Chapter 2

Related work

In this chapter we review selected publications related to the topics covered in this thesis.

We start in section 2.1 with an overview of publications on Web search and link analysis in general, in section 2.2 we review specific issues of Web crawling and their solutions, and in section 2.3 we cover the architecture of existing Web crawlers. In section 2.4 we summarize the results of several studies about Web characterization which include results relevant for Web crawling.

2.1 Searching and ranking Web pages

2.1.1 Traditional information retrieval and Web information retrieval

Information Retrieval (IR) is the area of computer science concerned with retrieving information about a subject from a collection of data objects. This is not the same as Data Retrieval, which in the context of documents consists mainly in determining which documents of a collection contain the keywords of a user query. Information Retrieval deals with satisfying a user need, which is not easy:

> "... the IR system must somehow 'interpret' the contents of the information items (documents) in a collection and rank them according to a degree of relevance to the user query. This 'interpretation' of a document content involves extracting syntactic and semantic information from the document text ..." [BYRN00]

Although there was an important body of Information Retrieval techniques published before the invention of the World Wide Web, there are unique characteristics of the Web which made them unsuitable or insufficient. A survey by Arasu *et al.* [ACGM$^+$01] on searching the Web notes that:

> "IR algorithms were developed for relatively small and coherent collections such as newspaper articles or book catalogs in a (physical) library. The Web, on the other hand, is massive, much less coherent, changes more rapidly, and is spread over geographically distributed computers ...".

This idea is also present in a survey about Web search by Brooks [Bro03], which states that a distinction could be made between the "closed Web", which comprises high-quality controlled collections on which a search engine can fully trust, and the "open Web", which includes the vast majority of Web pages and on which traditional IR techniques concepts and methods are challenged.

The low cost of publishing in the "open Web" is a key part of its success, but implies that searching information on the Web will always be inherently more difficult that searching information in traditional, closed repositories. One of the main challenges the open Web poses to search engines is "search engine spamming" (i.e.: malicious attempts to get an undeserved high ranking in the results). This has created a whole branch of Information Retrieval called "adversarial IR", which is related to retrieving information from collections in which a sub-set of the collection has been manipulated to influence the algorithms.

2.1.2 Connectivity-based ranking

Web links provide a source of valuable information. In a context in which the number of pages is very large, and there are, in general, no trusted measures for asserting the quality of pages, Web links can be used as a collective, "emerging" measure of page quality.

The key assumption of connectivity-based ranking is that a hyperlink from a page P' to a page P, means, in a certain way, that the content of page P is endorsed by the author of page P'. Several algorithms for connectivity-based ranking are the subject of a survey by Henzinger [Hen01]. The algorithms can be partitioned into:

- *Query-independent ranking*, which assign a fixed score to each page in the collection.
- *Query-dependent ranking*, or topic-sensitive ranking, which assign a score to each page in the collection in the context of a specific query.

2.1.2.1 Query-independent ranking

The first connectivity based query-independent ranking method was called Hyperlink Vector Voting (HVV) and was introduced by Li [Li98]. The HVV method uses the keywords appearing inside links to a Web page to confer to that page a higher score on those keywords. Only the count of keyword-link pairs is used, so this ranking function is relatively easy to manipulate.

The Pagerank algorithm, introduced by Page *et al.* [PBMW98], is currently used as an important part of its ranking function by the Google search engine [goo04]. The definition of Pagerank is recursive, stating in simple terms that "a page with high Pagerank is a page referenced by many pages with high Pagerank". Pagerank can be seen as a recursive HVV method.

To calculate the Pagerank, each page on the Web is modeled as a state in a system, and each hyperlink as a transition between two states. The Pagerank of a page is the probability of being in a given page when the system is in its stationary state.

A good metaphor for understanding this is to consider a "random surfer", which visits pages at random, and upon arrival to each page, chooses an outgoing link uniformly at random from the links in that page. The Pagerank of a page is the fraction of time that the random surfer spends at each page.

This simple system can be modeled by the following equation of a "simplified Pagerank" ($\Gamma^-(p)$ is the set of pages pointing to p, and $\Gamma^+(p)$ is the set of pages p points to, see Section **??** for the notation).

$$Pagerank'(P) = \sum_{x \in \Gamma^-(P)} \frac{Pagerank'(x)}{|\Gamma^+(x)|} \quad (2.1)$$

However, actual Web graphs include many pages with no out-links, which act as “rank sinks”, as they accumulate rank but never distribute it to other pages. Also, we would like pages to accumulate ranking, and not passing all of their score to other pages. For these reasons, most of the implementations of Pagerank add “random jumps” to each page, which are hyperlinks to all pages in the collection, including itself.

In terms of the random surfer model, we can state that when choosing the next step, the random surfer either chooses a page at random from the collection with probability ε, or chooses to follow a link from the current page with probability $1-\varepsilon$. This is the model used for calculating Pagerank in practice, and it corresponds to the following equation:

$$Pagerank(P) = \frac{\varepsilon}{N} + (1-\varepsilon) \sum_{x \in \Gamma^-(P)} \frac{Pagerank(x)}{|\Gamma^+(x)|} \tag{2.2}$$

N is the number of pages in the collection, and the parameter ε is typically between 0.1 and 0.2, based on empirical evidence. Pagerank is a global, static measure of quality of a Web page, and that is very useful in terms of computation time, as it only has to be calculated once at indexing time, and is later used repeatedly at query time.

A different paradigm for static ranking on the Web is the network flows model introduced by Tomlin [Tom03]. For ranking pages, a (sub)graph of the Web is considered as carrying a finite amount of fluid, and edges between nodes are pipes for this fluid. Using an entropy maximization method, two measures are obtained: a “TrafficRank” which is an estimation of the amount of flow through a node in this network model, and a “page temperature”, which is a measure of the importance of a Web page, obtained by solving the dual of this optimization problem. Both measures can be used for ranking Web pages.

The models presented in this section summarize each page on the Web to a single number, or a pair of numbers, but as the creators of Pagerank note, “the importance of a Web page is an inherently subjective matter, which depends on readers interests, knowledge and attitudes” [PBMW98]; that is why query-dependent ranking is introduced to create ranking functions which are sensitive to user’s needs.

2.1.2.2 Query-dependent ranking

In query-dependent ranking, the starting point is a “neighborhood graph”, a set of pages which are expected to be relevant to the given query. Carriere and Kazman [CK97] propose to build this graph by starting with a set of pages containing the query terms; this set can be the list of results given by a full-text search engine. This *root set* is augmented by its “neighborhood”, which comprises all (or a large sample) of the pages directly pointing, or directly pointed by, pages in the root set. The construction procedure is shown in Algorithm 1.

Figure 2.1 depicts the process of creation of the neighborhood set. The idea of limiting the number of pages added to the neighborhood set by following back links was not part of the original proposal, but was introduced later [BH98].

It is customary that when considering links in the neighborhood set, only links in different Web sites are considered, as links between pages in the same Web site are usually created by the same authors as the pages themselves, and do not reflect the relative importance of a page for the community in general.

The most-cited algorithm, presented by Yuwono and Lee [YL96], is the most simple form of connectivity-based query-dependent ranking: after the neighborhood set has been built, each page P in it is assigned a score which is the sum of the number of query terms appearing in the pages pointing to P. This algorithm performs poorly when compared with pure content-based analysis, so the authors concluded that links by themselves are not a reliable indicator of semantic relationship between Web pages.

Algorithm 1 Creation of the neighborhood set S_σ of query σ

Require: σ query
Require: $t > 0$, size of root set.
Require: $d > 0$ number of back-links to include per page.
$R_\sigma \leftarrow$ top t results using a search engine.
$S_\sigma \leftarrow \emptyset$
for all $p \in R_\sigma$ **do**
  Let $\Gamma^+(p)$ denote all pages p points to
  Let $\Gamma^-(p)$ denote all pages pointed by p
  $S_\sigma \leftarrow S_\sigma \cup \Gamma^+(p)$
  if $|\Gamma^-(p)| \leq d$ **then**
    $S_\sigma \leftarrow S_\sigma \cup \Gamma^-(p)$
  else
    $S_\sigma \leftarrow S_\sigma \cup$ an arbitrary set of pages in $\Gamma^-(p)$
  end if
end for
S_σ is the neighborhood set of query σ

A more elaborated idea is the HITS algorithm presented by Kleinberg [Kle99], which is based on considering that relevant pages can be either "authority pages" or "hub pages". An authority page is expected to have relevant content for a subject, and a hub page is expected to have many links to authority pages.

