International Journal of Engineering & Technology, 7 (3) (2018) 1119-1123

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET
doi: 10.14419/ijet.v7i3.12924
Research paper

An XML based Web Crawler with Page Revisit Policy and Updation in Local Repository of Search Engine

Jyoti Mor [1] *, Dr. Dinesh Rai [2], Dr. Naresh Kumar [3]

[1] Ph. D. Research Scholar, School of Engineering and Technology, Ansal University, Gurugram, India
[2] Associate Professor, School of Engineering and Technology, Ansal University, Gurugram, India
[3] Associate Professor, CSE Department, MSIT, Janakpuri, New Delhi, India
**Corresponding author E-mail: jyotimor@ansaluniversity.edu.in*

Abstract

In a large collection of web pages, it is difficult for search engines to keep their online repository updated. Major search engines have hundreds of web crawlers that crawl the WWW day and night and send the downloaded web pages via a network to be stored in the search engine's database. These results in over utilization of network resources like bandwidth, CPU cycles and so on. This paper proposes an architecture that tries to reduce the utilization of shared network resources with the help of an advanced XML based approach. This focused crawling based architecture is trained to download only the high quality data from the internet leaving behind the web pages which are not relevant to the desired domain. Here, a detailed layout of the proposed system is described which is capable of reducing the load on network and reducing the problem arise in residency of mobile agent at the remote server.

Keywords: *WWW; Search Engine; Web Crawler; Network Resources; Page Revisit.*

1. Introduction

In such a data intensive world, managing information is an unmatched task. Every second, thousands of Search Engine (SE) crawl through different websites with Web Crawler (WC) to keep their repositories updated and to offer the most pertinent information. With growing popularity of WC, extensive researches have been done in the field of web crawling [1] [2]. According to [3], any SE can search only 16% of the entire web. So, most of the data available on the web remained as unsearched or un-crawled. Lots of web crawling techniques have been proposed to increase the relevancy of search results. One of the techniques that have gained a lot of attention recently is focused crawler [4] [5] [6] [7]. Explored by a lot of researchers, focused crawlers primarily aim to selectively seek out pages that are relevant to predefined set of topics rather than to exploit all regions of web. These types of crawlers only consider the topics of interest and ignore other topics present on the internet [7]. In this paper, a domain specific focused web crawler with novel recommendations has been implemented and promising results in the form of page change behavior, load on the network and bandwidth preservation are presented. The rest of this research article has been ordered as follows: Section II, explain the literature review of different techniques that have been proposed in the past. It gives the overview of different researches that have been recognized by international journals and are worth making a note of. Section III describes the major issues that the current web crawling approaches are facing. In order to better explain our proposed approach, we have recommended a couple of solutions corresponding to each issue. Section IV explains the proposed architecture and its components. Section V and VI describes the working of proposed architecture, its implementation and experimental results respectively. Section VII and VIII provides the benefits and conclusion of this paper.

2. Related work

The authors of [3] proposed a parallel domain focused crawler that retrieves the web pages that are domain specific to reduce the load on network. It makes use of frequency change in the web page and download only those pages that are being changed since the last crawl. In [8] [9] [10] [11], author explains the detailed working of a WC. These articles also explain different crawling techniques like focused crawlers. The basic idea behind focused crawling is that the URLs are classified based on the interest and a specific domain. This way only one specific domain is assigned to the crawler and just the relevant pages are crawled which also helps in getting the high quality content for the SE repository. Another well researched approach is distributed crawling where a central server manages the working of various crawlers that are being distributed throughout the web. It uses page ranking algorithms to assign the URLs to different crawlers. One major disadvantage of these approaches as mentioned in the paper [12] is the repeated downloads or duplicate content of web pages. P. Dahiwale et al. in [13] and M. Kausar et al. in [14], described a detailed study of information retrieval using the concept of genetic algorithm. Main aim of their work was to decide the most promising link and applying the approach of focused crawling and genetic algorithm to produce the optimal and accurate results. A Context Driven Focused Crawler (CDFC) having inbuilt GUI has been proposed in [15]. Augmented hypertext documents with. TVI extension to store various tags and their content have been used in CDFC. Here user has freedom of topic selection with related example and its context through category tree. Crawling was performed by using three agents i.e. User agent, Matcher agent and Dbase agent with specific crawling responsibilities.

