International Journal of Scientific Engineering and Research (IJSER)
www.ijser.in
ISSN (Online): 2347-3878, Impact Factor (2014): 3.05

Study of a Web Crawler

Pallavi[1], Rajiv Sharma[2]

[1]M.Tech Student, Shri Baba Mast Nath Engineering College, Maharshi Dayanand University, Rohtak, Haryana, India

[2]Assistant Professor, Department of CSE, Shri Baba Mast Nath Engineering College, Rohtak, Haryana, India

Abstract: ***This is a study of the science and practice of web crawling. Web crawling may appear to be merely an application of breadth-first-search, the truth is that there are many challenges ranging from systems concerns such as managing very large data structures to theoretical questions such as how often to revisit evolving content sources.***

Keywords: Webcrawler, URL, Search Engine**,** Crawling Policies.

1. Introduction

A web crawler (also known as a robot or a spider) is a system for the bulk downloading of web pages. Web crawlers are used for a number of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. A related use is web archiving (a service provided by e.g., the Internet archive), where large sets of web pages are periodically collected and archived for posterity. A third use is web data mining, where web pages are analyzed for statistical properties, or where data analytics is performed on them (an example would be Attributor , a company that monitors the web for copyright and trademark infringements).

2. Crawler Architecture

It describes the general architecture and key design points of modern scalable crawlers.

2.1 Architecture Overview

Figure 2.1 shows the high-level architecture of a prototypical distributed web crawler. The crawler consists of multiple processes running on different machines connected by a high-speed network. Each crawling process consists of multiple worker threads, and each worker thread performs repeated work cycles. At the beginning of each work cycle, a worker obtains a URL from the Frontier data structure, which dispenses URLs according to their priority and to politeness policies. After that the worker thread invokes the HTTP fetcher. The fetcher first calls a DNS sub-module to resolve the host component of the URL into the IP address of the corresponding web server (using cached results of prior resolutions if possible), and then connects to the web server, checks for any robots exclusion rules (which are cached as well), and attempts to download the web page. If the download succeeds, the web page may or may not be stored in a repository of harvested web pages (not shown). In either case, the page is passed to the link extractor, that parses the page’s HTML content and extracts hyperlinks contained therein. The corresponding URLs are then passed to a URL distributor, which assigns each URL to a crawling process. This assignment is typically made by hashing the URLs host component, its domain, or its IP address (the latter requires additional DNS resolutions). Since most hyperlinks refer to pages on the same web site, assignment to the local crawling process is the common case. Next, the URL passes through the Custom URL filter (e.g., to exclude URLs belonging to “black-listed” sites, or URLs with particular file extensions that are not of interest) and into the Duplicate URL eliminator, which maintains the set of all URLs discovered so far and passes on only never-before-seen URLs. Finally, the URL prioritizer selects a position for the URL in the Frontier, based on factors such as estimated page importance or rate of change.

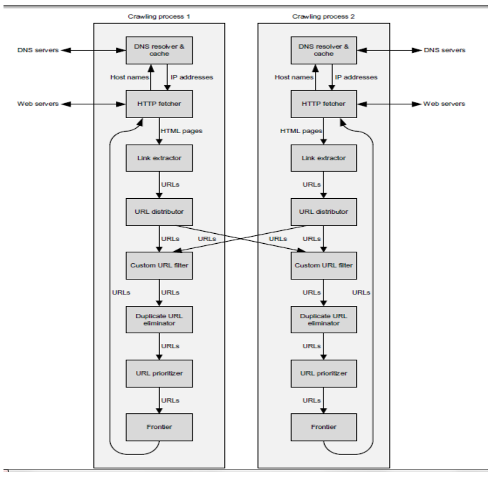

Figure 2.1: Basic crawler architecture.

2.2 Key Design Points

Web crawlers download web pages by starting from one or more seed URLs, downloading each of the associated pages, extracting the hyperlink URLs contained therein, and recursively downloading those pages. Therefore, any web crawler needs to keep track both of the URLs that are to be downloaded, as well as those that have already been downloaded (to avoid unintentional downloading the same page). The required state is a set of URLs, each associated with a flag indicating whether the page has been downloaded. The operations that must be supported are: Adding a new URL, retrieving a URL, marking a URL as downloaded, and testing whether the set contains a URL. There are many alternative in-memory data structures (e.g., trees or sorted lists) that support these operations. However, such an implementation does not scale to web corpus sizes that exceed the amount of memory available on a single machine.