The HITS algorithm produces two scores for each page, which are called "authority score" and "hub score". These two scores have a mutually-reinforcing relationship, namely: a page with high authority score is pointed to by many pages with a high hub score, and a page with a high hub score points to many pages with a high authority score, as shown in Figure 2.2.

An iterative version of this algorithm is shown in Algorithm 2; in this version, the number of iterations is fixed, but the algorithm can be adapted to stop based on the convergence of the sequence of iterations.

Algorithm 2 Hub and authority score for each page in S_σ

Require: S_σ neighborhood set of query σ
Require: k number of iterations
$n \leftarrow |S_\sigma|$
Let z denote the vector $(1, 1, 1, \ldots 1) \in R^n$
$H_0 \leftarrow z$
$A_0 \leftarrow z$
for $j = 1$ to k **do**
  for $i = 1$ to n **do**
    $H_j[i] \leftarrow \sum_{x \in \Gamma^+(i)} A_{j-1}[x]$ $\{\Gamma^+(i)$ are pages i points to$\}$
    $A_j[i] \leftarrow \sum_{x \in \Gamma^-(i)} H_j[x]$ $\{\Gamma^-(i)$ are pages pointing to $i\}$
  end for
  Normalize H_j and A_j so their components sum 1
end for
H_k is the vector of hub scores
A_k is the vector of authority scores

The HITS algorithm suffers from several drawbacks in its pure form. Some of them were noted and solved

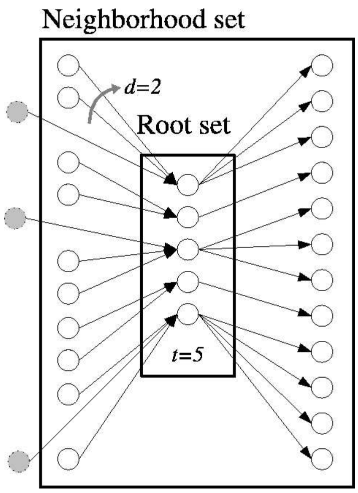

Figure 2.1: Expansion of the root set with $t = 5$ and $d = 2$. t is the number of pages in the root set, and d is the maximum number of back-links to include in the neighborhood set.

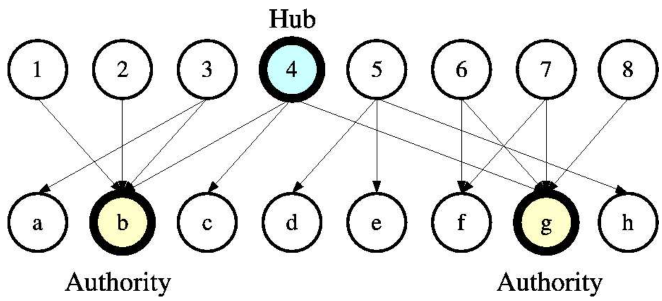

Figure 2.2: Hubs and authorities in a small graph. Node 4 is the best Hub, as it points to many authorities, and nodes b and g are the best authorities.

by Bharat and Henzinger [BH98]:

(a) Documents in the neighborhood set which are not about the original topic ("topic drifting").

(b) Nepotistic, mutually-reinforcing relationships between hosts.

(c) Automatically generated links.

Problem (a) is the most important, as while expanding the root set, it is common to include popular pages which are highly-linked, but unrelated to the query topic. The solution is to use analysis of the contents of the documents when executing Algorithm 1, and pruning the neighborhood graph by removing documents which are too different from the query. This is done using a threshold for the standard TF-IDF measure of similarity [SB88] between documents and queries.

Problems (b) and (c) can be avoided using the following heuristic is used: if there are k edges from documents on a host to documents in another host, then each edge is given a weight of $1/k$. This gives each document the same amount of influence on the final score, regardless of the number of links in that specific document.

A different variation of the HITS algorithm, designed specifically to avoid topic drifting, was presented by Chakrabarti *et al.* [CDR$^+$98]. In their approach, for each link, the text near the hyperlink in the origin page, and the full text of the destination page are compared. If they are similar, then that link is given a high weight, as it carries information about semantic similarity between the origin and destination pages. As this heuristic keeps the pages in the neighborhood more closely related, a more relaxed expansion phase can be done. The authors propose to follow two levels of links forward and backward from the root set, instead of just one.

Another approach to query-dependent ranking is topic-sensitive Pagerank, introduced by Haveliwala [Hav02], multiple scores for each page are pre-computed at indexing time, using an algorithm similar to Pagerank. Each score represents the importance of a page; not the global importance, but the relevance of the page for each topic from a set of pre-defined topics. At query time, the ranking is done using the query to assign weights to the different topic-sensitive Pagerank scores of each page.

2.2 Web crawling issues

There are two important characteristics of the Web which generate a scenario in which Web crawling is very difficult: its large volume and its rate of change, as there is a huge amount of pages being added, changed and removed every day. The large volume implies that the crawler can only download a fraction of the Web pages in a given time, so it needs to prioritize its downloads. The high rate of change implies that by the time the crawler is downloading the last pages from a site, it is very likely that new pages have been added to the site, or that pages that have already been updated or even deleted.

Crawling the Web, in a certain way, resembles watching the sky in a clear night: what we see reflects the state of the stars at different times, at the light travels different distances. What a Web crawler gets is not a "snapshot" of the Web, because it does not represents the Web at a given instant of time [BYRN00]. The last pages being crawled are probably very accurately represented, but the first pages that were downloaded have a larger probability of being changed. This idea is depicted in Figure 2.3.

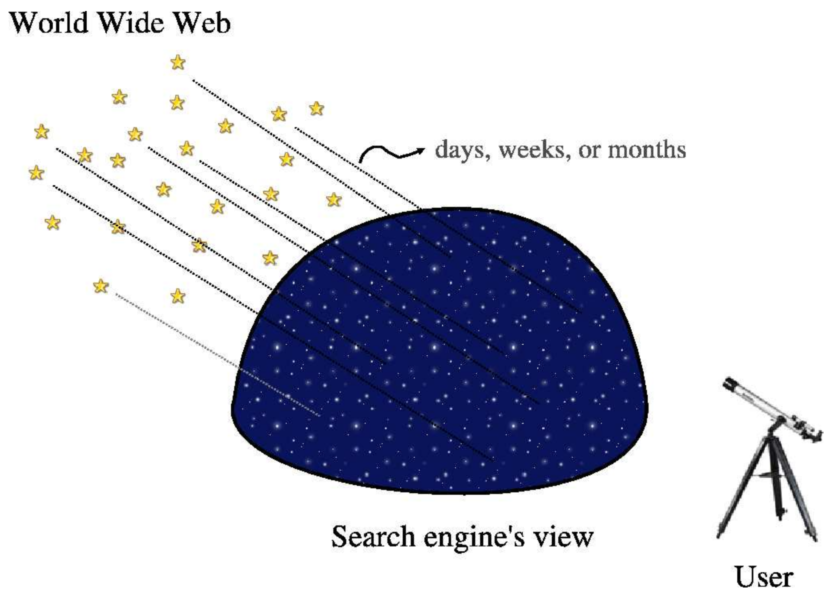

Figure 2.3: The search engine's view of the Web represents the state of Web pages at different times, as the view of the sky at night presents the state of stars at different times.

As Edwards *et al.* note, "Given that the bandwidth for conducting crawls is neither infinite nor free it is becoming essential to crawl the web in a not only scalable, but efficient way if some reasonable measure of quality or freshness is to be maintained." [EMT01]. A crawler must carefully choose at each step which pages to visit next.

The behavior of a Web crawler is the outcome of a combination of policies:

- A *selection policy*, which states which pages to download.
- A *re-visit policy*, which states when to check for changes to the pages.
- A *politeness policy*, which states how to avoid overloading Web sites.
- A *parallelization policy*, which states how to coordinate distributed Web crawlers.

2.2.1 Selection policy

Given the current size of the Web, even large search engines index only a portion of the publicly available content; a study by Lawrence and Giles [LG00] showed that no search engine indexes more than 16% of the Web. As a crawler always downloads just a fraction of the Web pages, it is highly desirable that the downloaded fraction contains the most relevant pages, and not just a random sample of the Web.

This requires a metric of importance for prioritizing Web pages. The importance of a page is a function of its intrinsic quality, its popularity in terms of links or visits, and even of its URL; the later is the case of vertical search engines restricted to a single top-level domain, or search engines restricted to a fixed Website.