3. Issues in the Current Web Crawling Techniques

The study of literatures discussed in [3] [5] [8] [11] [16] have some problems which are listed below:

1) Many irrelevant Web Pages (WP) have been downloaded by the focused crawlers which results in considerable consumption of network bandwidth [8]. They set up multiple crawlers and implement polling method to maintain the repository up to date but both of these methods consume lot of bandwidth.

Focused crawler with multiple mobile agents sometimes creates so many processes on the network which itself

2) produced so much traffic on the network which may results in slower the network or traffic jam [3].
3) RS some time does not allow the mobile agents to crawl some of the web pages or to stay there, in which case the time and resources to visit to the remote server are completely waste [5].
4) If a web page changes continuously and every time the change is being detected and reported to the SE then this again results in wastage of resources even for a single change [11].
5) Link priority evaluation consume lot of memory space which may results in increase in execution time [16].

To overcome these limitations, this paper proposes an architecture that uses XML structure of the WP, meta tag information of the WP and caption of the figure to capture the change in the WP.

4. Architecture of Proposed Crawler

The proposed crawler tackles the above mentioned problems in the following ways:

1) To check the relevancy of a WP, proposed crawler is designed to consider various meta tags, along with image alt attributes and figure captions. The designed crawler is also calculating the number of times a search word is appearing in the page source. This way it only downloads the relevant pages ignoring the ones which are irrelevant to the query or the domain.
2) For making Proposed WC as efficient as possible, it is also designed in such a way that it can ignore the advertisements and banners on the web pages. Advertisements are used to gain financial benefits and increase the popularity of a web page and are totally irrelevant to a page's actual content.
3) Due to the fact that remote server does not allow outside resources to reside on its own server, it is not feasible for a WC to lie on the server for a longer time. Instead of staying at the RS, we have increased the initial frequency of a web crawler like 6 times a week. First 12 visits will allow the crawler to determine the frequency of change of a web page. This would surely reduce the usage of resources on the remote server.
4) To tackle the problem mentioned above, The Proposed WC scans the WP and keeps a count of the number of changes. If the count has reached a certain limit (like 100 new words added), the crawler will stop the scan and download the web page. This will reduce the computation power of the crawler to scan the complete page.

The detailed architecture and working of the proposed crawling technique is explained in next section.

5. Proposed Crawler

Web crawler is designed and sent to the RS for WP change detection. If the changes are found up to a certain threshold, then WP is downloaded. The main components of the proposed crawler are: Page Ranker, Crawler Hand, Data Base File, Change Calculator and Link Retriever. The architecture of proposed web crawler is shown in Fig. 1. And descriptions of these modules are discussed in this subsection.

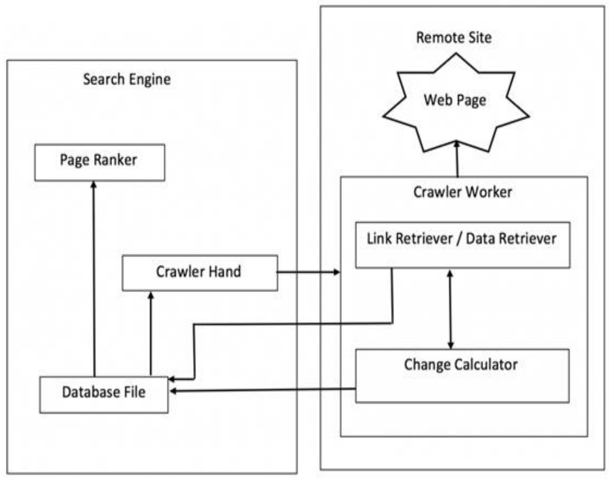

Fig. 1: Architecture of Proposed Crawler.