3. Crawl Ordering Problem

The crawl order is extremely significant, because for the purpose of crawling the web can be considered infinite, especially due to dynamically generated content. Indeed, despite their impressive capacity, modern commercial search engines only index (and likely only crawl) a fraction of discoverable web pages. The crawler ordering question is even more crucial for the countless smaller scale crawlers that perform scoped crawling of targeted subsets of the web a good crawler order, with a focus on two basic considerations:

- **Coverage.** The fraction of desired pages that the crawler acquires successfully.
- **Freshness.** The degree to which the acquired page snapshots remain up-to-date, relative to the current “live” web copies.

3.1 Model

In the model, all pages require the same amount of time to download; the constant rate of page downloading is called the crawl rate, typically measured in pages/second. The crawl rate is not relevant to batch crawl ordering methods, but it is a key factor when scheduling page revisitations in incremental crawling.

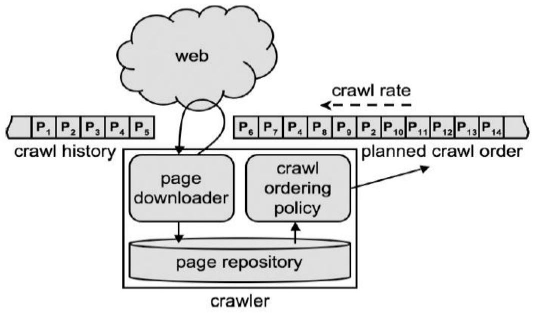

Figure 3.1: Model

Pages downloaded by the crawler are stored in a repository. The future crawl order is determined, at least in part, by analyzing the repository. For example, one simple policy mentioned earlier, breadthfirst search, extracts hyperlinks from pages entering the repository, identifies linked-to pages that are not already part of the (historical or planned) crawl order, and adds them to the end of the planned crawl order.

3.2 Limitations

This model has led to a good deal of research with practical implications. However, as with all models, it simplifies reality. For one thing, a large-scale crawler maintains its frontier data structure on disk, which limits opportunities for reordering. Generally speaking, the approach of maintaining a prioritized ensemble of FIFO queues can be used to approximate a desired crawl order.

4. Avoiding Problematic and Undesirable Content

This section discusses detection and avoidance of content that is redundant, wasteful or misleading.

4.1 Redundant Content

There is a prevalence of duplicate and near-duplicate content on the web. Shingling is a standard way to identify near-duplicate pages, but shingling is performed on web page content, and thus requires these pages to have been crawled. As such, it does not help to reduce the load on the crawler; however, it can be used to limit and diversify the set of search results presented to a user. Some duplication stems from the fact that many web sites allow multiple URLs to refer to the same content, or content that is identical modulo ever-changing elements such as rotating banner ads, evolving comments by readers, and timestamps. Another source of duplication is mirroring. Providing all or parts of the same web site on different hosts. Mirrored web sites in turn can be divided into two groups: Sites that are mirrored by the same organization, for example by having one web server serving multiple domains with the same content, or having multiple web servers provide synchronized content), and content that is mirrored by multiple organizations (for example, schools providing Unix man pages on the web, or web sites republishing Wikipedia content, often somewhat reformatted). Detecting mirrored content differs from detecting DUST in two ways: On the one hand, with mirroring the duplication occurs across multiple sites, so mirror detection algorithms have to consider the entire corpus. On the other hand, entire trees of URLs are mirrored, so detection algorithms can use URL trees (suitably compacted e.g., through hashing) as a feature to detect mirror candidates, and then compare the content of candidate subtrees (for example via shingling).

4.2 Crawler Traps

Another phenomenon that inflates the corpus without adding utility is crawler traps: Web sites that populate a large, possibly infinite URL space on that site with mechanically generated content. Some crawler traps are non-malicious, for example web-based calendaring tools that generate a page for every month of every year, with a hyperlink from each month to the next (and previous) month, thereby forming an unbounded chain of dynamically generated pages. Other crawler traps are malicious, often set up by “spammers” to inject large amounts of their content into a search engine, in the hope of having their content show up high in search result pages.

4.3 Web Spam

Web spam may be defined as “web pages that are crafted for the sole purpose of increasing the ranking of these or some affiliated pages, without improving the utility to the viewer” Web spam is motivated by the monetary value of achieving a prominent position in search engine result pages. There is a multi-billion dollar industry devoted to search engine optimization (SEO), most of it being legitimate but some of it misleading. Web spam can be broadly classified into three categories: Keyword stuffing, populating pages with highly searched or highly monetizable terms; link spam, creating cliques of tightly interlinked web pages with the goal of biasing link-based ranking algorithms such as PageRank; and cloaking, serving substantially different content to web crawlers than to human visitors (to get search referrals for queries on a topic not covered by the page). Over the past few years, many heuristics have been proposed to identify spam web pages and sites, see for example the series of AIRweb workshops [7]. The problem of identifying web spam can be framed as a classification problem, and there are many well-known classification approaches (e.g., decision trees, Bayesian classifiers, support vector machines).

