2012 International Conference on Computer Technology and Science (ICCTS 2012)
IPCSIT vol. 47 (2012) © (2012) IACSIT Press, Singapore
DOI: 10.7763/IPCSIT.2012.V47.36

Web Crawling Methodology

Ayoub Mohamed H. Elyasir[1], KalaiarasiSonaiMuthu Anbananthen[2]

Multimedia University, Melaka, Malaysia
[1]Email: ayoub_it@msn.com
[2]Email: kalaiarasi@mmu.edu.my

Abstract. Web crawling and its techniques are still in the shadow and possess many secrets due to its involvement in the giant search engine applications where they tend to obscure it, as it is the secret recipe for their success. There is also secrecy involved to protect against search spamming and ranking functions, thereby it is rare to announce or publish complete web crawling architectures.. Web crawler is as an important and fragile component for many applications, including business competitive intelligence, advertisements, marketing and internet usage statistics. In this work, we propose focused web crawler architecture to expose the underneath secrets of web crawling implementation.

Keywords: Web Crawling, Focused Crawler, Search Engine, Uniform Resource Locator, Canonicalization

1. Introduction

The internet is growing tremendously large from thousands of web pages in the nineteenth to billions last few years. In order to traverse and keep track of this huge pool of data, sophisticated search engines indexes thousands of pages in seconds and enables the users to traverse through, using inserted search queries. Web crawler accomplishes the core mission inside the search engine. Whereby it walks through the internet to collect as many web pages as possible according to certain criteria, indexes the relevant pages, keeps the indexed pages up to date and maintains the route paths to various web pages. That is, a web crawler automatically discovers and collects resources in an orderly fashion from the internet according to the user requirements. Different researchers and programmers use different terms to refer to the web crawlers like aggregators, agents and intelligent agents, spiders, due to the analogy of how spiders and crawlers traverses through the networks, or the term (robots) where the web crawlers traverses the web using automated manner.

There are only limited number of papers that explore the crawling programmatic methodology and its various processes, in this paper we surf through focused web crawler and discuss the underneath crawling technique. We propose a crawling architecture with detailed diagrams such as, data flow and flow chart, to unlock the potential secrets of crawling implementation as an extension of our earlier work (Elyasir and Muthu 2012). Diving into the programmatic implementation details gives a better understanding of how search engines work as the web crawler is a vital part of any search engine, in which provides indexing, parsing and repository storage.

2. Review on Focused Web Crawler

Search engine web sites are the most visited in the internet worldwide due to their importance in our daily life. Web crawler isthe dominant function or module in the entire World Wide Web (WWW) as it is the heart of any search engine. Standard crawler is a powerful technique for traversing the web, but it is noisy in terms of resource usage on both client and server. Thus,most of the researchers focus on the architecture of the algorithms that are able to collect the most relevant pages with the corresponding topic of interest. The term focused crawling was originally introduced by (Chakrabarti, Berg, & Dom, 1999) which indicates the

crawl of topic-specific web pages. In order to save hardware and network resources, a focused web crawler analyzes the crawled pages to find links that are likely to be most relevant for the crawl and ignore the irrelevant clusters of the web.

Chakrabarti, Berg and Dom (1999) described a focused web crawler withthree components, a classifier to evaluate the web page relevance to the chosen topic, a distiller to identify the relevant nodes using few link layers, and a reconfigurable crawler that is governed by the classifier and distiller.They try to impose various features on the designed classifier and distiller: Explore links in terms of their sociology, extract specified web pages based on the given query, and explore mining communities (training) to improve the crawling ability with high quality and less relevant web pages.

Liu, Milios and Korba (2008) presented a framework for focused web crawler based on Maximum Entropy Markov Models (MEMMs) that enhanced the working mechanism of the crawler to become among thebest Best-First on web data mining based on two metrics, precision and maximum average similarity. Using MEMMs, they were able to exploit multiple overlapping and correlated features,including anchor text and the keywords embedded in the URL. Through experiments, using MEMMs and combination of all features in the focused web crawler performs better than using Viterbi algorithm and dependent only on restricted number of features.

Liu and Milios (2010) developed their previous framework (Liu, Milios and Korba, Exploiting Multiple Features with MEMMs for Focused Web Crawling 2008), in which they proposed two probabilistic models to build a focused crawler, MEMMs and Linear-chain Conditional Random Field (CRF).Their experiments show improvementson the focused crawling and gave advantage over context graph (Diligenti, et al. 2000) and their previous model.

3. Our Contribution

First we explore the crawling methodology before proposing our architecture to gain better understanding of the implementation details. As shown in Figure 1Web Crawler has a typical life cycle which starts with set of the seed known as Uniform Resource Locator (URL), which is originally extracted from the web cache or the domain name server, and then digs deeper (Download) into the URL to fetch the sub URLs found into the main page. There are enormous issues and sophistications related to the web crawling development such as the network bandwidth and connection consumption, HTML tags and hyperlinks, web sites policies and regulations, storage and repositories, resources usage and scalability.