5.1. Components of Proposed Crawler

Following are the main components of the proposed approach:

a) Crawler Hand: Crawler hand as shown in the Figure 1 retrieves Uniform Resource Locator (URLs) from DNS queue in first come first serve manner and assigns these URLs to the crawler worker. It also keeps note of the domain of the URLs that are being assigned to each crawler worker.
b) Crawler Worker: After receiving URLs from the crawler hand, crawler worker retrieves the robots.txt file to check if the WC is allowed to visit on the website or not. If the WC is allowed, it goes to the website along with the original page source of the assigned URL and does its computations. Figure 2 gives a list of things that each crawler worker is responsible for.
c) Link Retriever: Link retriever is the component that retrieve all the links one by one that are embedded in the WP and is being crawled by the WC.
d) Change Calculator: While WC crawls the page, change calculator keeps a count of the number of changes that occurred in the WP since the last crawl. It divides the entire page in five equal parts based on the total number of characters in the WP and compares each part one by one and counts the number of words that are not there in the original page source. It uses Knuth–Morris–Pratt algorithm [17] for string matching. If the number of words that are not present in the original page source increases the 100, the crawler does not compare the remaining of the five parts and downloads the page.

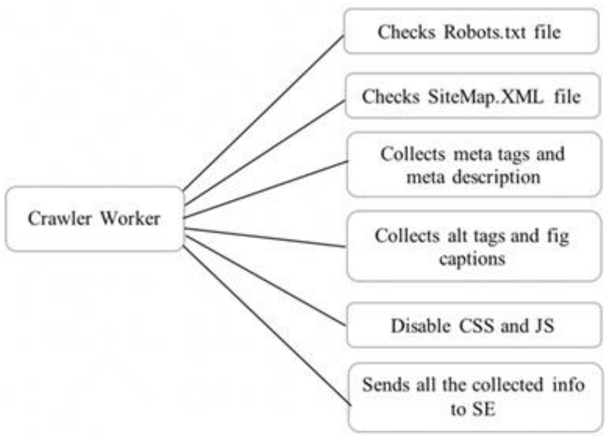
Crawler Worker
Checks Robots.txt file
Checks SiteMap.XML file
Collects meta tags and meta description
Collects alt tags and fig captions
Disable CSS and JS
Sends all the collected info to SE

Fig. 2: Tasks of the Crawler Worker.

e) Page Ranker: Page ranker stays on the SE side and is responsible for ranking the page based on the information sent by the link retriever and the change calculator. It also compares the prepared list of keywords with the downloaded page content and gives weight to the retrieved URL. This way, the crawler would get a prioritized list to crawl. Greater the number of times search keyword appears in the page source higher will be the page ranking and its priority [18]. Besides the number of times search keyword appears, page ranker also considers the number of links that are embedded in the page. It is assumed that the greater number of outgoing links, lesser is the page content value [19].

f) Static Database File: Static database stays on the SE and has all the information related to previous crawls of all the web pages.

6. Working of Proposed Architecture

The working of proposed architecture is represented with the help of flow chart and is shown in Figure 3. Crawler hand collects the URLs from DNS queue and sends the URL one by one to the crawler worker. Besides sending the URLs to crawler worker, it also keeps note of the domains of the URLs that are being sent to the crawler worker. Crawler worker sends these domains along with URLs to the Static Database that is being stored on the client side. After receiving the URLs from the crawler hand, crawler worker checks for the Robots.txt file. Robots.txt file is the file that directs WC how to crawl their website. It also contains instructions of blocking WC to crawl certain WP of the site. If the crawler is allowed to crawl the website, it looks for the SiteMap.XML file. If any new WP link is added to the SiteMap.XML file, crawler goes ahead and downloads the new web page.

