I.J. Modern Education and Computer Science, **2017, 11, 33-40**
Published Online November 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2017.11.04

Improved Architecture of Focused Crawler on the basis of Content and Link Analysis

Bhupinderjit Singh
Department of Computer Science & Engineering,
Dr. B R Ambedkar National Institute of Technology, Jalandhar, India
Email: bhupinderdhani@gmail.com

Deepak Kumar Gupta and Raj Mohan Singh
Department of Computer Science & Engineering,
Dr. B R Ambedkar National Institute of Technology, Jalandhar, India
Email: guptadk@nitj.ac.in, rm_singh14@yahoo.com

Received: 05 August 2017; Accepted: 16 October 2017; Published: 08 November 2017

Abstract—World Wide Web is a vast, dynamic and continuously growing collection of web documents. Due to its huge size, it is very difficult for the users to search for the relevant information about a particular topic of interest. In this paper, an improved architecture of focused crawler is proposed, which is a hybrid of various techniques used earlier. The main goal of a focused crawler is to fetch the web documents which are related to a pre-defined set of topics/domains and to ignore the irrelevant web pages. To check the relevancy of a web page, *Page Score* is computed on the basis of content similarity of the web page with reference to the topic keywords. URLs Priority Queue is implemented by calculating the *Link Score* of extracted URLs based on URLs attributes. URLs queue is also optimized by removing the duplicate contents. Topic Keywords Weight Table is expanded by extracting more keywords from the relevant pages database and recalculating the keywords weight. The experimental result shows that our proposed crawler has better efficiency than the earlier crawlers.

***Index Terms*—**Focused Crawler, Topic Weight Table, Search Engine, Page Score, Link Score, URL Queue Optimization.

I. Introduction

Among the world's total population of 7.5 billion, 3.6 billion are internet users. It means approximately half of the population of the world is on the internet. There are over 1 billion websites on the World Wide Web (WWW) today [1]. On average, Google processes more than 40,000 search queries per second, which means 3.5 billion search queries per day worldwide [1]. As the information and the users on the WWW are growing at rapid rate, so it becomes challenging for Search Engines to meet the needs of all users to search information about a particular topic of interest.

Typing any query on Search Engines gives millions of web documents as result. Most of these documents are irrelevant to the user interest. It is very difficult for the users to find the relevant information from this huge result set. Also, it is very challenging for a general purpose search engine to give the relevant documents for every topic. Search Engine uses Web Crawler to collect web pages from the WWW by following the hyperlinks on these web pages. Web Crawler is the most important component of the Search Engines, and its optimization improves the efficiency of Search Engines [2]. Because of the continuous growth and dynamic nature of WWW, it is almost impossible for a web crawler to crawl the entire web. There arises the need of a special purpose crawler that crawls a particular area of the web and ignores the other regions of the web. This type of special purpose crawler is known as Focused Crawler. Focused Crawler is a program used to find the web documents on a specific topic from the Internet [3]. A primary issue in designing a focused crawler is how to determine which document is relevant and which one is irrelevant to the desired topic. In this paper, we use “Computer Science” domain as a sample topic for the implementation of an improved architecture of focused crawler based on content and link analysis.

A. Web Crawler

A Web Crawler is a program that visits web pages, reads their contents and creates entries for the index of search engines. All the major search engines on the WWW have such a program, which is also known as a "spider" or "bot". Google's web crawler is known as GoogleBot. Web crawlers fetch web pages at a time, following the hyperlinks to other web pages until all web pages have been read [4].

A crawling process basically contains three steps. First, the web crawler starts by downloading a web page, then it extracts new URLs from that web page and stores the relevant information of the web page like Title, Meta Keywords, Meta Description and URL into the index of the search engine, and finally, it visits the hyperlinks (URLs) that are found on the web page. Fig. 1 shows the working of a basic web crawler.

The working of a basic web crawler involves following steps:

1. Select URL(s) from the seed URLs set.
2. Insert it into the URLs Queue.
3. Dequeue URL from the Queue.
4. Fetch the web page corresponding to that URL.
5. Extract new URLs from that web page.
6. Insert the newly found URLs into the Queue.
7. Extract and store the relevant information into the search engine database (index).
8. Go to step 3 and repeat until the Queue is not empty or a specified limit exceeds.

