pages that have actually changed (Brandman, 2000). This problem is magnified when hundreds of independent, competing web crawlers visit a web server over a period of time (Yuan, 2002).
Figure 2.2 Effect of increasing number of crawlers on crawler cache freshness (Gregory, 2002)

Figure 2.2 illustrates how freshness degrades as more crawlers are added to the system (as N increases) in the face of competition for shared resources (bandwidth). Increasing the freshness for web crawlers should not depend on using more crawlers; instead, crawlers should examine the fundamental way in which they regard competing peers.

Extra load on server and bandwidth

The network traffic depends on the frequency with which a crawler requests documents from the server. When this frequency increases, it has a high impact on the level of service offered by the server to web users. For example, when the crawler downloads page P on site S, the site needs to be transferred from its file system, consuming disk and CPU resources. The
retrieved page needs to be transferred through the network, which is another resource, shared by multiple organizations. If the server is busy with other jobs, then this becomes an additional load on the server (Martjin, 1997). Table 2.5 shows the activity of web crawlers as they request information from two busy web servers at the University of Waterloo: mefo7.unwaterloo.c1 and goedel.unwaterloo.ca.

<table>
<thead>
<tr>
<th>SERVER</th>
<th>AVERAGE % HITS DUE TO WEB CRAWLERS</th>
<th>AVERAGE % BYTES DUE TO WEB CRAWLERS PER DAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goedel</td>
<td>6.5 %</td>
<td>13.2 %</td>
</tr>
<tr>
<td>Mefo7</td>
<td>10.5 %</td>
<td>5.5 %</td>
</tr>
</tbody>
</table>

The Table 2.5 shows that even though web crawlers constitute about 0.5% of the overall number of clients, they account for significant portion of the clients causing data traffic to and from the web servers (Gregory, 2002).

Very little research is carried out on resource utilized by web crawlers in building and maintaining search engine repository. According to Yuan et.al estimates, 40% of current Internet traffic is due to web crawlers retrieving pages for indexing. To study the load imposed by crawlers, several experiments were conducted using the log files for the week from June 29, 2001 to July 5, 2001, for the web server by Yuan et al in the Department of Computing Science at the University of Alberta. Figure 2.3 represents the crawler hit and byte percentages. The average hit and byte percentages are 27.3% and 20.9% respectively. A maximum of 40.6% of all hits was due to
crawler. The percentage of bytes fetched by crawlers reaches a maximum of 29.5%.

![Crawler hit versus byte percentage](Yuan_2002)

**Figure 2.3 Crawler hit versus byte percentage (Yuan, 2002)**

**Comparison of crawler algorithms**

Crawler research has lead to introduction of different type of crawlers such as mobile crawlers, distributed crawlers, autonomic cooperating crawlers and focused crawlers.

Mobile crawlers are transferred to the source(s) where the data resides in order to filter out any unwanted data locally before transferring it back to the search engine. The mobility in the context of web crawling is explained as the ability of a crawler to transfer itself to each web server of interest before collecting pages on that server. After completing the collection process on a particular server, the crawler together with the collected data moves to the
next server or to its home system. Each crawler through crawling algorithm controls crawling strategy and path taken separately. By using mobile crawlers, the HTTP overhead is reduced by transferring the crawler to the host server but the crawler must have permission from the owner of the web site to operate locally from the server and this increases the server load, though remote page filtering helps, the remote page selection imposes load on the web server.

Table 2.6 Comparison of web crawlers

<table>
<thead>
<tr>
<th>CRAWLER</th>
<th>BENEFITS</th>
<th>DRAWBACKS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mobile Crawlers</td>
<td>HTTP overhead is reduced</td>
<td>• Needs permission of the owner of web site</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Operating locally increases the server load</td>
</tr>
<tr>
<td>Distributed web crawler</td>
<td>Resilient against system crashes</td>
<td>Population explosion of more capacity distributed crawler may lead to clogging of network.</td>
</tr>
<tr>
<td>Parallel crawler</td>
<td>Maximizes download rate</td>
<td>Coordination suffers significant communication overhead</td>
</tr>
<tr>
<td>Focused Crawler</td>
<td>Retrieves relevant pages only</td>
<td>Assigning of reliable credit to the pages</td>
</tr>
</tbody>
</table>