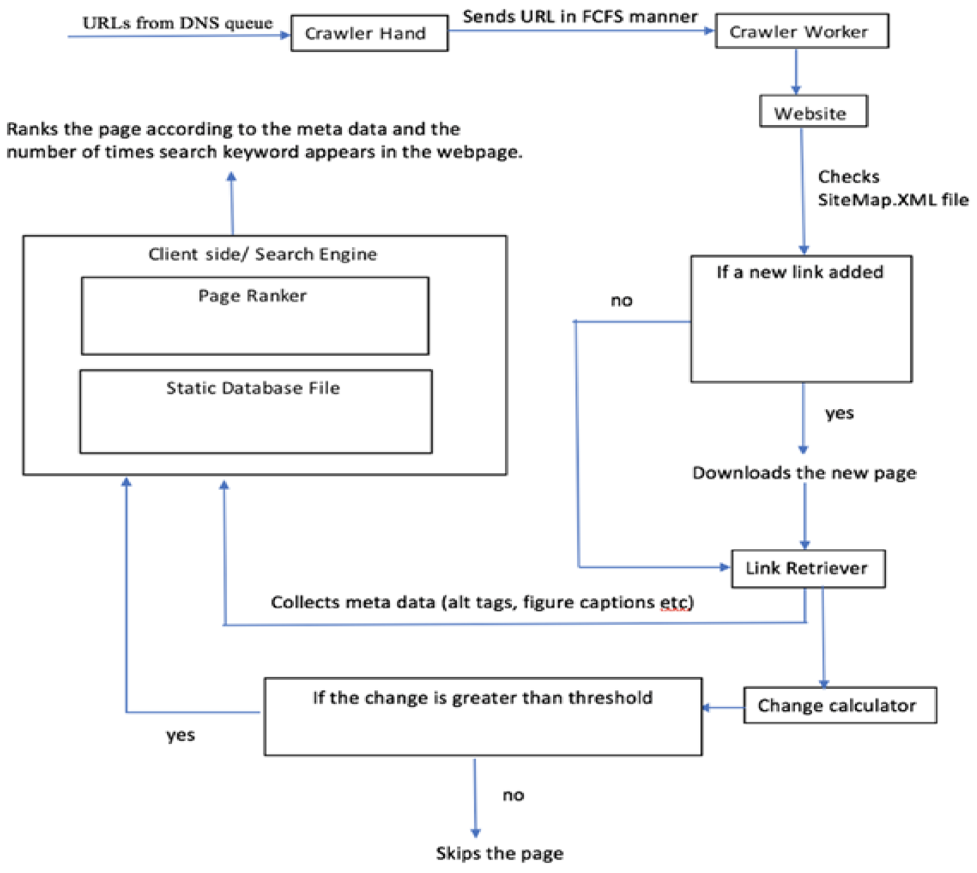

Fig. 3: Flow Chart of the Proposed Architecture.

If no new web page is added, it starts crawling the old pages one by one. While parsing the pages, crawler collects all the alt tags, meta data and figure captions by viewing the page source of the WP. It also counts the number of times the search word appears in the WP. All these meta data are being sent to the static database. Simultaneously, change calculator looks for all the changes and keeps a count of it. If the number of changes is above a pre decided threshold, it stops its further computations and downloads the page. By the end of the WP, if the number of changes is less than the threshold, it is assumed that the WP is not changed enough to be downloaded again and send via the network. After collecting all the meta data and relevant tags, page ranker ranks the WP. It looks for the number of times a search word or a synonym appears in the WP. It also examines the meta data and meta description along with the alt tags and decides if the page is relevant to the corresponding stored domain of the WP or not.

Proposed WC can also to download data from hidden web. For downloading the data from hidden web, WC is trained to disable CSS and JavaScript before downloading the web pages. This way all the styling of the WP will be deactivated. By disabling the JavaScript of the web page, it also removes the possibility of disabling text using document object model. When all the meta data and meta descriptions along with alt tags and figure captions are collected by viewing the page source of the assigned URL is send over to the database of SE. Page Ranker concludes the relevance of the webpage corresponding to the already stored domain of that URL. This way it ranks the collected WP according to the relevancy computed.