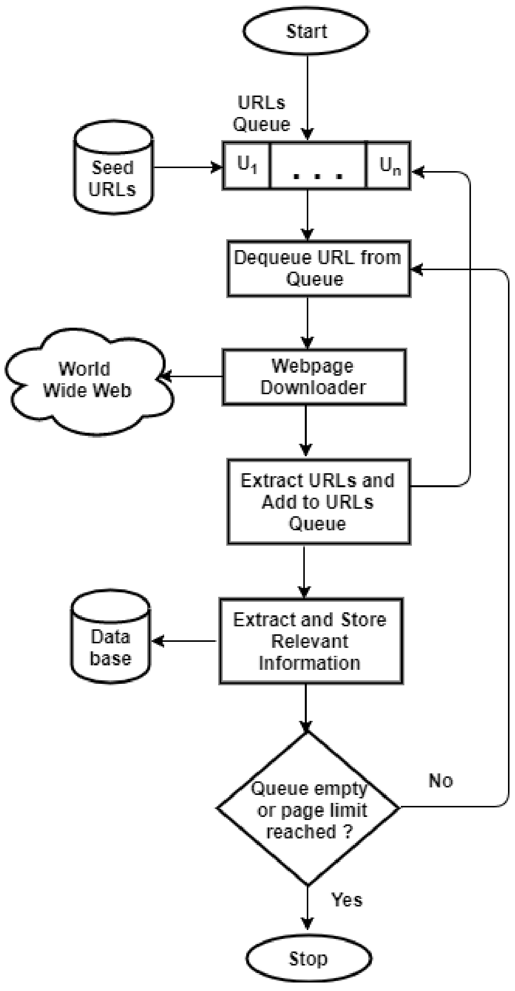
Start
URLs Queue
Seed URLs
U1
. . .
Un
Dequeue URL from Queue
World Wide Web
Webpage Downloader
Extract URLs and Add to URLs Queue
Data base
Extract and Store Relevant Information
Queue empty or page limit reached ?
No
Yes
Stop

Fig.1. Flowchart of Basic Web Crawler

B. Basic Crawling Terminology

Following are some of the basic terminology that is related with web crawlers:

a) Seed URLs

A crawler begins its crawling process by selecting URL(s) from the set of Seed URLs also known as starting URLs. The selection of relevant seed URLs is the most important factor in any crawling process.

b) URLs Queue

After selecting the URLs from the given seed URLs, crawler inserts them into an unvisited list of URLs known as, “URLs Queue or Frontier or Processing Queue”. Then the crawler scheduler dequeue URLs from the URLs Queue for further processing.

c) Parser

Once a web page corresponding to the selected URL is downloaded from WWW, it needs to be processed to extract relevant information from it. The main task of any parser is to parse the downloaded web page to extract a list of new URLs and other relevant information to be stored in the database of the search engine [4].

C. Focused Crawling

A basic web crawler collects all web pages by following each and every URL, whereas a focused crawler gathers web pages that are relevant to a pre-defined set of topics, thus reducing the network traffic and server overhead.

II. Related work

In 1999, S. Chakrabarti, M. Berg, and B. Dom [3] introduced the focused crawler. To achieve focused crawling, they designed two programs: *Classifier* and *Distiller* for mining of web documents. The *Classifier* evaluates the relevancy of a web document with respect to the topic specified, and *Distiller* is used to identify the web documents that lead to a large number of relevant resources.

Our work is most related to A. Pal, D. S. Tomar, and S. C. Shrivastava [6]. They proposed focused crawling based on content and link structure analysis. Cosine similarity measure is used to calculate the relevance of the web page on a particular topic. Link Score is calculated, to assign scores to unvisited URLs extracted from the downloaded web page using the contents of the web page and the metadata of hyperlinks. Link Score is also used in dealing with the irrelevant web pages.

Meenu, P. Singla and R. Batra [7] proposed a focused crawler design where topic-specific weight table is constructed dynamically according to the queries of the users.