Shkapenyuk(2002) research on web crawlers concluded with distributed web crawler which runs on a network of workstations. The crawler scales to (at least) several hundred pages per second, is resilient against system crashes and other events and can be adapted to various crawling applications. Many crawlers now make use of distributed crawlers. Population explosion of more capacity distributed crawlers may lead to clogging of the network.
Many search engines often run multiple processes in parallel to perform the task of crawling in order to maximize the download rate. Clearly, these parallel crawlers should not be coordinated properly, so that different crawlers do not visit the same website multiple times, and adopted crawling policy should be strongly enforced. The coordination can incur significant communication overhead, limiting the number of simultaneous crawlers (Arasu, 2001).

Focussed crawlers (Chakrabarthi, 1999) aim to search and retrieve only the subset of the world wide web that pertains to a specific topic of relevance. The ideal focussed crawler retrieves the maximal set of relevant pages while simultaneously traversing the minimal number of irrelevant documents on the web. Diligenti, Coetzee et al (2000) state the major open problem in focussed crawling is that of properly assigning credit to all pages along a crawl route that yields a highly relevant document. In the absence of reliable credit assignment strategy, focussed crawlers suffer from a limited ability to sacrifice short term document retrieval gains in the interest of better overall crawl performance.

All the crawlers discussed above focus to solve some problem of general crawlers in order to enhance its performance like mobile crawlers aims to reduce the HTTP overhead, parallel crawlers maximizes the download rate etc. All the web crawlers follow the bottom up approach in building and maintaining the search engine repository. The web page is updated in the remote server, and the bottom up approach followed by the crawlers may not
improve the freshness of the repository due to lack of mechanism to exactly predict the frequency change.

2.3 PROBLEM OF COHERENCE

Web crawlers use polling technique to update the search engine repository. The results of the polling technique are only an estimation of the actual values and there are possible chances of more than one web page change between two accesses.

2.3.1 Needless requests

Under the conventional crawling scheme, a crawler must make an HTTP request to find out if a page has changed. If the page has not changed then the HTTP request was needless. Figure 2.4 illustrates how needless downloads occur. Say a page changes C times in a fixed time period while a crawler requests the page R times at regular intervals Onn Bradmann et al calculate the minimum needless requests as follows

\[
\text{MinNeedlessReq}(C; R) = \begin{cases} 
R - (C + 1) & \text{if } R > C + 1 \\
0 & \text{otherwise}
\end{cases}
\]

The value C is incremented by one to account for gathering the initial version of the page. For example, consider a page that changes five times over a month and a crawler that downloads the page ten times for that month. The crawler's first request is not needless because the crawler has no copy of the document. However, of the subsequent nine requests, at least four are needless. Figure 2.5 represents the needless downloads.
Figure 2.4 Representation of requests and changes (Cho, 2000)

The horizontal axis of Figure 2.5 represents the number of times each page was visited by the crawler during the 100 day period and the vertical axis shows the fraction of needless downloads. It can be seen that the crawlers download more and more needless pages as they visit pages more often. For instance, when the crawler visits all pages once every day, around 77% of downloads are needless.

Figure 2.5 Needless requests (Brandman, 2000)
Needless requests that occur in the crawler’s maintenance process need not result in needless download. The crawler can use HTTP’s If-Modified-since feature to avoid downloading a file as a result of a needless request. This feature causes the web server to only transfer a file if it has change since some date (e.g., the date of the last crawl). However, many crawlers ignore this feature and many web servers do not export this functionality. According to Bradmann et al 87% of crawlers do not check time-stamps before downloading a page. For such crawlers, needless requests result in needlessly downloaded pages, wasting even more web server (and crawler) resources.