7. Implementation and Experimental Results Discussion

The provided crawling architecture has been coded in Java programming language and implemented using IntelliJ IDEA version ULTIMATE. The proposed architecture has been coded in a way that it takes both http and https web URLs.The crawler parses through the WP to find the embedded links in the WP. Besides URLs, this WC also fetches the meta tags, image tag, alt tags and stores them in the output file. To help increasing the relevance of the page, it also takes into consideration the iframe tags and stores them in the output file. After retrieving the embedded URLs from the entered page, WC recursively goes to the entire list of retrieved web URLs and fetches meta tags, image alt tags and iframe tags of each retrieved embedded links. The design has been coded in a way that it downloads the entire HTML content of the entire WP along with the page contents of the embedded URLs.

For each domain, a comprehensive list of keywords is being made that is stored in the client side. Once the entire page source is retrieved, the page source is compared to the prepared list of exhaustive keywords that are relevant to the corresponding domain. Each keyword has a weight. Number of times a keyword appears in the page source is calculated on the client side and the weight is accumulated. Based on the total weight of the URL, the retrieved URLs from that link is prioritized and is handed over to the crawler for its next crawl. Higher the weight, higher would be the priority of its retrieved URLs. If the weight is not above a threshold, it is considered irrelevant and the URLs retrieved from the page are not parsed further. This way, the architecture makes sure that only the high quality content is downloaded without wasting the network resources which are not relevant to the corresponding domains. For searching the keywords in the page source, we have used IntelliJ IDEA.

In the user interface, users can also enter a search keyword which would be searched from the page content and would give a count of number of times the search keyword has appeared in the page content.

Finally, the proposed design gives WP in order of relevance based on the search keyword starting from the WP that contains the highest frequency of the search keyword. Each time the program is being run, an output file is generated by the name of date and time in the downloads folder containing all the retrieved information.

7.1. Collected Data

Every time the java programmed WC is being executed, a file is generated on the SE local directory in the specified path (./users/download folder in this case) by the name of current date and time. The file has .txt extension and stores the data group wise for each URL crawled by the WC. At the starting of the file, it gives the URL that is being accessed by the WC which is followed by the content type of that WP. After that the files generates the complete page content (page source) of the WP. Subsequently, the file has img tags, meta tags and embedded Iframes that are present in the WP. This data is collected and displayed in the output file for all the embedded URLs of the page that is entered in User Interface of the program. At the end of the file, it gives the ranked URLs in order of their relevance of the search keyword being entered by the user.

Accessing [URL] ...
Content-Type: [text/html; charset]
Page content of URL
Img tags found on URL
Meta tags found in URL
Embedded Iframes in URL
Total Page size (bytes)

Fig. 4: Structure of the Output File.

Authors ran the WC continuously for five days on three specific URLs:

- https://en.wikipedia.org/wiki/Main_Page
- http://www.rediff.com
- https://www.yahoo.com

Main reason for choosing these URLs is their continuous dynamic nature which helped us to experiment the proposed WC on the websites that are not static. Following is the graph for an average one-day record of the page size (KB) of these URLs. The graph shown in Figure 5 describes the readings collected after every 2 hours.

The graph shown in Figure 6 represents the number of times the search keyword appears in these three URLs. Readings have been taken for four continuous days. Authors chose the keyword 'news' for the experimental purposes.