M. S. Safran, A. Althagafi, and D. Che [8] has proposed an approach of focused crawling to improve the relevance prediction based on URLs attributes using Naïve Bayesian classifier. Firstly, the training set is built, which contains the value of four relevance attributes: URL word, Anchor text, Parent page, and surrounding text. Then, the Naïve Bayesian classifier is trained to predict the relevancy of unvisited URLs.

D. Hati and A. Kumar [9] proposed an approach in which crawler first fetches those URLs which have high link score as compared to the all other link scores. Link score is calculated on the basis of relevancy of parent pages and division score, which is based on the number of topic keywords present in the division in which the particular link exists. If the link score is greater than the specified threshold value, then the link is treated as a relevant link, otherwise, the link is discarded.

J. Choudhary and D. Roy [10] proposed a priority based semantic focused crawling. This crawler assigns priority values to unvisited URLs according to their semantic score. The semantic score is calculated based on the semantic similarity score of anchor text, web page of unvisited URL and their parent page with respect to the focused topic ontology. URLs and their semantic scores are stored into a priority queue. The URL with the highest semantic score is fetched from the priority queue for crawling.

P. Gupta, A. Sharma, J. P. Gupta, and K. Bhatia [11] present a design of a context based distributed focused crawler. The aim is to get the contextual meaning and senses of the keywords in the user queries and the contents of web pages. Seed URLs are distributed to multiple crawlers to download the web pages corresponding to the URLs. The contextual meanings of the keywords present on web pages and in the user queries are extracted from the Word Net dictionary to create entries for the search engine index.

M. Jamali, H. Sayyadi, B. B. Hariri and H. Abolhassani [12] uses a combination of the link structure of web documents and content similarity of web documents with respect to the given topic. They treated hyperlinks and contents present in the web pages as the authors view about other web pages and it relates them to a particular domain.

X. Chen and X. Zhang [13] proposed a focused crawler with content and link analysis which combines search strategy based on web page content and link structure analysis to improve the page relevancy.

S. Kumar and N. Chauhan [14] proposed a context model for focused web search. In the earlier approach of information retrieval, the user context is ignored. S. Kumar and N. Chauhan proposed a framework that incorporates various context features that lead to highly relevant search results.

III. Proposed Work

We have proposed an improved architecture of focused crawler, which is hybrid of various techniques used earlier in related work. For better efficiency, we have proposed following enhancements in focused crawler architecture:

- ➢ Optimization of URLs queue is done by removing the duplicate URLs and the URLs which lead to the duplicate contents.
- ➢ Topic Keywords Weight Table is expanded by adding more keywords extracted from the relevant pages database and recalculating the keywords weight.
- ➢ URLs Priority Queue is implemented firstly by, assigning the priorities to the Seed URLs and then by calculating the Link Score of extracted URLs based on URLs attributes. Priority values are assigned to the URL’s, and then the URLs are merged into URLs priority queue according to its priority.

We used “Computer Science” domain as a sample topic for the implementation of proposed focused crawler. For Constructing Topic Keyword Weight Table, “CsePedia, An Encyclopedia of Computer and Internet” [15] is used. Focused Crawler is implemented using PHP and DOM (Document Object Model). Fig. 2 shows the working of proposed focused crawler.

Following are the various techniques and algorithms used in the implementation of the proposed focused crawler:

A. URLs Queue Optimization

In WWW, a single web page may have different sources. So, when we crawl web pages, it is possible that a URL may occur in many web pages. Due to this reason, the URL may be extracted a number of times and placed in the URLs queue. Also, there can be a case where multiple web pages may have the same copy of the contents. So, we need to optimize the URLs Queue by removing the duplicate contents. We have used hashing technique, which is very helpful for removing duplicate contents. To remove duplicate contents, we have compared the signatures generated by the MD5 hashing algorithm with the earlier signatures stored in the database. Following steps are used for URLs Queue Optimization:

a) Removing Duplicate URL

To remove duplicate URL, we have compared the URL signature of every URL from URLs queue with earlier URLs signatures stored in the database, so that not to crawl same pages again. If URL signature exists in the database, that URL will be discarded.

b) Removing Duplicate Content

To remove URL having duplicate content, we will compare the content signature of a web page with content signatures earlier stored in the database. If body content signature of the page already exists in the database, that URL will be discarded. Otherwise, that URL will be stored in the new queue, which is optimized one from the earlier queue. In this way with content signature generation, similar or duplicate pages which are syntactically different will be discarded from re-crawling. This works perfectly in reducing processing of duplicate pages.