### 2.3.2 Freshness metrics

A web page is said to be fresh or coherent when the local copy matches the source exactly. A local web page become stale or outdated when the page at the source is updated. The crawler based search engine freshness depends on freshness metrics to visit the web page and check for updation. The freshness of a local element $e_i$ at time $t$ as defined by (Cho and Molina 2003) is

$$F(e_i, t) = \begin{cases} 
1 & \text{if } e_i \text{ is up to date at time } t \\
0 & \text{otherwise}
\end{cases}$$

Similarly the age of web page $e_i$ at time (Cho and Molina, 2000) is

$$A(e_i, t) = \begin{cases} 
0 & \text{if } e_i \text{ is up-to-date at time } t \\
t & \text{modification time of } e_i \text{ otherwise}
\end{cases}$$
Since the freshness and age of local page change overtime, not a specific instance in time $t_i$ was found useful to define a metric that considered freshness over a period of time (Cho and Molina, 2000), the average overtime metric defined as follows

$$F(e_i) = \lim_{t \to \infty} F(e_i, t) \, dt$$

The goal of another study conducted by Cho and Molina (2003) is to synchronise web pages to maximize the freshness as perceived by users, since users may access some pages more than other. A weight $w_i$ as a measure of importance, is assigned to each element $e_i$. Freshness of a collection $S$ of size $N$ is then defined as

$$F(s, t) = \frac{\sum_{i=1}^{N} w_i F(e_i, t)}{\sum_{i=1}^{N} w_i}$$

A similar equation follows the age. The study found that the optimal synchronization frequency is not proportional to weight. It was also found that when the weight of $e_1$ is $K$ times as large as that of $e_2$, the optimal age curve that $e_1$ follows is $(k)$ times larger than the curve $e_2$ follows.

### 2.3.3 Factors involved in crawler based updation

The factors involved in crawler-based updation include relevance of change, frequency of change and degree of change of web pages in the remote server. The crawlers predetermine the values of these factors and operate based on them. The performance of crawler-based updation is purely based on the accuracy of the factors. The following sections describe the factors in detail.
2.3.3.1 Relevance of change

When a change is made to a web page, it may be irrelevant, resulting in a page unnecessarily being refreshed. Detecting changes can be easily automated, but evaluating the change is difficult. Revilla and Koehler (2000) dealt with deciding if certain types of changes to web pages are relevant. These papers present the results of a user survey in relative importance of changes in structure (navigation pages), content (content pages) and presentation. Of these changes, people expressed a desire to be notified of structural changes as they found it difficult to notice when links changed to point elsewhere. Table 2.7 represents the extant distribution by level in percentages.

<table>
<thead>
<tr>
<th></th>
<th>DEC 1996</th>
<th>FEB 2001</th>
<th>MAY 2003</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Navigation</strong></td>
<td>50.4</td>
<td>61.3</td>
<td>72.2</td>
</tr>
<tr>
<td><strong>Content</strong></td>
<td>49.6</td>
<td>38.7</td>
<td>27.8</td>
</tr>
<tr>
<td><strong>Total N</strong></td>
<td>361</td>
<td>124</td>
<td>122</td>
</tr>
</tbody>
</table>

Content changes were rated as overall more significant, but less necessary to require notification. Changes to presentation were considered largely irrelevant. Kohelar (2003) concludes that once a collection has sufficiently aged, it may stabilize in the sense at least that its URLs may be more durable in time. But few well-behaved crawlers consider the relevance of the change when they are updating the repository.
### 2.3.3.2 Frequency of change

Frequency of change means how many times a page changed within a particular interval (for example 3 changes in 1 month). When estimating frequency of change, there are challenges that need to be overcome. Firstly, the change history may be incomplete; it may not be known when and how often a web page is changed. Secondly, web pages can be accessed at regular intervals.

Noutlas (2004) et al observed in their 54-week study (Fig.2.6) that a significant fraction of pages (around 50%) that occurred in each weekly snapshot remained unchanged throughout the course of their study. Another quite large portion of pages changed very often: approximately 15% of pages underwent at least one change between each weekly download. These two extremes account for more than 65% of the pages. The remaining pages occurring in snapshots have average change intervals ranging across the spectrum in a roughly U-shaped pattern, with most pages concentrated near one of the two extremes. The tendency is that most pages either change very frequently or very infrequently.

![Figure 2.6 Distribution of average change intervals of pages (Noutlas, 2004)](image)
Work done by Cho and Molina (2003) estimated the frequency of change of incomplete data, assuming that an element was accessed repeatedly through normal activities such as crawling. They developed an estimator for the existence of change that performed well even if the changes were not following a Poisson model.