Designing a good selection policy has an added difficulty: it must work with partial information, as the complete set of Web pages is not known during crawling.

In experiments by Cho *et al.* [CGMP98], a series of importance metrics were tested to download pages from the `stanford.edu` domain. They found that ordering pages by Pagerank (calculated over the partial set of pages already downloaded) leads to crawl highly-referenced pages first, regardless of the starting URLs, while ordering by pure count of references results in a bias towards locally important pages, and does not achieve a good global ordering.

In a different study by Najork and Wiener [NW01], several million pages across 7 million Web sites were downloaded using different policies. They discovered that breadth-first crawl is the best strategy to download pages with good Pagerank early in the crawl, and that the quality of downloaded pages deteriorates as the crawling process advances. The explanation given by the authors for this result is that "the most important pages have many links to them from numerous hosts, and those links will be found early, regardless of on which host or page the crawl originates".

The importance of a page for a crawler can also be expressed as a function of the interest of the page to the user, measured as the similarity of a page to a query. This is called "focused crawling" and was introduced by Chakrabarti *et al.* [CvD99]. The problem with focused crawling is that, in the context of a Web crawler, we would like to be able to predict the similarity of the text of a given page to the query *before* actually downloading the page. A possible predictor is the anchor text of links; this was the approach taken by Pinkerton [Pin94] in a crawler developed in the early days of the Web. Diligenti *et al.* [DCL$^+$00] propose to use the complete content of the pages already visited to infer the similarity between the driving query and the pages that have not been visited yet.

2.2.2 Re-visit policy

The Web has a very dynamic nature, and crawling a fraction of the Web can take a long time, usually measured in weeks or months. By the time a Web crawler has finished its crawl, many events could have happened. We characterize these events as creations, updates and deletions [BYCSJ04]:

Creations When a page is created, it will not be visible on the public web space until it is linked, so we assume that at least one page update -adding a link to the new web page- must occur for a web page creation to be visible.

Updates Page changes are difficult to characterize: an update can be either *minor* -at the paragraph or sentence level, so the page is semantically almost the same and references to its content are still valid- or *major* -all references to its content are not valid anymore. It is customary to consider **any** update as *major*, as it is difficult to judge automatically if the page's content is semantically the same. Characterization of partial changes is studied in [LWP$^+$01, NCO04].

Deletion A page is deleted if it is removed from the public web, or if all the links to that page are removed. Undetected deletions are more damaging for a search engine's reputation than updates, as they are more evident to the user. The study by Lawrence and Giles about search engine performance [LG00] reports that on average 5.3% of the links returned by search engines point to deleted pages.

2.2.2.1 Cost functions

From the search engine's point of view, there is a cost associated with not detecting an event, and having an outdated copy of a resource. The most used cost functions, introduced in [CGM00], are freshness and age.

Freshness This is a binary measure that indicates whether the local copy is fresh or not. The freshness of a page p in the repository, at time t is defined as:

$$F_p(t) = \begin{cases} 1 & \text{if } p \text{ is not modified at time } t \\ 0 & \text{otherwise} \end{cases} \tag{2.3}$$

Age This is a measure that indicates how outdated the local copy is. The age of a page p in the repository, at time t is defined as:

$$A_p(t) = \begin{cases} 0 & \text{if } p \text{ is not modified at time } t \\ t - \text{modification time of } p & \text{otherwise} \end{cases} \tag{2.4}$$

The evolution of these two quantities is depicted in Figure 2.4.

Coffman *et al.* [EGC98] worked with a definition of the objective of a Web crawler which is equivalent to freshness, but use a different wording: they propose that a crawler must minimize the fraction of time pages remain outdated. They also noted that the problem of Web crawling can be modeled as a multiple-queue, single-server polling system, on which the Web crawler is the server and the Web sites are the queues. Page modifications are the arrival of the customers, and switch-over times are the interval between page accesses to a single Web site.

2.2.2.2 Estimating freshness and age

The probability that a copy of p is up-to-date at time t, $u_p(t) = P(F_p(t) = 1)$ decreases with time if the page is not re-visited.

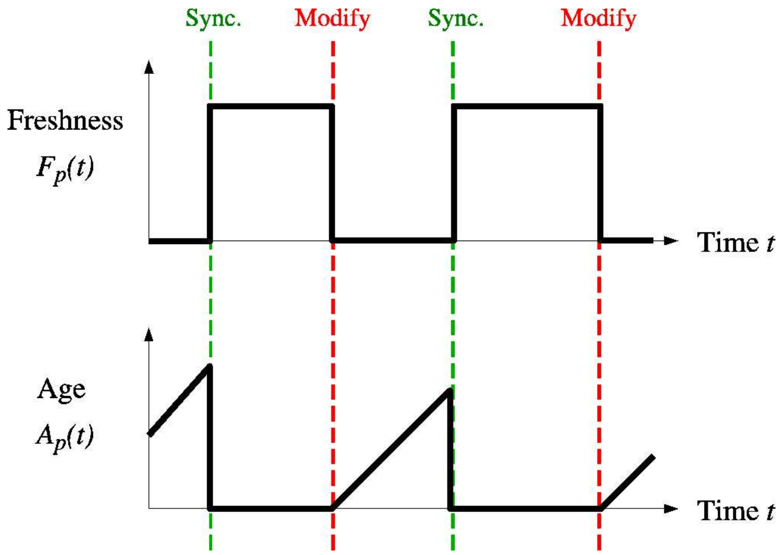

Figure 2.4: Evolution of freshness and age with time. Two types of event can occur: modification of a web page in the server (event *modify*) and downloading of the modified page by the crawler (event *sync*).

Brewington and Cybenko [BCS$^+$00] considered that if changes to a given page occur at intervals which are independent, i.e., page change is a memory-less process, then this can be modeled as a Poisson process. However, it is worth noticing that most Web page changes exhibit certain periodicity - because most of the updates occur during business hours in the relevant time zone for the studied sample - so the estimators that do not account for this periodicity are more valid on the scales of weeks or months than on smaller scales.

When page changes are modeled as Poisson process, then if t units of time have passed since the last visit, then:

$$u_p(t) = e^{-\lambda_p t} \tag{2.5}$$

The parameter λ_p characterizes the rate of change of the page p and can be estimated based on previous observations, specially if the web server provides the last modification date of the page whenever it is visited. This estimation for λ_p was obtained by Cho and Garcia-Molina [CGM03b]:

$$\lambda_p \approx \frac{(X_p - 1) - \frac{X_p}{N_p \log(1 - X_p/N_p)}}{S_p T} \tag{2.6}$$

- N_p number of visits to p.
- S_p time since the first visit to p.
- X_p number of times the server has informed that the page has changed.
- T_p total time with no modification, according to the server, summed for all the visits.

If the server does not give the last-modified time, we can still check for modifications by comparing the downloaded copies at two different times, so X_p now will be the number of times a modification is detected. The estimation for the parameter in this case is:

$$\lambda_p \approx \frac{-N_p \log(1 - X_p/N_p)}{S_p} \tag{2.7}$$

The above equation requires $X_p < N_p$, so if the page changes every time it is visited, we cannot estimate its change frequency.

2.2.2.3 Strategies

The objective of the crawler will be to keep the average freshness of pages in its collection as high as possibly, or to keep the average age of pages as low as possible, which is not equivalent. In the first case, the crawler is just concerned with *how many* pages are out-dated, while in the second case, the crawler is concerned with *how old* the local copies of pages are.

Two simple re-visiting policies were studied by Cho and Garcia-Molina [CGM03a]:

Uniform policy This involves re-visiting all pages in the collection with the same frequency, regardless of their rates of change.

Proportional policy This involves re-visiting more often the pages that change more frequently. The visiting frequency is directly proportional to the (estimated) change frequency.

In both cases, the repeated crawling order of pages can be done either at random or with a fixed order.

Cho and Garcia-Molina proved the surprising result that, in terms of average freshness, the *uniform policy* outperforms the *proportional policy* in both a simulated Web and a real Web crawl. The explanation for this result comes from the fact that, when a page changes too often, the crawler will waste time by trying to re-crawl it too fast and still will not be able to keep its copy of the page fresh. ”To improve freshness, we should penalize the elements that change too often” [CGM03a].