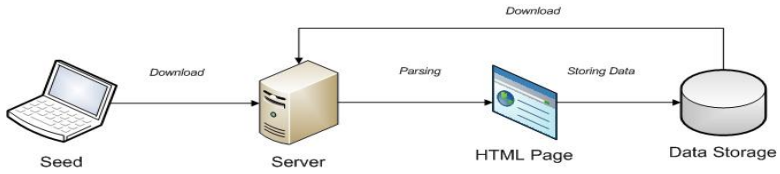

Fig. 1: Web Crawler Methodology

Figure 2shows that any crawler contains what so-called as a (Frontier) which is basically a queue or a list that saves the unvisited URLs and usually followed by a frontier manager to define and organize the queue working mechanism. The user initializes the frontier with seed URLs to start with the crawling process. After the frontier manager passes the URLs for retrieving, it downloads the corresponding web pages and then extracts the entire sub URLs found in the page. The downloading process is accomplished using HyperText Transfer Protocol (HTTP) and recently the secured version of HTTP (HTTPS), whereby the crawler starts acting as a web client and sends HTTP requests to the server and reads the response contents. The crawler finishes the connection by setting a time out state, to ensure that the time is in control and not wasted in waiting for responses from crippled servers and put download restriction to prevent from reading large size web pages. In the focused web crawler, only web pages that match the inserted search query are downloaded and presented in hierarchical way. Hence, the frontier manager contains a classifier to deal with page relationship and association based on the learned pattern in which is originally derived from the search query.

Most of the frontiers in the web crawlers are placed in the main memory for faster data access and easy cashing. The main issue with the frontiers is the URL priority assignment which affects how fast the frontier is filled up due to the huge number of the traversed web pages per unit of time. Thus, the crawler has to be equipped with efficient extraction method in order to fetch the URLs from the frontier manager in optimized manner.

After extracting the web pages and downloading them from the internet, the crawler starts the process of parsing and reading the contents to enable fetching the sub URLs and keeps the crawler in the extraction loop. Parsing can be as simple as extracting hyperlinks from the fetched web pages or it may involve more advanced techniques to analyze the HTML code such as linear scanning or deep tree traversing. Linear scanning is to split the HTML tags based on their functionality like, headers, forms, tables, paragraphs and so on. Deep tree traversing tends to break down the HTML code into a tree of objects to simplify the task of finding the next traversed URL.

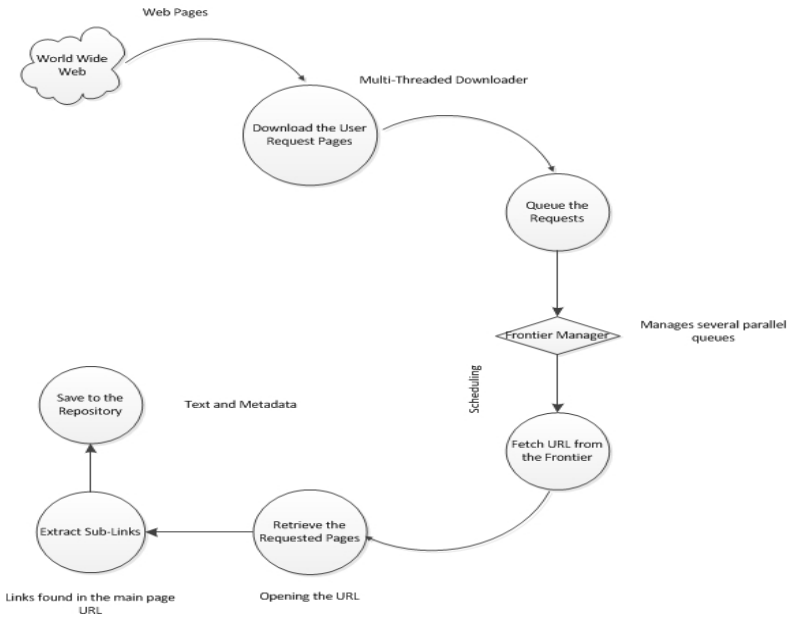

Fig.2: Data Flow for Basic Web Crawler

Process of converting URLs into a special form that is understood by internet routers, switches and Domain Name Servers (DNS) is one of web crawler processes and is called canonicalization. Conversion to the canonical form is independent process with somehow unique mechanism for each crawler. The user is free to specify what kind of rules and conditions the crawler will follow for canonicalization like protocol type and port number. However, crawlers may involve the heuristic approaches when such predefined and algorithmic approaches become impractical.