2.3.3.3 Degree of change

From the point of view of a web search engine, degree of change is as important as, if not more important than, presence of change. Due to the immense scale and highly dynamic nature of the web, search engines are faced with a constrained optimization problem: maximize the accuracy of the local search repository and index, given a constrained amount of resources available for re (downloading) pages from the web incorporating them into search index. Search engines that ignore degree of change may waste precious resources downloading pages that have changed in only trivial ways and have little impact on the quality of search service. Effective search engine crawlers ignore insignificant changes and devote resources to incorporating important changes instead.

2.3.4 Dead links

A dead page is a page that is not publicly available over the web. A page can be dead for any of the following reasons: (1) its URL is malformed; (2) its host is down or non-existent; or (3) it does not exist on the host. The first two types of dead pages are easy to detect: the former fail the URL parsing and the latter fail the resolution of the host address. When fetching pages that are not
found on a host, the web server of the host is supposed to return an error; typically, it is the (in) famous 404 HTTP return code. However, it turns out that many web servers today do not return an error code even when they receive HTTP requests for non-existent pages. Instead, they return an OK code (200) and some substitute page; typically, this substitute is an error message page or the home page of that host or even some completely unrelated page. The authors Yossef et al term those non-existent pages that behave as above as "soft-404 pages". The existence of soft-404 pages makes the task of identifying dead pages non-trivial. According to their study the soft 404 contributes to 25% of the dead links.

The second problem of dead links as a decay signal is that this is a very noisy signal. One reason is because it is easy to manipulate. Indeed, many commercial sites use content management systems and quality check systems that automatically remove any link that results in a 404 code. Youssef et al (2004) experiments indicate that the Yahoo! Taxonomy is continuously purged of any dead links (Fig 2.7). However, this is hardly an indication that every piece of the Yahoo! taxonomy is up-to-date. Another reason for the noisiness is that pages of certain types tend to live "forever" even though no one maintains them.
CERTIFICATE

This is to certify that the thesis entitled "Design and Development of Autonomic Updation and Semantic Search Process for Search Engines" submitted to the Bharathiar University, in Partial fulfillment of the requirements for the award of the Degree of Doctor of Philosophy in Computer Science is a record of original research work done by Mrs. A. Vijayalakshmi during the period Jan 2000 – Oct 2005 of his/her study in the Department of Computer Science and Engineering at Bharathiar University (College / Research Institute / Bharathiar University), under my supervision and guidance and the thesis has not formed the basis for the award of any Degree/ Diploma /Associateship /Fellowship or other similar title to any candidate of any University.

Countersigned

Signature of the Guide

Lecturer
Dept. of Comp. Sci. & Engg.
Bharathiar University
Coimbatore - 641 048

Head of the Department

K. K. Gnanasekar

21/11/05

Department of Computer Science and Engineering
Bharathiar University
Coimbatore - 641 048.
Figure 2.7 Average decay scores and fraction of dead links for 30 Yahoo nodes (Youssef, 2004)

2.4 DISCUSSION

Cho and Molina (2003) pioneers in search engine repository research in their paper are trying to synchronise the point of releasing a crawler to look for updated web page and the point of updation of the web page. They discuss the several dimensions to the synchronization process such as synchronization frequency, resource allocation, synchronization order and synchronization points. The synchronization order policies categorized by them include fixed order, random order and purely random order.

In the fixed order, all elements in the database are synchronized in the same order repeatedly. Random order synchronization policy synchronizes all elements repeatedly but synchronization order may be different in each
iteration. Purely random policy proposed by authors is a hypothetical policy, where at each synchronization point, a random element is selected from the database and synchronized. The poisson distribution of various refresh policies depict a significant difference in the temporal quality of the data that is collected.

Studies of Noutlas (2004) et al conclude that the frequency of change of the web pages is not a reliable factor and therefore the effectiveness of the synchronization policies stay questioned. The updating scheme proposed in the thesis eliminates the dependency of the uncertain frequency of change factor of the remote server and provides an alternate solution to avoid the needless requests of the web crawlers.

Web Recency Maintenance Protocol (WRMP) employs push mechanism to maintain the current status of the world wide web at search engine site. WRMP ensures the real time update by initiating an update agent as and when the server is modified or updated. The availability of high processing speed enables the search engine to maintain the status of the whole WWW in real time. WRMP transfers the entire content of the updated document. This research concentrates on partial upload of the updated document and proposes to upload the changes by notifications from the web servers of the global database.