The optimal re-vising policy is neither the uniform policy nor the proportional policy. The optimal for keeping average freshness high includes ignoring the pages that change too often, and the optimal for keeping average age low is to use access frequencies which monotonically, sub-linearly increase with the rate of change of each page. In both cases, the optimal is closer to the uniform policy than to the proportional policy: as Coffman et al. note, “in order to minimize the expected obsolescence time, the accesses to any particular page should be kept as evenly spaced as possible” [EGC98].

Explicit formulas for the re-visit policy are not attainable in general, but they are obtained numerically, as they depend on the distribution of page changes. Note that the re-visiting policies considered here regard all pages as homogeneous in terms of quality (all pages on the Web are worth the same) which is not a realistic scenario, so further information about the Web page quality should be included to achieve a better crawling policy.

2.2.3 Politeness policy

As noted by Koster [Kos95], the use of Web robots is useful for a number of tasks, but comes with a price for the general community. The costs of using Web robots include:

- Network resources, as robots require considerable bandwidth, and operate with a high degree of parallelism during a large period of time.

- Server overload, specially if the frequency of accesses to a given server is too high.
- Poorly written robots, which can crash servers or routers, or which download pages which they cannot handle.
- Personal robots, which, if deployed by too many users, can disrupt networks and Web servers.

A partial solution to these problems is the robots exclusion protocol [Kos96], which is a standard for administrators to indicate which parts of their Web servers should not be accessed by robots. This standard does not include a suggestion for the interval of visits to the same server, which is the most effective way of avoiding server overload.

The first proposal for the interval between connections was given in [Kos93] and was 60 seconds. However, if we download pages at this rate from a Web site with more than 100,000 pages over a perfect connection with zero latency and infinite bandwidth, it would take more than 2 months to download only that entire Web site; also, we would be permanently using one of the processes or threads from that Web server. This does not seems acceptable.

Cho [CGM03b] uses 10 seconds as an interval for accesses, and the WIRE crawler [BYC02] uses 15 seconds as the default. The Mercator Web crawler [HN99] follows an adaptive politeness policy: if it took t seconds to download a document from a given sever, the crawler waits for $10 \times t$ seconds before downloading the next page.

Anecdotal evidence from access logs shows that access intervals from known crawlers vary between 20 seconds and 3–4 minutes. It is worth noticing that even when being very polite, and taking all the safeguards to avoid overloading Web servers, some complaints from Web server administrators are received. Brin and Page note that:

> "... running a crawler which connects to more than half a million servers (...) generates a fair amount of email and phone calls. Because of the vast number of people coming on line, there are always those who do not know what a crawler is, because this is the first one they have seen." [BP98].

2.2.4 Parallelization policy

A parallel crawler is a crawler which run multiple process in parallel. The goal is to maximize the download rate while minimizing the overhead from parallelization and to avoid repeated downloads of the same page.

To avoid downloading the same page more than once, the crawling system requires a policy for assigning the new URLs discovered during the crawling process, as the same URL can be found by two different crawling processes. Cho and Garcia-Molina [CGM02] studied two types of policy:

Dynamic assignment With this type of policy, a central server assigns new URLs to different crawlers dynamically. This allows the central server to, for instance, dynamically balance the load of each crawler.

With dynamic assignment, typically the systems can also add or remove downloader processes. The central server may become the bottleneck, so most of the workload must be transfered to the distributed crawling processes for large crawls.

There are two configurations of crawling architectures with dynamic assignment, which have been described by Shkapenyuk and Suel [SS02]:

- A small crawler configuration, in which there is a central DNS resolver and central queues per Web site, and distributed downloaders.
- A large crawler configuration, in which the DNS resolver and the queues are also distributed.

Static assignment With this type of policy, there is a fixed rule stated from the beginning of the crawl which defines how to assign new URLs to the crawlers.

For static assignment, a hashing function can be used to transform URLs (or, even better, complete Web site names) into a number which corresponds to the index of the corresponding crawling process. As there are external links which will go from a Web site assigned to one crawling process to a Web site assigned to a different crawling process, some exchange of URLs must occur.

To reduce the overhead due to the exchange of URLs between crawling processes, the exchange should be done in batch, several URLs at a time, and the most cited URLs in the collection should be known by all crawling processes before the crawl (e.g.: using data from a previous crawl) [CGM02].

An effective assignment function must have three main properties: each crawling process should get approximately the same number of hosts (balancing property), if the number of crawling processes grows, the number of hosts assigned to each process must shrink (contra-variance property), and the assignment must be able to add and remove crawling processes dynamically. Boldi *et al.* [BCSV02] propose to use consistent hashing - which replicates the buckets, so adding or removing a bucket does not requires re-hashing of the whole table - to achieve all of the desired properties.

2.3 Web crawler architecture

A crawler must have a good crawling strategy, as noted in the previous sections, but it also needs a highly optimized architecture. Shkapenyuk and Suel noted that:

> "While it is fairly easy to build a slow crawler that downloads a few pages per second for a short period of time, building a high-performance system that can download hundreds of millions of pages over several weeks presents a number of challenges in system designed, I/O and network efficiency, and robustness and manageability." [SS02].

Web crawlers are a central part of search engines, and details on their algorithms and architecture are kept as business secrets. When crawler designs are published, there is often an important lack of details that prevents other from reproducing the work. There are also emerging concerns about "search engine spamming", which prevent major search engines from publishing their ranking algorithms.

The typical high-level architecture of Web crawlers is shown in Figure 2.5. The following is a list of published crawler architectures for general-purpose crawlers (excluding focused Web crawlers), with a brief description which includes the names given to the different components and outstanding features:

RBSE [Eic94] was the first published Web crawler. It was based on two programs: the first program, "spider" maintains a queue in a relational database, and the second program "mite", is a modified `www` ASCII browser which downloads the pages from the Web.

WebCrawler [Pin94] was used to build the first publicly-available full-text index of a sub-set of the Web. It was based on lib-WWW to download pages, and another program to parse and order URLs for breadth-first exploration of the Web graph. It also included a real-time crawler which followed links based on the similarity of the anchor text with the provided query.

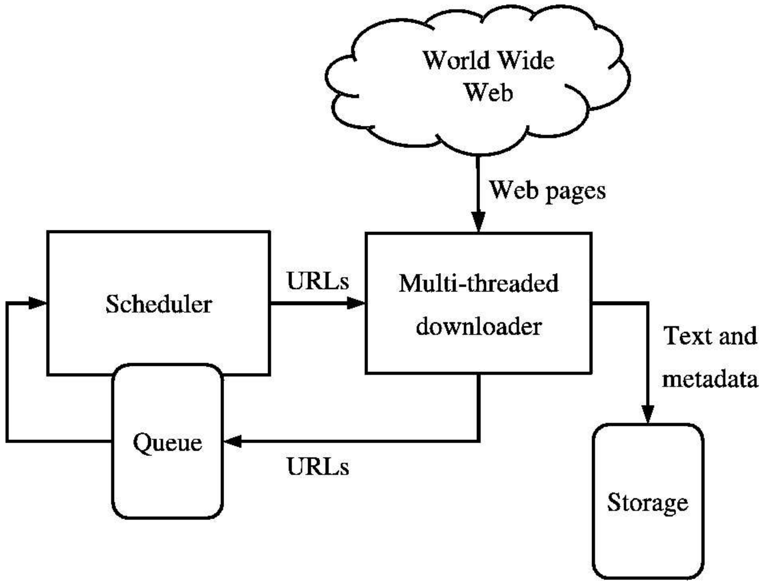

Figure 2.5: Typical high-level architecture of a Web crawler.

World Wide Web Worm [McB94] was a crawler used to build a simple index of document titles and URLs. The index could be searched by using the `grep` UNIX command.

Internet Archive Crawler [Bur97] is a crawler designed with the purpose of archiving periodic snapshots of a large portion of the Web. It uses several process in a distributed fashion, and a fixed number of Web sites are assigned to each process. The inter-process exchange of URLs is carried in batch with a long time interval between exchanges, as this is a costly process. The Internet Archive Crawler also has to deal with the problem of changing DNS records, so it keeps an historical archive of the hostname to IP mappings.

WebSPHINX [MB98] is composed of a Java class library which implements multi-threaded Web page retrieval and HTML parsing, and a graphical user interface to set the starting URLs, to extract the downloaded data and to implement a basic text-based search engine.

Google Crawler [BP98] is described in some detail, but the reference is only about an early version of its architecture, which was based in C++ and Python. The crawler was integrated with the indexing process, because text parsing was done for full-text indexing and also for URL extraction. There is an URL server which sends lists of URLs to be fetched by several crawling processes. During parsing, the URLs found were passed to a URL server which checked if the URL have been previously seen. If not, the URL was added to the queue of the URL server.