The above process of URLs Queue Optimization will reduce the number of crawling steps and minimize the operation time, leading to a better efficiency of a focused crawler.

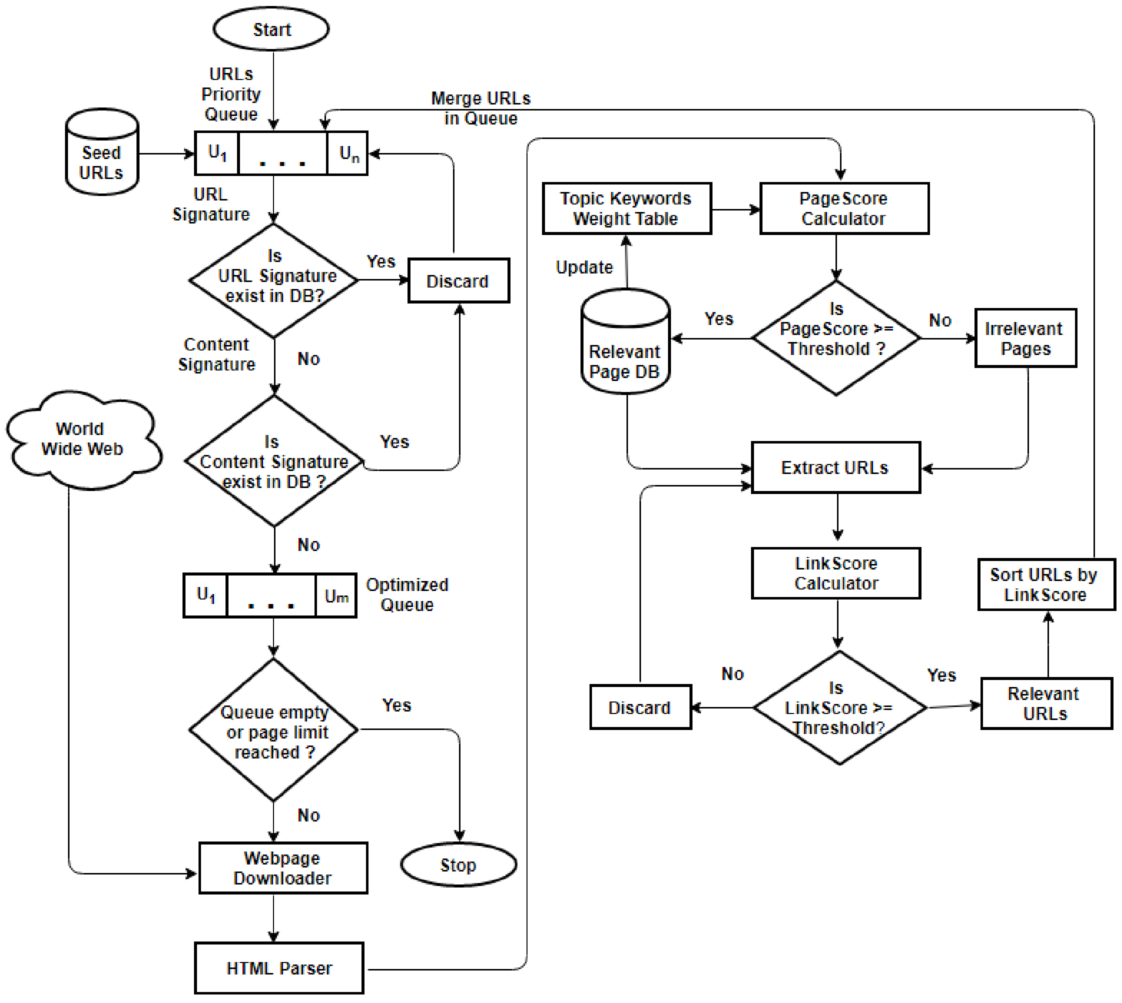

Fig.2. Proposed Architecture of Focused Crawler

B. Algorithm for Seed URLs Selection

Step1: Topic Specific Queries are sent to the Top Three Search Engines: Google, Bing, and Yahoo

Step2: Top N-URLs are fetched from the results of each Search Engine.