In Figure 3after canonicalization, resolving URLs into Internet Protocol (IP) addresses and pass it through the network to the DNS as a request to fetch web pages. It is found that, connection establishment between the web crawler, acting as a client, and DNS is the process that makes web crawler a fragile component. The protocol type involved in the connection determines how fast the web pages extraction is. For example, using Unigram Data Protocol (UDP) avoids the troublesome of packet acknowledgment and what so-called three way handshake due to its connectionless property, while Transmission Connection Protocol (TCP) tend to acknowledge and commit three way handshake to open the transmission channel that result in connection overhead. Thus, it is recommended to use UDP to minimize this overhead, network congestion and memory conserve. In addition, asynchronous connection fosters the crawling when a single process is allowed to maintain hundreds of simultaneous connections.

4. Implementation

Figure 4 shows the overall flowchart for our web crawler architecture. The design of the web crawler depends exclusively on the used programming language as there are many libraries that simplify the

implementation details. For example, Java, Perl, Python and C# provide application programming interfaces and libraries for opening URL stream, buffer reading, queue construction, input/output operations and pattern recognition. Thus, implementation tools play a significant role in designing and developing a customized web crawler.

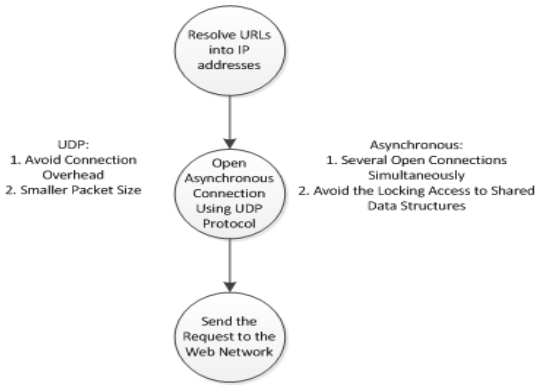

Fig.3: Canonicalization and IP Address Conversion

The first step in our implementation is to feed the crawler with URL input, and then we open an input stream based on that URL which is saved and added to the list by constructing a buffer to read the fed stream line by line. Append all the stream output to one string buffer to simplify the process of text manipulation and control over the attached strings using loops and control statements. All the URLs are added to the linked list after testing certain conditions and constructing another linked list to save the visited URLs to avoid redundancy, hence it optimizes the retrieving process while fetching the URLs for page download.

In the following, we explain the implementation details shown in Figure 4:

- The method Fetch( ) is implemented with one parameters called links which provides the method with a list of URL's to start with
- Crawler will start visiting the first URL and then extract the remaining list of URLs and queue them in (ToVisit) list
- As long as the contents of URLs are eligible for politeness, protocol and type rules, the crawler will remain in the loop of extraction
- To be more polite, we added the (delay) parameter to ensure that the crawler waits for the specified amount of seconds before revisiting the same URL, the more seconds the less the traffic you put on the server and vice versa
- We provide additional control to the frontier manager to ensure that the crawler does not surf the entire web in an endless loop. Depth( ) is a method to maintain the number of visited links within the same domain. Breadth( ) is the other method that takes care of the other domains in the frontier manager
- The Breadth( ) method is capable of queuing the entire cache of web, but that will flood the memory and cause "Memory out of Buffer" error. That is where the priority check of the frontier comes in, whereby best first search is performed to pop the sub URLs in the frontier manager queue that has a limit of customized number.

That was the overview of the details of implementing our focused crawler. What makes our crawler focused is the addition of dictionary that fills up the memory with words and phrases to compare against the extracted URLs

5. Conclusion

Web crawler is basically a simple search engine that traverses, indexes, saves and updates the web pages. Therefore, the giant search engines such as Google, Bing (Microsoft) and Yahoo do not expose most of the advanced crawling techniques. To illustrate the crawling methodology and discover the way it works, we proposed focused web crawler architecture using linked list implementation. Focused and topical web crawlers produce quality results and save more computing resources compare to the standard crawler. Our

architecture depicts the way focused crawler picks the URL and start extracting it, and we show the looping techniques together with the dictionary mechanism that adds on the ability of being focused crawler.

6. Acknowledgement

This work was supported by Project “Mining Opinions Using Combination of Word Sense and Bag of words approach for Educational Environments”, funded by the Fundamental Research Grant Scheme under the 2010–2012.