CobWeb [dSVG$^+$99] uses a central "scheduler" and a series of distributed "collectors". The collectors parse the downloaded Web pages and send the discovered URLs to the scheduler, which in turns assign them to the collectors. The scheduler enforces a breadth-first search order with a politeness policy to avoid overloading Web servers. The crawler is written in Perl.

Mercator [HN99] is a modular Web crawler written in Java. Its modularity arises from the usage of interchangeable "protocol modules" and "processing modules". Protocols modules are related to how to acquire the Web pages (e.g.: by HTTP), and processing modules are related to how to process Web pages. The standard processing module just parses the pages and extract new URLs, but other processing modules can be used to index the text of the pages, or to gather statistics from the Web.

WebFountain [EMT01] is a distributed, modular crawler similar to Mercator but written in C++. It features a "controller" machines which coordinates a series of "ants" machines. After repeatedly downloading pages, a change rate is inferred for each page and a non-linear programming method must be used to solve the equation system for maximizing freshness. The authors recommend to use this crawling order in the early stages of the crawl, and then switch to a uniform crawling order, in which all pages being visited with the same frequency.

PolyBot [SS02] is a distributed crawler written in C++ and Python, which is composed of a "crawl manager", one or more "downloaders" and one or more "DNS resolvers". Collected URLs are added to a queue on disk, and processed later to search for seen URLs in batch mode. The politeness policy considers both third and second level domains (e.g.: `www.example.com` and `www2.example.com` are third level domains) because third level domains are usually hosted by the same Web server.

WebRACE [ZYD02] is a crawling and caching module implemented in Java, and used as a part of a more generic system called eRACE. The system receives requests from users for downloading Web pages, so the crawler acts in part as a smart proxy server. The system also handles requests for "subscriptions" to Web pages which must be monitored: when the pages changes, they must be downloaded by the crawler and the subscriber must be notified. The most outstanding feature of WebRACE is that, while most crawlers start with a set of "seed" URLs, WebRACE is continuously receiving new starting URLs to crawl from.

Ubicrawler [BCSV02] is a distributed crawler written in Java, and it has no central process. It is composed of a number of identical "agents"; and the assignment function is calculated using consistent hashing of the host names. There is zero overlap, meaning that no page is crawled twice, unless a crawling agent crashes (then, another agent must re-crawl the pages from the failing agent). The crawler is designed to achieve high scalability and to tolerant to failures.

FAST Crawler [RM02] is the crawler used by the FAST search engine, and a general description of its architecture is available. It is a distributed architecture in which each machine holds a "document scheduler", which maintains a queue of documents to be downloaded by a "document processor" which are stored in a local storage subsystem. Each crawler communicates with the other crawler via a "distributor" module which exchanges hyperlink information.

WIRE [BYC02, CBY02] is the crawler developed for this research, and is described in detail in Chapter **??** of this thesis.

In addition to the specific crawler architectures listed above, there are general crawler architectures published by Cho [CGM02] and Chakrabarti [Cha03].

A few Web crawlers have been released under the GNU public license: Larbin [Ail04], WebBase [Dac02], a free version of WebSPHINX [Mil04], GRUB [gru04] and HT://Dig [htd04]. For commercial products, see [SS04, bot04].

About practical issues of building a Web crawler, which is the subject of Chapter **??**, a list of recommendations for building a search engine was written by Patterson [Pat04].

2.4 Web characterization

2.4.1 Methods for sampling

One of the main difficulties involved in any attempt of Web characterization is how to obtain a good sample. As there are very few important pages lost in a vast amount of unimportant pages (according to any metric: Pagerank, reference count, page size, etc.), just taking a URL at random is not enough. For many applications, pages with little or no meaningful content should be excluded, so estimating the importance of each page based on partial information is needed [HHMN00].

We distinguish two main methods for sampling Web pages:

Vertical sampling involves gathering pages restricted by domain names. As the domain name system induces a hierarchical structure, vertical sampling can be done at different levels of the structure. When vertical sampling is done at top-level it can select entire countries such as `.cl`, `.it`, `.au`, which are expected to be cohesive in terms of language, topics, history, or it can select general top-level domains such as `.edu` or `.com`, which are less coherent, except for the `.gov` domain. When vertical sampling is done at second level, it will choose a set of pages produced by members of the same organization (e.g. `stanford.edu`).

Countries which have been the subject of Web characterization studies include Brazil [VdMG$^+$00], Chile [BYP03], Portugal [GS03], Spain [BY03], Hungary [BCF$^+$03] and Austria [RAW$^+$02].

Horizontal sampling involves a criteria of selection which is not based on domain names. In this case, there are two approaches for gathering data: using a log of the transactions in the proxy of a large organization or ISP, or using a Web crawler. There are advantages and disadvantages for each method: when monitoring a proxy it is easy to find popular pages, but the revisit period is impossible to control, as it depends on users; using a crawler the popularity of pages has to be estimated but the revisit period can be fine-tuned.

In horizontal sampling, a "random walk" can be used to obtain a set in which pages are roughly visited with probability proportional to their Pagerank values, and then obtain a sample taken from this set with probability inversely proportional to Pagerank, so the sample is expected to be unbiased [HHMN99, HHMN00].

2.4.2 Web dynamics

There are two areas of Web dynamics: studying the Web growth and studying the document updates [RM02]; we will focus on the study of document updates, i.e.: the change of the Web in terms of creations, updates and deletions. For a model of the growth of the number of pages per Web site, see the study by Huberman and Adamic [HA99].

When studying document updates, the data is obtained by repeated access to a large set of pages during a period of time.

For each page p and each visit, the following information is available.

- The access time-stamp of the page: visit_p.
- The last-modified time-stamp (given by most web servers; about 80%-90% of the requests in practice): modified_p.

- The text of the page, which can be compared to an older copy to detect changes, specially if modified_p is not provided.

The following information can be estimated if the re-visiting period is short:

- The time at which the page first appeared: created_p.
- The time at which the page was no longer reachable: deleted_p. Koehler [Koe04] noted that pages that are unreachable may become reachable in the future, and many pages exhibit this behavior, so he prefers the term "comatose page" instead of "dead page".

In all cases, the results are only an estimation of the actual values because they are obtained by **polling** for events (changes), not by the resource **notifying** events, so it is possible that between two accesses a Web page changes more than once.

There are different time-related metrics for a Web page, the most used are:

- Age: $\text{visit}_p - \text{modified}_p$.
- Lifespan: $\text{deleted}_\text{p} - \text{created}_p$.
- Number of changes during the lifespan: changes_p.
- Average change interval: $\text{lifespan}_p/\text{changes}_p$.

Once an estimation of the above values has been obtained for Web pages in the sample, useful metrics for the entire sample are calculated, for instance:

- Distribution of change intervals.
- Average lifespan of pages.
- Median lifespan of pages, i.e.: time it takes for 50% of the pages to change. This is also called the "half-life" of the Web.

Selected results about Web page changes are summarized in Table 2.1.

The methods for the studies of these parameters vary widely. Some researches focus on the lifespan of pages, as they are concerned with the "availability" of Web content. This is an important subject from the point of view of researchers, as it is being common to cite on-line publications as sources, and they are expected to be somewhat "permanent" (but they are not).

Other publications focus on the rate of change of pages, which is more directly related to Web crawling, as knowing the rate of change can help to produce a good re-visiting order.

2.4.3 Link structure

2.4.3.1 Scale-free networks

Scale-free networks, as opposed to random networks, are characterized by an uneven distribution of links. These networks have been the subject of a series of studies by Barabási [Bar02], and are characterized as networks in which the distribution of links follow a power law:

Reference	Sample	Observations
[Koe04]	360 random pages, long-term study	Half-life $\approx$ 2 years 33% of pages lasted for 6 years
[MB03]	500 scholarly publications	Half-life $\approx$ 4.5 years
[GS96]	2,500 pages, university Website	Average lifespan $\approx$ 50 days Median age $\approx$ 150 days
[Spi03]	4,200 scholarly publications	Half-life $\approx$ 4 years
[DFKM97]	950,000 pages	Average age between 10 days and 10 months Highly-linked pages change more frequently
[Cho00]	720,000 pages, popular sites	Average lifespan $\approx$ 60 – 240 days 40% of pages in `.com` change every day 50% of pages in `.edu` and `.gov` remain the same for 4 months
[NCO04]	4 million pages, popular sites	8% of new pages every week 62% of the new pages have novel content 25% of new links every week 80% of page changes are minor (less than 0.2 in cosine distance)
[FMNW03]	150 million pages,	65% of pages don't change in a 10-week period 30% of pages have only minor changes Large variations of availability across domains
[BCS$^+$00]	800 million pages	Average lifespan $\approx$ 140 days

Table 2.1: Summary of selected results about Web page changes, ordered by increasing sample size.

$$P(\Gamma(p) = k) \propto k^{-\theta} \tag{2.8}$$

A scale-free network is characterized by a few highly-linked nodes which act as "hubs" connecting several nodes to the network. The difference between a random network and a scale-free network is depicted in Figure 2.6.

