IJCSMS International Journal of Computer Science & Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231 –5268
www.ijcsms.com

Web Crawler: A Review

Dhiraj Khurana[1], Satish Kumar[2]

[1]Assistant Professor, CSE Department
University Institute of Engineering & Technology
Maharshi Dayanand University, Rohtak(Haryana)
dhirajkhurana23@rediffmail.com

[2]Assistant Professor, CSE Department
Vaish College of Engineering, Rohtak (Haryana)
Krsk23@gmail.com

ABSTRACT

In a large distributed system like the Web, users find resources by following hypertext links from one document to another. When the system is small and its resources share the same fundamental purpose, users can find resources of interest with relative ease. However, with the Web now encompassing millions of sites with many different purposes, navigation is difficult. WebCrawler, the Web's first comprehensive full-text search engine, is a tool that assists users in their Web navigation by automating the task of link traversal, creating a searchable index of the web, and fulfilling searchers' queries from the index. Conceptually, WebCrawler is a node in the Web graph that contains links to many sites on the net, shortening the path between users and their destinations.

Keywords: Crawler, Optimization, Duplicate, Webpage, Prioritization

1. INTRODUCTION

A crawler is a program that downloads and stores Web pages, often for a Web search engine. Roughly, a crawler starts off by placing an initial set of URLs, in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated. Collected pages are later used for other applications, such as a Web search engine or a Web cache.

With web crawling, the general processes that a crawler takes are as follows:

- Check for the next page to download - the system keeps track of pages to download in a "queue"
- Check to see if the page is "allowed" to be downloaded - checking a "robots exclusion" file and also reading the header of the page to see if any exclusion instructions were provided do this. Some people don't want their pages archived by search engines.
- Download the whole page.
- Extract all links from the page (additional web site and page addresses) and add those to the queue mentioned above to be downloaded later.
- Extract all words, save them to a database associated with this page, and save the order of the words so that people can search for phrases, not just keywords
- Optionally filter for things like adult content, language type for the page, etc.
- Save the summary of the page and update the "last processed" date for the page so that the system knows when it should re-check the page at a later date.

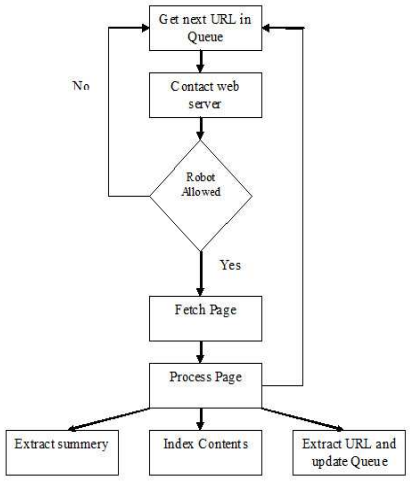

Fig. 1.1 How Crawler Works

These general steps describe how web crawlers work, although they are typically much more complex. The process of downloading the pages

and indexing the contents is very difficult. For instance, if the system downloaded one page at a time, covering the Internet would take several years, as there are typically "bottlenecks" or slow-downs between the web crawler and the various sites it wants to analyze.
Also, when indexing the pages and saving the contents for future searches, this must be done very quickly, and the information must be saved such that future searches don't take huge amounts of time.

2. ROBOTS EXCLUSION PRINCIPLE

2.1 Basic of Robots Exclusion

In 1993 and 1994 there have been occasions where robots have visited WWW servers where they weren't welcome for various reasons. Sometimes these reasons were robot specific, e.g. certain robots swamped servers with rapid-fire requests, or retrieved the same files repeatedly. In other situations robots traversed parts of WWW servers that weren't suitable, e.g. very deep virtual trees, duplicated information, temporary information, or cgi-scripts with side-effects (such as voting).
These incidents indicated the need for established mechanisms for WWW servers to indicate to robots which parts of their server should not be accessed. This standard addresses this need with an operational solution.

2.2 Method

The method used to exclude robots from a server is to create a file on the server, which specifies an access policy for robots. This file must be accessible via HTTP on the local URL "/robots.txt". This approach was chosen because it can be easily implemented on any existing WWW server, and a robot can find the access policy with only single document retrieval.

3. The different data structures involved in the web crawlers are followings: -

Repository: -Manages and stores a large collection of web pages. It contains full HTML of every web page. The web pages are compressed in the repository. Different compression techniques can be used for compressing web pages, which is a tradeoff between speed and compression ratio. In repository documents are stored one after the other.

Document index: - keeps information about each document. It is a fixed width ISAM (Index sequential access mode) index, ordered by doc ID including document status, a pointer into the repository, a document checksum, and various statistics.