5. Deep Web Crawling

The deep web crawling problem is closely related to the problem known as federated search or distributed information retrieval, in which a mediator forwards user queries to multiple searchable collections, and combines the results before presenting them to the user.

5.1 Problem Overview

Deep web crawling has three steps:
(1) **Locate deep web content sources.** A human or crawler must identify web sites containing form interfaces that lead to deep web content.

(2) **Select relevant sources.** For a scoped deep web crawling task (e.g., crawling medical articles), one must select a relevant subset of the available content sources. In the unstructured case this problem is known as database or resource selection. The first step in resource selection is to model the content available at a particular deep web site,e.g., using query-based sampling.

(3) **Extract underlying content.** Finally, a crawler must extract the content lying behind the form interfaces of the selected content sources

6. Future Work- Conclusion

As this study indicates, crawling is a well-studied problem. However, there are at least as many open questions as there are resolved ones. Even in the material we have covered, the reader has likely noticed many open issues, including:

1) **Parameter tuning.** Many of the crawl ordering policies rely on carefully tuned parameters, with little insight or science into how best to choose the parameters. For example, what is the optimal level of greediness for a scoped crawler
2) **Retiring unwanted pages.** Given finite storage capacity, in practice crawlers discard or retire low-quality and spam pages from their collections, to make room for superior pages.
3) However, we are not aware of any literature that explicitly studies retirement policies. There is also the issue of how much metadata to retain about retired pages, to avoid accidentally
4) rediscovering them, and to help assess the quality of related pages (e.g., pages on the same site, or pages linking to or linked from the retired page).
5) **Holistic crawl ordering.** Whereas much attention has been paid to various crawl ordering sub-problems (e.g., prioritizing the crawl order of new pages, refreshing content from old pages, revisiting pages to discover new links), there is little work on how to integrate the disparate approaches into a unified strategy.
6) **Deep web.** Clearly, the science and practice of deep web crawling is in its infancy. There are also several nearly untouched directions.
7) **Crawling scripts.** Increasingly, web sites employ scripts (e.g., JavaScript, AJAX) to generate content and links on the fly. Almost no attention has been paid to whether or how to crawl these sites.
8) **Personalized content.**Web sites often customize their content to individual users, e.g., Amazon gives personalized recommendations based on a user's browsing and purchasing
9) patterns. It is not clear how crawlers should treat such sites, e.g., emulating a generic user versus attempting to specialize the crawl based on different user profiles. A search engine that aims to personalize search results may wish to push some degree of personalization into the crawler.
10) Collaboration between content providers and crawlers. Crawling is a pull mechanism for discovering and acquiring content. Modern commercial crawlers employ a hybrid of push and pull, but there is little academic study of this practice and the issues involved.

References

[1] S. Abiteboul, M. Preda, and G. Cobena, "Adaptive on-line page importance computation," in Proceedings of the 12th International World Wide Web Conference, 2003.
[2] E. Adar, J. Teevan, S. T. Dumais, and J. L. Elsas, "The web changes everything: Understanding the dynamics of web content," in Proceedings of the 2nd International Conference on Web Search and Data Mining, 2009.
[3] Advanced Triage (medical term), http://en.wikipedia.org/wiki/Triage# Advanced triage.
[4] A. Agarwal, H. S. Koppula, K. P. Leela, K. P. Chitrapura, S. Garg, P. K. GM, C. Haty, A. Roy, and A. Sasturkar, "URL normalization for de-duplication of web pages," in Proceedings of the 18th Conference on Information and Knowledge Management, 2009.
[5] C. C. Aggarwal, F. Al-Garawi, and P. S. Yu, "Intelligent crawling on the world wide web with arbitrary predicates," in Proceedings of the 10th International World Wide Web Conference, 2001.
[6] D. Ahlers and S. Boll, "Adaptive geospatially focused crawling," in Proceedings of the 18th Conference on Information and Knowledge Management, 2009.
[7] International Workshop Series on Adversarial Information Retrieval on the Web, 2005–

Author Profile

Pallavi received the B.TECH degree in information technology and M.TECH. degree in Computer Science and Engineering from Maharshi Dayanand University in 2013 and 2015,respectively.