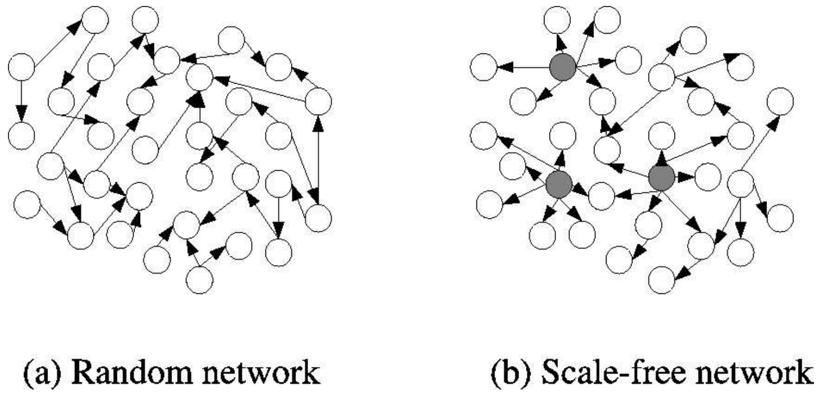

Figure 2.6: Examples of a random network and a scale-free network. Each graph has 32 nodes and 32 links. Note that both were chosen to be connected and as planar as possible for better legibility, so they are not entirely random.

Scale-free networks arise in a wide variety of contexts, and there is a substantial amount of literature about them, so we will cite in the following just a few selected publications.

Some examples of scale-free network arising outside the realm of computer networks include:

- Acquaintances, friends and social popularity in human interactions. The Economist commented "in other words, some people have all the luck, while others have none." [Eco02].
- Sexual partners in humans, which is highly relevant for the control of sexually-transmitted diseases.
- Power grid designs, which most of them designed in such a way that if a few key nodes fail, the entire system goes down.
- Collaboration of movie actors in films.
- Citations in scientific publications.
- Proteins interaction.
- Cellular metabolism.

About computer networks, Barabási noted: "While entirely of human design, the emerging network appears to have more in common with a cell or an ecological system than with a Swiss watch." [Bar01]. Examples of scale-free networks related to the Internet are:

- Geographic connectivity of Internet nodes.
- Number of links on Web pages.

- User participation in interest groups and communities.
- E-mail exchanges.

These scale-free networks do not arise by chance alone. Erdõs and Rényi [ER60] studied a model of growth for graphs in which, at each step, two nodes are chosen uniformly at random and a link is inserted between them. The properties of these random graphs are not consistent with the properties observed in scale-free networks, and therefore a model for this growth process is needed.

The connectivity distribution over the entire Web is very close to a power law, because there are a few Web sites with huge numbers of links, which benefit from a good placement in search engines and a established presence on the Web. This have been called the "winners take all" phenomenon.

Barabási and Albert [BA99] propose a "rich get richer" generative model in which each new Web page creates link to existent Web pages with a probability distribution with is not uniform, but proportional to the current in-degree of Web pages. According to this process, a page with many in-links will attract more in-links that a regular page.

A different generative model is the "copy" model studied by Kumar *et al.* [KRR+00], in which new nodes choose an existent node at random and copy a fraction of the links of the existent node. This also generates a power law.

However, if we look at communities of interests in a specific topic, discarding the major hubs of the Web, the distribution of links is no longer a power law but resembles more a Gaussian distribution, as observed by Pennock *et al.* [PFL+02] in the communities of the home pages of universities, public companies, newspapers and scientists. Based on these observations, the authors propose a generative model which mixes preferential attachment with a baseline probability of gaining a link.

2.4.3.2 Macroscopic structure

The most complete study of the Web structure [BKM+00] focus on the connectivity of a subset of 200 million Web pages from the Altavista search engine. This subset is a connected graph, if we ignore the direction of the links.

The study starts by identifying in the Web graph a single large strongly connected component (i.e.: all of the pages in this component can reach one another along directed links). They call the larger strongly connected component "MAIN". Starting in MAIN, if we follow links forward we find OUT, and if we follow links backwards we find IN. All of the Web pages with are part of the graph but do not fit neither MAIN, IN, nor OUT are part of a fourth component called TENTACLES.

A page can describe several documents and one document can be stored in several pages, so we decided to study the structure of how websites were connected, as websites are closer to real logical units. Not surprisingly, we found in [BYC01] that the structure in the domain at the website level was similar to the global Web – another example of the autosimilarity of the Web – and hence we use the same notation of [BKM+00]. The components are defined as follows:

(a) MAIN, sites that are in the strong connected component of the connectivity graph of sites (that is, we can navigate from any site to any other site in the same component);

(b) IN, sites that can reach MAIN but cannot be reached from MAIN;

(c) OUT, sites that can be reached from MAIN, but there is no path to go back to MAIN; and

(d) other sites that can be reached from IN or can only reach OUT (TENTACLES), sites in paths between IN and OUT (TUNNEL), and unconnected sites (ISLANDS).

Figure 2.7 shows all these components.

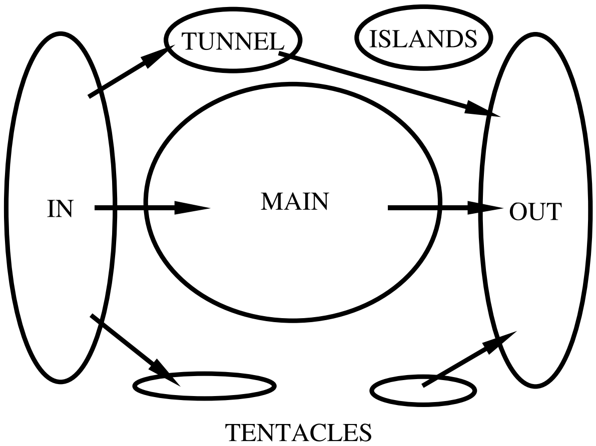

Figure 2.7: Macroscopic structure of the Web.

2.4.4 User sessions on the Web

User sessions on the Web are usually characterized through models of random surfers, such as the ones studied by Diligenti et al. [DGM04]. As we have seen, these models have been used for page ranking with the Pagerank algorithm [PBMW98], or to sample the web [HHMN00].

The most used source for data about the browsing activities of users are the access log files of Web servers, and there are a number of log file analysis software available: [Tur04, web04, Bou04, Bar04]. A common goal for researchers in this area is to to infer rules in user browsing patterns, such as "40% users that visit page *A* also visit page *B*" to assist in Web site re-design. Log file analysis has a number of restrictions arising from the implementation of HTTP, specially caching and proxies, as noted by Haigh and Megarity [HM98]. *Caching* implies that re-visiting a page is not always recorded, and re-visiting pages is a common action, and can account for more than 50% of the activity of users, when measuring it directly in the browser [TG97]. *Proxies* implies that several users can be accessing a Web site from the same IP address.

To process log file data, careful data preparation must be done [CMS99, BS00, TT04]. An important aspect of this data preparation is to separate automated sessions from user sessions; robot session characterization was studied by Tan and Kumar [TK01].

The visits to a Web site have been modeled as a sequence of decisions by Huberman *et. al* [HPPL98, AH00]; they obtain a model for the number of clicks that follows a Zipf's law. Levene *et al.* [LBL01] proposed to use an absorbing state to represent the user leaving the Web site, and analyzed the lengths of user sessions when the probability of following a link increases with session length. Lukose and Huberman [LH98] also present an analysis of the Markov chain model of a user clicking through a Web site, and focus in designing an algorithm for automatic browsing, which is also the topic of a recent work by Liu *et al.* [LZY04].

2.5 Conclusions

In this chapter, we have surveyed selected publications from the related work which is relevant for this thesis. The topic of link analysis is an active research topic and there are some established methods in the Web Information Retrieval community. The topic of Web crawling design has less research available, and is also affected by business secrecy and concerns about search engine spamming.