Step3: Priorities are assigned to the fetched URLs (refer Table 1).

Step4: URLs are merged into a set according to its priority.

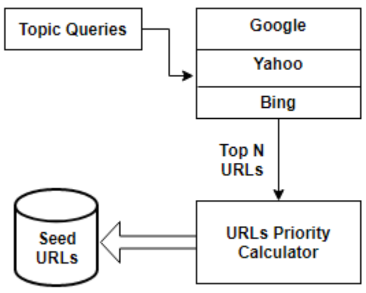

Fig.3. Seed URLs Generator

Table 1. Priority for URLs [16]

Priority	Description
1 (High)	URLs that appear in the result set of all Search Engines
2 (Medium)	URLs that appear in the result set of two Search Engines
3 (Low)	URLs that appear in the result set of only one Search Engine

C. Algorithm for Topic Keywords Weight Table Construction

For Constructing a Topic Specific Weight Table, We crawled over 50 web pages from CsePedia, which contains the information about different areas of Computer Science domain. From these web pages over 300 most relevant keywords are extracted and stored in the topic keywords weight table along with their weights. Fig. 4 shows the flowchart of *Topic Keywords Weight Table Construction.*

Following algorithm is used to extract the keywords from web pages of CsePedia and to calculate their weights.

a) Extract Keywords from Web pages

Step1: Download the web page from the server.
Step2: Remove HTML Tags, Scripts, Styles, and Decode HTML entities.
Step3: Remove Punctuation Character like full stops, commas, dashes, quotes, brackets and other symbols.
Step4: Convert Text to Lower case and Split Text into Tokens (Words List).
Step5: Stem the Words; it reduces words like "Compute", "Computed", and "Computing" to just "Compute".
Step6: Remove Stop Words; It removes English words like "the", "and", "a", "by", and other common words.

b) Calculate the Weights for Keywords

To calculate the weights of keywords, we have used TF-IDF method. It calculates importance of a word to a document in the collection of documents.
Step1: Normalized Term Frequency (TF) is calculated for each word.

$$TF(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document} \quad (1)$$

Step2: Inverse Document Frequency (IDF) is calculated for each word.

$$IDF(t) = lo\ g\left(\frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it}\right) \quad (2)$$

Step3: Keyword Weight is computed as

$$w = TF(t) * IDF(t) \quad (3)$$

Where w is the weight of term t.
Step4: Terms are sorted in descending order according to its weight.
Step5: Terms with the high weight are extracted as Topic Keywords.
Step6: Terms weights are normalized as

$$W = \frac{W_i}{W_{max}} \quad (4)$$

Where W is the normalized weight, W_i is the weight of the ith term and W_{max} is the weight of a keyword with the highest weight.