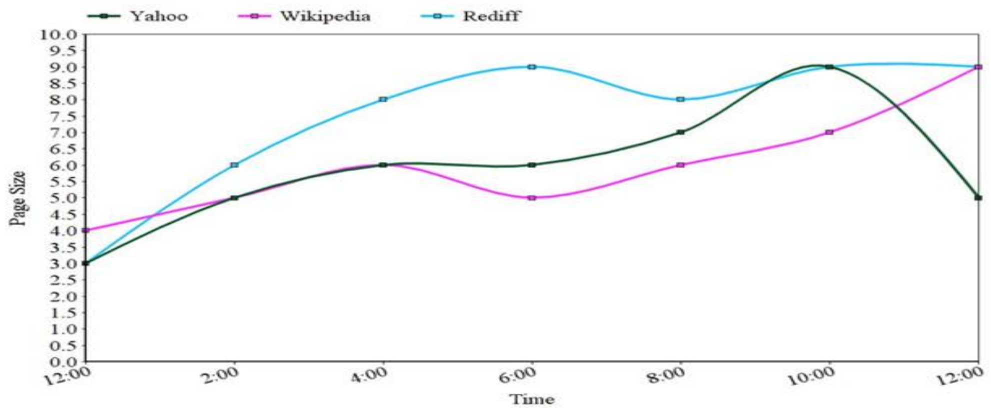

Fig. 5: Page Size (in KB) vs Time (in hours) and Graph for Data Collected.

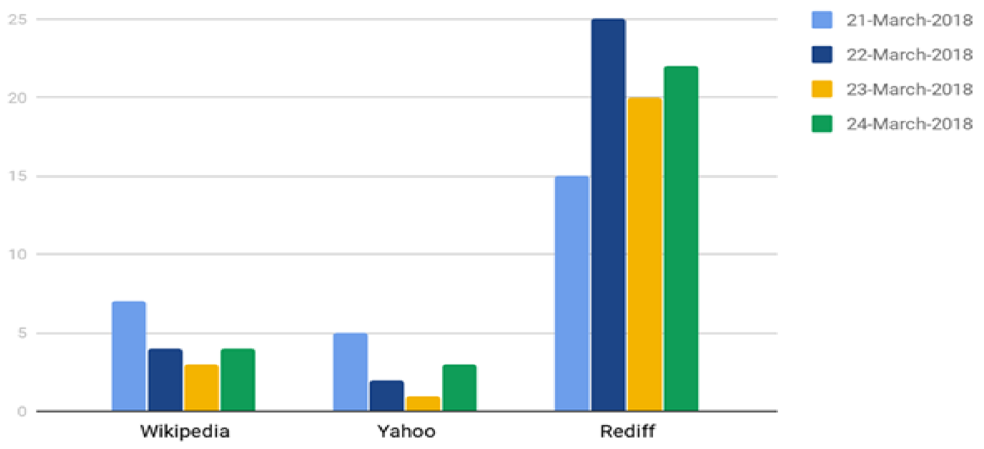

Fig. 6: Number of Time Search Keywords Appears in URL.

8. Benefits from the Proposed WC

Following are the detailed list of gained benefits from the proposed WC: -

a) By retrieving the meta data from the page source of a WP, one can determine if the page is relevant to the assigned domain or not.
b) By using proposed approach one is able to detect the rate of change of WP at a much lower resource utilization.
c) By checking the SiteMap.xml file before drilling into each and every page, authors are able to detect the new additions and changes in the website at a faster rate.
d) The security problem at remote server is also solved because no crawler will stay at remote server.

9. Conclusion

This paper proposes a novel approach on how a SE can effectively reduce the usage of the shared resources on the network with the use of proposed advanced XML based web crawling approach. The presented approach is able to effectively calculate the change frequency and intelligently detects if the WP is needed to be downloaded or have not been changed since the last crawl. Besides reducing the usage of network resources, the proposed approach is capable of acquiring the data from hidden web. This focused based design prioritize the retrieved URLs on the basis of highly focused list of keywords that are directly related to the specific domain of each crawler. This way, only high quality content is being downloaded which is in support of the overall goal of reducing the load and increasing the efficiency of the current SE's phase. With the experimental results, the architecture has been proved to be robust and effective to resolve the current web crawling issues.