Indexer: - It is a program that “reads” the pages, which are downloaded. Web indexing includes indexes to individual websites or web documents to provide a more useful vocabulary for Internet search engines. The indexer also examines the HTML code, which makes up the web page and looks for words that are considered important. Words in bold, italics or header tags are given more priority. Different indexing methods are available.

Hit list: -Stores the list of occurrences of a particular word in a particular document including position, font, and capitalization information. There are two types of hits: fancy hits and plain hits. Fancy hits consider hits occurring in a URL, title, anchor text, or Meta tag. Plain hits include everything else.

4. DIFFERENT TYPES OF WEB CRAWLERS

Different types of web crawlers are available depending upon how the web pages are crawled and how successive web pages are retrieved for accessing next pages. Some of which are following ones

4.1 BREADTH FIRST CRAWLER

Starts out with a small set of pages and then explores other pages by following links in the breadth-first [6] fashion. Actually web pages are not traversed strictly in breadth first fashion but may use a variety of policies. For example it may crawl most important pages first.

4.2 INCREMENTAL WEB CRAWLER

An incremental crawler [5], is one, which updates an existing set of downloaded pages instead of restarting the crawl from scratch each time. This involves some way of determining whether a page has changed since the last time it was crawled. A crawler, which will continually crawl the entire web, based on some set of crawling cycles. An adaptive model is used, which uses data from previous cycles to decide which pages should be checked for updates, thus high freshness and results in low peak load is achieved.

4.2.1 Architecture of Incremental Crawler

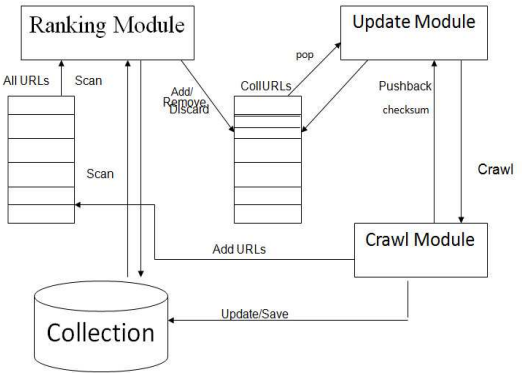

Fig. 4.2 Architecture of Incremental Crawler

4.3 FORM FOCUSED CRAWLER

To deal with the sparse distribution of forms on the Web, our Form Crawler [9] avoids crawling through unproductive paths by: limiting the search to a particular topic; learning features of links and paths that lead to pages that contain searchable forms; and employing appropriate stopping criteria. The architecture of the Form Crawler is depicted

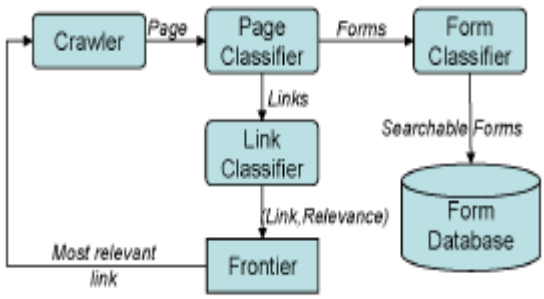

Fig. 4.3 Form Focused Crawler

The crawler uses two classifiers to guide its search: the page and the link classifiers. A third classifier, the form classifier, is used to filter out useless forms. The page classifier is trained to classify pages as belonging to topics in taxonomy. It uses the same strategy as the best-first crawler.

4.4 FOCUSED CRAWLER

The focused crawler [9] has three main components: a classifier which makes relevance judgments on pages crawled to decide on link expansion, a distiller which determines a measure of centrality of crawled pages to determine visit priorities, and a crawler with dynamically reconfigurable priority controls which is governed by the classifier and distiller.
The focused crawler aims at providing a simpler alternative for overcoming the issue that immediate pages that are lowly ranked related to the topic at hand. The idea is to recursively execute an exhaustive search up to a given depth d, starting from the ``relatives" of a highly ranked page.

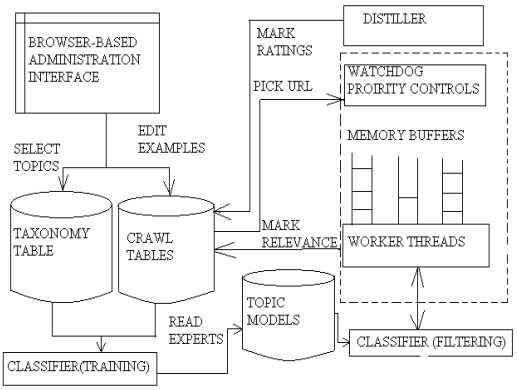

Fig 4.4 Block Diagram of a focused Crawler

4.5 HIDDEN WEB CRAWLER

A lot of data on the web actually resides in the database and it can only be retrieved by posting appropriate queries or by filling out forms on the web. Recently interest has been focused on access of this kind of data called “deep web” or “hidden web [2] ”. Current day crawlers crawls only publicly indexable web (PIW) i.e set of pages which are accessible by following hyperlinks ignoring search pages and forms which require authorization or prior registration. In reality they may ignore huge amount of high

quality data, which is hidden behind search forms.