The next chapter starts the main part of this thesis by presenting a new crawling model and architecture.

Bibliography

[ACGM+01] Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, and Sriram Raghavan. Searching the Web. *ACM Transactions on Internet Technology (TOIT)*, 1(1):2–43, August 2001.

[AH00] Eytan Adar and Bernardo A. Huberman. The economics of web surfing. In *Poster Proceedings of the Ninth Conference on World Wide Web*, Amsterdam, Netherlands, May 2000.

[Ail04] Sebastien Ailleret. Larbin, a multi-purpose web crawler. http://larbin.sourceforge.net/index-eng.html, 2004. GPL software.

[BA99] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, October 1999.

[Bar01] Albert-László Barabási. The physics of the web. *PhysicsWeb.ORG, online journal*, July 2001.

[Bar02] Albert-László Barabási. *Linked: the new science of networks*. Perseus Publishing, 2002.

[Bar04] Bradford L. Barrett. WebAlizer: log file analysis program. http://www.mrunix.net/webalizer/, 2004.

[BCF+03] András A. Benczúr, Károly Csalogány, Dániel Fogaras, Eszter Friedman, Tamás Sarlós, Máté Uher, and Eszter Windhager. Searching a small national domain – a preliminary report. In *Poster Proceedings of Conference on World Wide Web*, Budapest, Hungary, May 2003.

[BCS+00] Brian Brewington, George Cybenko, Raymie Stata, Krishna Bharat, and Farzin Maghoul. How dynamic is the web? In *Proceedings of the Ninth Conference on World Wide Web*, pages 257 – 276, Amsterdam, Netherlands, May 2000.

[BCSV02] Paolo Boldi, Bruno Codenotti, Massimo Santini, and Sebastiano Vigna. Ubicrawler: A scalable fully distributed web crawler. In *Proceedings of the eight Australian World Wide Web Conference (AusWeb)*, 2002.

[BH98] Krishna Bharat and Monika R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 104–111, Melbourne, Australia, August 1998. ACM Press.

[BKM+00] Andrei Broder, Ravi Kumar, Farzin Maghoul, Prabhakar Raghavan, Sridhar Rajagopalan, Raymie Stata, Andrew Tomkins, and Janet Wiener. Graph structure in the web: Experiments and models. In *Proceedings of the Ninth Conference on World Wide Web*, pages 309–320, Amsterdam, Netherlands, May 2000.

[bot04] Botspot. http://www.botspot.com/, 2004.

[Bou04] Thomas Boutell. WUsage: Web log analysis software. http://www.boutell.com/wusage/, 2004.

[BP98] Sergei Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, April 1998.

[Bro03] Terrence A. Brooks. Web search: how the Web has changed information retrieval. *Information Research*, 8(3):(paper no. 154), April 2003.

[BS00] Bettina Berendt and Myra Spiliopoulou. Analysis of navigation behaviour in web sites integrating multiple information systems. *The VLDB journal*, (9):56–75, 2000.

[Bur97] Mike Burner. Crawling towards eternity - building an archive of the world wide web. *Web Techniques*, 2(5), May 1997.

[BY03] Ricardo Baeza-Yates. The Web of Spain. *UPGRADE*, 3(3):82–84, 2003.

[BYC01] Ricardo Baeza-Yates and Carlos Castillo. Relating web characteristics with link based web page ranking. In *Proceedings of String Processing and Information Retrieval*, pages 21–32, Laguna San Rafael, Chile, November 2001. IEEE Cs. Press.

[BYC02] Ricardo Baeza-Yates and Carlos Castillo. Balancing volume, quality and freshness in web crawling. In *Soft Computing Systems - Design, Management and Applications*, pages 565–572. IOS Press, 2002.

[BYCSJ04] Ricardo Baeza-Yates, Carlos Castillo, and Felipe Saint-Jean. *Web Dynamics*, chapter Web Dynamics, Structure and Page Quality. Springer, 2004. To appear.

[BYP03] Ricardo Baeza-Yates and Bárbara Poblete. Evolution of the chilean web structure composition. In *Proceedings of Latin American Web Conference*, pages 11–13, Santiago, Chile, 2003. IEEE Cs. Press.

[BYRN00] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press / Addison Wesley, 2000.

[CBY02] Carlos Castillo and Ricardo Baeza-Yates. A new crawling model. In *Poster proceedings of the eleventh conference on World Wide Web*, Honolulu, Hawaii, USA, May 2002. (Extended Poster).

[CDR+98] Soumen Chakrabarti, Byron Dom, Prabhakar Raghavan, Sridhar Rajagopalan, David Gibson, and Jon Kleinberg. Automatic resource compilation by analyzing hyperlink structure and associated text. In *World Wide Web Conference*, pages 65–74, Brisbane, Australia, 1998. Elsevier.

[CGM00] Junghoo Cho and Hector Garcia-Molina. Synchronizing a database to improve freshness. In *Proceedings of ACM International Conference on Management of Data (SIGMOD)*, pages 117–128, May 2000.

[CGM02] Junghoo Cho and Hector Garcia-Molina. Parallel crawlers. In *Proceedings of the eleventh international conference on World Wide Web*, pages 124–135, Honolulu, Hawaii, USA, May 2002. ACM Press.

[CGM03a] Junghoo Cho and Hector Garcia-Molina. Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4), December 2003.

[CGM03b] Junghoo Cho and Hector Garcia-Molina. Estimating frequency of change. *ACM Transactions on Internet Technology*, 3(3), August 2003.

[CGMP98] Junghoo Cho, Hector García-Molina, and Lawrence Page. Efficient crawling through URL ordering. In *Proceedings of the seventh conference on World Wide Web*, Brisbane, Australia, April 1998.

[Cha03] Soumen Chakrabarti. *Mining the Web.* Morgan Kaufmann, 2003.

[Cho00] Junghoo Cho. The evolution of the web and implications for an incremental crawler. In *Proceedings of 26th International Conference on Very Large Databases (VLDB)*, pages 527–534, Cairo, Egypt, September 2000. Morgan Kaufmann.

[CK97] S. Jeromy Carriére and Rick Kazman. Webquery: searching and visualizing the web through connectivity. *Computer Networks and ISDN Systems*, 29(8-13):1257–1267, September 1997.

[CMS99] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, 1999.

[CvD99] Soumen Chakrabarti, Martin van den Berg, and Byron Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 31(11–16):1623–1640, 1999.

[Dac02] Lois Dacharay. Webbase web crawler. www.freesoftware.fsf.org/webbase/, 2002. GPL Software.

[DCL+00] Michelangelo Diligenti, Frans Coetzee, Steve Lawrence, C. Lee Giles, and Marco Gori. Focused crawling using context graphs. In *Proceedings of 26th International Conference on Very Large Databases (VLDB)*, pages 527–534, Cairo, Egypt, September 2000.

[DFKM97] Fred Douglis, Anja Feldmann, Balachander Krishnamurthy, and Jeffrey C. Mogul. Rate of change and other metrics: a live study of the world wide web. In *USENIX Symposium on Internet Technologies and Systems*, pages 147–158, Monterey, California, USA, December 1997.

[DGM04] Michelangelo Diligenti, Marco Gori, and Marco Maggini. A unified probabilistis framework for web page scoring systems. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):4–16, 2004.

[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 Cs. Press.

[Eco02] The Economist. What does the internet look like? *The Economist*, October 2002.

[EGC98] R. Weber Edward G. Coffman, Z. Liu. Optimal robot scheduling for web search engines. *Journal of Scheduling*, 1(1):15–29, 1998.

[Eic94] D. Eichmann. The RBSE spider: balancing effective search against web load. In *Proceedings of the first World Wide Web Conference*, Geneva, Switzerland, May 1994.

[EMT01] Jenny Edwards, Kevin S. McCurley, and John A. Tomlin. An adaptive model for optimizing performance of an incremental web crawler. In *Proceedings of the Tenth Conference on World Wide Web*, pages 106–113, Hong Kong, May 2001. Elsevier.

[ER60] Paul Erdõs and Alfred Rényi. Random graphs. *Publication of the Mathematical Institute of the Hungarian Acadamy of Science*, 5:17 – 61, 1960.

[FMNW03] Dennis Fetterly, Mark Manasse, Marc Najork, and Janet L. Wiener. A large-scale study of the evolution of web pages. In *Proceedings of the Twelfth Conference on World Wide Web*, pages 669 – 678, Budapest, Hungary, May 2003. ACM Press.