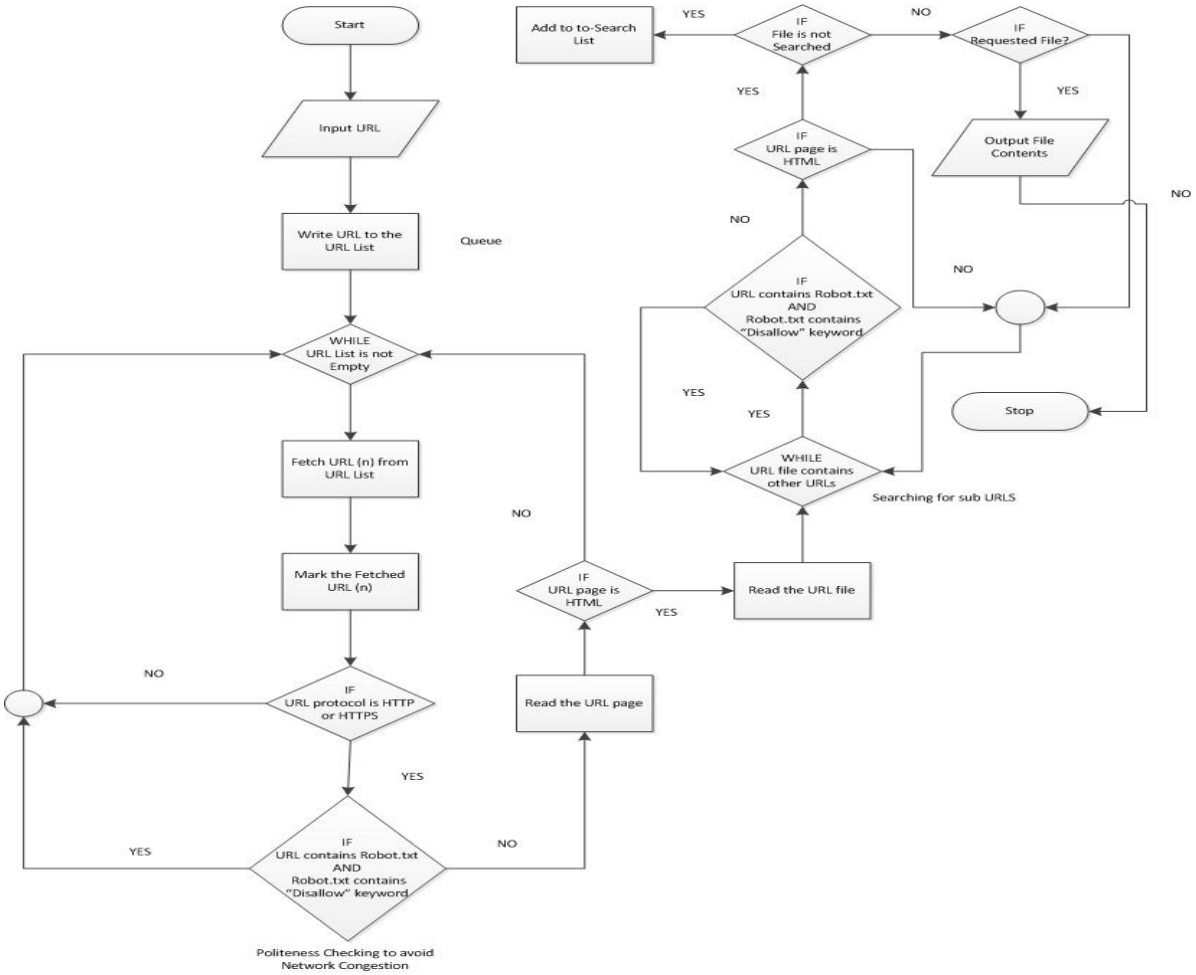

Fig.4: Focused Web Crawler Flowchart

7. References

[1] Chakrabarti, Soumen, Martin van den Berg, and Byron Dom. "Focused crawling: a new approach to topic-specific Web resource discovery." *Elsevier*, 1999.

[2] Diligenti, Coetzee, Lawrence, Giles, and Gori. "Focused Crawling Using Context Graphs." *26th International Conference on Very Large Databases, VLDB 2000.* Cairo, Egypt, 2000. 527–534.

[3] Elyasir, Ayoub Mohamed, and Kalaiarasi Sonai Muthu. "Focused Web Crawler." *International Conference on Information and Knowledge Management (ICIKM 2012).* Kuala Lumpur: ICIKM 2012, 2012.

[4] Liu, Bing. *Web Data Mining* . Chicago: Springer , 2008.

[5] Liu, Hongyu, and Evangelos Milios. "Probabilistic Models for Focused Web Crawling." *Computational Intelligence*, 2010.

[6] Liu, Hongyu, Evangelos Milios, and Larry Korba. "Exploiting Multiple Features with MEMMs for Focused Web Crawling." *NRC*, 2008.

[7] Rennie, Jason, and Andrew Kachites McCallum. "Using Reinforcement Learning to Spider the Web Efficiently." *Proc. International Conference on Machine Learning (ICML).* 1999.