4.6 PARALLEL CRAWLER

As the size of the Web grows, it becomes more difficult to retrieve the whole or a significant portion of the Web using a single process. Therefore, many search engines often run multiple processes in parallel to perform the above task, so that download rate is maximized. This type of crawler is known as a parallel crawler [2].

4.7 DISTRIBUTED WEB CRAWLER

This crawler [11] runs on network of workstations. Indexing the web is a very challenging task due to growing and dynamic nature of the web. As the size of web is growing it becomes mandatory to parallelism the process of crawling to finish the crawling process in a reasonable amount of time. A single crawling process even with multithreading will be insufficient for the situation. In that case the process needs to be distributed to multiple processes to make the process scalable. It scales to several hundred pages per second. The rate at which size of web is growing it is imperative to parallelism the process of crawling. In distributed web crawler a URL server distributes individual URLs to multiple crawlers, which download web pages in parallel, the crawlers then send the downloaded pages to a central indexer on which links are extracted and sent via the URL server to the crawlers. This distributed nature of crawling process reduces the hardware requirements and increases the overall download speed and reliability along with.

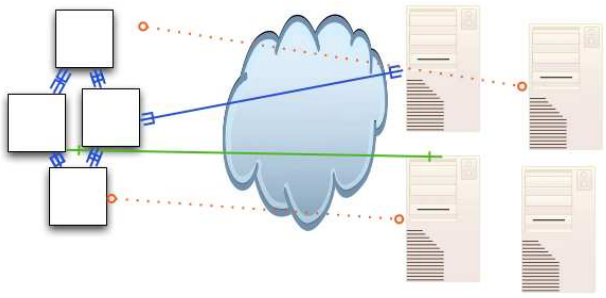

Fig. 4.8 Crawler: PIER nodes

5. CONCLUSION

In this work, Different research paper have been studied on search engines and web crawler and on the basis of that we described general web crawler architecture, Robot Exclusion principle, different data structure involved in web crawling, Architecture and working of different types of web crawlers etc.

REFERENCES

[1] Birrell, A.D., Levin, R., Needham, R.M. and Schroeder, M.D. Grapevine: an exercise in distributed computing. Communications of the ACM, 25 (4) 260-274.1992

[2] A.K. Sharma, J.P. Gupta, D. P. Agarwal, "Augmented Hypertext Documents suitable for parallel crawlers", communicated to 21st IASTED International Multi-conference _ Applied Informatics AI-2003, Feb 10-13,2003, Austria.

[3] Eichmann, D., McGregor, T., Dudley, D., The RBSE Spider - Balancing Effective Search Against Web Load. in First International World-Wide Web Conference, (Geneva, 1994).

[4] Hunt, G.C. and Scott, M.L., The Coign Automatic Distributed Partitioning System. in 3rd Symposium on Operating Systems Design and Implementation, (New Orleans, LA, 1999).

[5] Jumgoo Cho and Hector Garcia-Molina, "The evolution of the Web and implications for an incremental crawler", Prc. Of VLDB Conf, 2000.

[6] Bing Liu, "Web Content Mining" the 14th international world wide web conference (www 2005) China .japan.

[7] Chau Michael, Chen Hsinchun, "Personalized and Focused Crawlers", Department of Management Information Systems, University of Arizona.

[8] Popek, G., Walker, B., Chow, J., Edwards, D., Kline, C., Rudisin, G. and Thiel, G. LOCUS: a network transparent, high reliability distributed system. Operating Systems Review, 15 (5). 169-177

[9] Paolo Boldi_ Bruno Codenotti† Massimo Santini‡ Sebastiano Vigna," UbiCrawler: A Scalable Fully Distributed Web Crawler"

[10] Jumgoo Cho and Hector Garcia-Molina, "The evolution of the Web and implications

for an incremental crawler", Prc. Of VLDB Conf., 2000

[11] Vladislav Shkapenyuk and Torsten Suel, "Design and Implementation of a High performance Distributed Web Crawler", Technical Report, Department of Computer and Information Science, Polytechnic University, Brooklyn, July 2001