[goo04] Google search engine. http://www.google.com/, 2004.

[gru04] Grub, a distributed crawling project. http://www.grub.org, 2004. GPL software.

[GS96] James Gwertzman and Margo Seltzer. World-wide web cache consistency. In *Proceedings of the 1996 Usenix Technical Conference*, San Diego, California, USA, January 1996.

[GS03] Daniel Gomes and Mrio J. Silva. A characterization of the portuguese web. In *Proceedings of 3rd ECDL Workshop on Web Archives*, Trondheim, Norway, August 2003.

[HA99] Bernardo A. Huberman and Lada A. Adamic. Evolutionary dynamics of the World Wide Web. *Condensed Matter*, January 1999. (paper 9901071).

[Hav02] Taher H. Haveliwala. Topic-sensitive pagerank. In *Proceedings of the Eleventh World Wide Web Conference*, pages 517–526, Honolulu, Hawaii, USA, May 2002. ACM Press.

[Hen01] Monika Henzinger. Hyperlink analysis for the web. *IEEE Internet Computing*, 5(1):45–50, 2001.

[HHMN99] Monika R. Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. Measuring index quality using random walks on the Web. *Computer Networks*, 31(11–16):1291–1303, 1999.

[HHMN00] Monika Henzinger, Allan Heydon, Michael Mitzenmacher, and Marc Najork. On near–uniform url sampling. In *Proceedings of the Ninth Conference on World Wide Web*, pages 295–308, Amsterdam, Netherlands, May 2000. Elsevier.

[HM98] Susan Haigh and Janette Megarity. Measuring web site usage: Log file analysis. *Network Notes*, (57), 1998.

[HN99] Allan Heydon and Marc Najork. Mercator: A scalable, extensible web crawler. *World Wide Web Conference*, 2(4):219–229, April 1999.

[HPPL98] Bernardo A. Huberman, Peter L. T. Pirolli, James E. Pitkow, and Rajan M. Lukose. Strong regularities in world wide web surfing. *Science*, 280(5360):95–97, April 1998.

[htd04] HT://Dig. http://www.htdig.org/, 2004. GPL software.

[Kle99] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604–632, 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.

[Kos95] Martijn Koster. Robots in the web: threat or treat ? *ConneXions*, 9(4), April 1995.

[Kos96] Martijn Koster. A standard for robot exclusion. http://www.robotstxt.org/wc/exclusion.html, 1996.

[KRR$^+$00] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 57–65. IEEE Cs. Press, 2000.

[LBL01] Mark Levene, Jose Borges, and George Loizou. Zipf's law for web surfers. *Knowledge and Information Systems*, 3(1):120–129, 2001.

[LG00] Steve Lawrence and C. Lee Giles. Accessibility of information on the web. *Intelligence*, 11(1):32–39, 2000.

[LH98] Rajan M. Lukose and Bernardo A. Huberman. Surfing as a real option. In *Proceedings of the first international conference on Information and computation economies*, pages 45–51. ACM Press, 1998.

[Li98] Yanhong Li. Toward a qualitative search engine. *IEEE Internet Computing*, pages 24 – 29, July 1998.

[LWP$^+$01] Lipyeow Lim, Min Wang, Sriram Padmanabhan, Jeffrey Scott Vitter, and Ramesh Agarwal. Characterizing Web document change. In *Proceedings of the Second International Conference on Advances in Web-Age Information Management*, volume 2118 of *Lecture Notes in Computer Science*, pages 133–144, London, UK, July 2001. Springer.

[LZY04] Jiming Liu, Shiwu Zhang, and Jie Yang. Characterizing web usage regularities with information foraging agents. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):566 – 584, 2004.

[MB98] Robert Miller and Krishna Bharat. Sphinx: A framework for creating personal, site-specific web crawlers. In *Proceedings of the seventh conference on World Wide Web*, Brisbane, Australia, April 1998.

[MB03] John Markwell and David W. Brooks. Link-rot limits the usefulness of Web-based educational materials in biochemistry and molecular biology. *Biochem. Mol. Biol. Educ.*, 31:69–72, 2003.

[McB94] Oliver A. McBryan. GENVL and WWWW: Tools for taming the web. In *Proceedings of the first World Wide Web Conference*, Geneva, Switzerland, May 1994.

[Mil04] Rob Miller. Websphinx, a personal, customizable web crawler. http://www-2.cs.cmu.edu/ rcm/websphinx, 2004. Apache-style licensed, open source software.

[NCO04] Alexandros Ntoulas, Junghoo Cho, and Christopher Olston. What's new on the web?: the evolution of the web from a search engine perspective. In *Proceedings of the 13th conference on World Wide Web*, pages 1 – 12, New York, NY, USA, May 2004. ACM Press.

[NW01] Marc Najork and Janet L. Wiener. Breadth-first crawling yields high-quality pages. In *Proceedings of the Tenth Conference on World Wide Web*, pages 114–118, Hong Kong, May 2001. Elsevier.

[Pat04] Anna Patterson. Why writing your own search engine is hard. *ACM Queue*, pages 49 – 53, April 2004.

[PBMW98] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation algorithm: bringing order to the web. In *Proceedings of the seventh conference on World Wide Web*, Brisbane, Australia, April 1998.

[PFL+02] David M. Pennock, Gary W. Flake, Steve Lawrence, Eric J. Glover, and C. Lee Giles. Winners don't take all: Characterizing the competition for links on the web. *Proceedings of the National Academy of Sciences*, 99(8):5207–5211, April 2002.

[Pin94] Brian Pinkerton. Finding what people want: Experiences with the WebCrawler. In *Proceedings of the first World Wide Web Conference*, Geneva, Switzerland, May 1994.

[RAW+02] Andreas Rauber, Andreas Aschenbrenner, Oliver Witvoet, Robert M. Bruckner, and Max Kaiser. Uncovering information hidden in web archives. *D-Lib Magazine*, 8(12), 2002.

[RM02] Knut Magne Risvik and Rolf Michelsen. Search engines and web dynamics. *Computer Networks*, 39(3), June 2002.

[SB88] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523, 1988.

[Spi03] Diomidis Spinellis. The decay and failures of web references. *Communications of the ACM*, 46(1):71–77, January 2003.

[SS02] Vladislav Shkapenyuk and Torsten Suel. Design and implementation of a high-performance distributed web crawler. In *Proceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 357 – 368, San Jose, California, February 2002. IEEE Cs. Press.

[SS04] Danny Sullivan and Chris Sherman. Search Engine Watch reports. http://www.searchenginewatch.com/reports/, 2004.

[TG97] Linda Tauscher and Saul Greenberg. Revisitation patterns in world wide web navigation. In *Proceedings of the Conference on Human Factors in Computing Systems CHI'97*, 1997.

[TK01] P.N. Tan and V. Kumar. Discovery of web robots session based on their navigational patterns. *Data Mining and Knowledge discovery*, 2001.

[Tom03] John A. Tomlin. A new paradigm for ranking pages on the world wide web. In *Proceedings of the Twelfth Conference on World Wide Web*, pages 350–355, Budapest, Hungary, May 2003. ACM Press.

[TT04] Doru Tanasa and Brigitte Trousse. Advanced data preprocessing for intersites Web usage mining. *IEEE Intelligent Systems*, 19(2):59–65, 2004.

[Tur04] Stephen Turner. Analog: WWW log file analysis. http://www.analog.cx/, 2004.

[VdMG+00] Eveline A. Veloso, Edleno de Moura, P. Golgher, A. da Silva, R. Almeida, A. Laender, B. Ribeiro-Neto, and Nivio Ziviani. Um retrato da web brasileira. In *Proceedings of Simposio Brasileiro de Computacao*, Curitiba, Brasil, July 2000.

[web04] WebTrends corporation. http://www.webtrends.com/, 2004.

[YL96] Budi Yuwono and Dik Lun Lee. Search and ranking algorithms for locating resources on the world wide web. In *Proceedings of the twelfth International Conference on Data Engineering (ICDE)*, pages 164–171, Washington, DC, USA, February 1996. IEEE Cs. Press.

[ZYD02] Demetrios Zeinalipour-Yazti and Marios D. Dikaiakos. Design and implementation of a distributed crawler and filtering processor. In *Proceedings of the fifth Next Generation Information Technologies and Systems (NGITS)*, volume 2382 of *Lecture Notes in Computer Science*, pages 58–74, Caesarea, Israel, June 2002. Springer.