References

[1] B. Mahar and C. K. Jha. "A Comparative Study on Web Crawling for searching Hidden Web." International Journal of Computer Science and Information Technologies, 6, (2015), 2159-2163.

[2] M. S. Ahuja, J. S. Bal and Varnica. "Web Crawler: Extracting the Web Data." International Journal of Computer Trends and Technology, 13(2014), 132-137. https://doi.org/10.14445/22312803/IJCTT-V13P128.

[3] R. Nath and N. Kumar. "A Novel Parallel Domain Focused Crawler for Reduction in Load on the Network." International Journal of Computational Engineering Research2 (2012), 77-84.

[4] A. Amaliae, D. Gunwan and A. Najwan. "Focused crawler for the acquisition of health articles" International Conference on Data and Software Engineering, 2016. https://doi.org/10.1109/ICODSE.2016.7936110.

[5] T. Harry, Y. Achsan and W. C. Wibow. "A Fast Distributed Focused-web Crawling." 24th DAAAM International Symposium on Intelligent Manufacturing and Automation, a proceeding of Science Direct (2014), 492 – 499, https://doi.org/10.1016/j.proeng.2014.03.017.

[6] A. Pranav and S. Chauhan. "Efficient Focused Web Crawling Approach for Search Engine." International Journal of Computer Science and Mobile Computing, 4(2015), 545-551.

[7] A. Gupta and P. Anad. "Focused web crawlers and its approaches." International Conference on Futuristic Trends on Computational Analysis and Knowledge Management, IEEE (2015). https://doi.org/10.1109/ABLAZE.2015.7154936.

[8] A. Garg, K. Gupta and A. Singh. "Survey of Web Crawler Algorithms." International Journal of Advanced Research in Computer Science, 8 (2017), 426-428.

[9] M. Kausar, V. S. Dhaka and S. K. Singh. "Web Crawler: A Review" International Journal of Computer Applications 63 (2013), 31-36.

[10] C. Saini and V. Arora. "Information retrieval in web crawling: A survey." International Conference on Advances in Computing, Communications and Informatics, IEEE (2016). https://doi.org/10.1109/ICACCI.2016.7732456.

[11] G. Pant, P. Srinivasan and F. Menczer "Crawling the Web." Web Dynamics. Springer, Berlin, Heidelberg, (2004), 153-177. https://doi.org/10.1007/978-3-662-10874-1_7.

[12] C. Castillo and R. Yates. "Practical Issues of Crawling Large Web Collections." URL: http://chato.cl/papers/castillo_05_practical_web_crawling.pdf.

[13] P. Dahiwale, M. M.Raghuwanshi and L. Malik. "Design and Implementation of Focused Web Crawler Using Genetic Algorithm: An Approach to Web Mining." International Journal of Scientific & Engineering Research, 6 (2015), 254-259.

[14] M. A. Kausar, M. Nasarand S. K. Singh. "A Detailed Study on Information Retrieval using Genetic Algorithm." Journal of Industrial and Intelligent Information, 1 (2013), 122-127. https://doi.org/10.12720/jiii.1.3.122-127.

[15] A. Sefyi, A. Patel and J.C. Junior. "Empirical evaluation of link and content-based Focused Treasure Crawler." Computer Standards & Interfaces, 44(2016) 54-62. https://doi.org/10.1016/j.csi.2015.09.007.

[16] H. Lu, D. Zhan, L. Zhou and D. He, "An Improved Focused Crawler: Using Web Page Classification and Link Priority Evaluation." Mathematical Problems in Engineering, 2016(2016), 1-11. https://doi.org/10.1155/2016/6406901.

[17] https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm, 03/02/2018, at 8.05am IST.

[18] M. Kumar, R. Bhatia and A Ohri. "Design of focused crawler for information retrieval of Indian Origin Academicians." IEEE (2016) https://doi.org/10.1109/ICACCA.2016.7578895.

[19] S. Brin and L. Page. "The Anatomy of a Large-Scale Hyper textual Web Search Engine." WWW conference (1998).