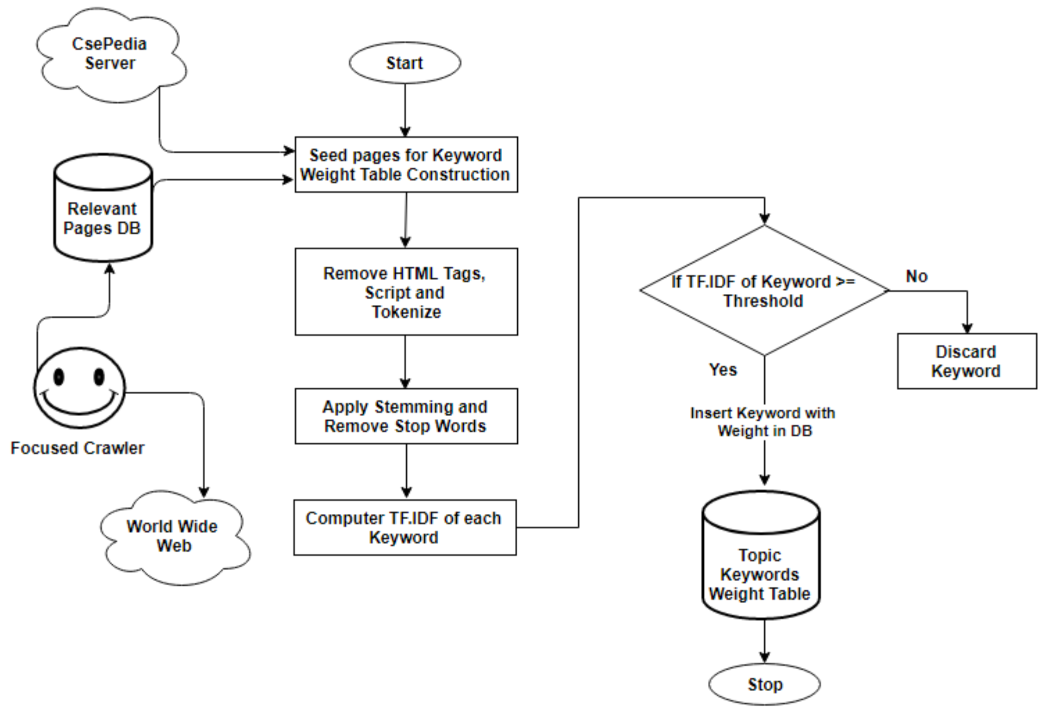
CsePedia Server
Start
Seed pages for Keyword Weight Table Construction
Relevant Pages DB
Remove HTML Tags, Script and Tokenize
If TF.IDF of Keyword >= Threshold
No
Discard Keyword
Yes
Apply Stemming and Remove Stop Words
Focused Crawler
Insert Keyword with Weight in DB
World Wide Web
Computer TF.IDF of each Keyword
Topic Keywords Weight Table
Stop

Fig.4. Flowchart of Topic Keywords Weight Table Construction

c) An Example of TF-IDF Calculation:

Suppose a document contains 100 words wherein the word *computer* appears 14 times. *TF for the word computer is then (14 / 100) = 0.14.* Now, assume we have 1000 documents and the word *computer* appears in 100 of these. *IDF is calculated as log (1000 / 100) = 2. So, the* TF-IDF weight for the word *computer* is the product of these quantities 0.14 * 2 = 0.28.

d) Expansion of Topic Keywords Weight Table

Topic Keywords Weight Table is expanded by extracting more keywords from the relevant pages database and recalculating the keywords weight. This dynamic computation of Topic Keywords Weight Table leads to a better efficiency of a focused crawler.

Table 2. Sample of Topic Keywords Weight Table

S.No.	Keyword	Keyword Weight
1.	Int	1
2.	Code	0.92
3.	Algorithm	0.87
4.	Program	0.81
5.	Database	0.78
6.	Internet	0.76
7.	HTML	0.69
8.	Computer	0.61
9.	Hardware	0.58
10.	Software	0.53

D. Finding Relevant Documents

a) Page Score

The *Page Score* is calculated using the cosine similarity measure (5) to find the relevancy of a web page on a specific topic. We have calculated the weights of keywords present in the web pages matching to the keywords in the Topic Keywords Weight Table. For this, the TF-IDF method is used, that we have already used in constructing the *Topic Keywords Weight Table* [6].

$$Page\ Score\ (t, p) = \frac{\sum k\ \varepsilon\ (\ t \cap p\ )\ wkt\ wkp}{\sqrt{\sum k\ \varepsilon\ t\ (wkt)^2\ \sum k\ \varepsilon\ t\ (wkp)^2}} \tag{5}$$

Where *t* is the T*opic Keywords Weight Table*, *p* is the web page under crawling, *wkt* is the weight of keyword k in the weight table and *wkp* is the weight of keyword *k* in the web page. If the *Page Score* of a web page is greater than or equal to the threshold specified, then this web page is added to the Relevant Pages Database.

b) Link Score

The *Link Score* (6) is the weight assigned to the unvisited URLs, which are extracted from the web pages. *Link Score* is calculated on the basis of metadata of URLs as well as the contents of the web page under crawling [6].

$$Link\ Score\ (u) = \alpha + \beta + \gamma + \omega \tag{6}$$

Where *Link Score* (*u*) is the sum of the different Scores for a URL is calculated based on the different attributes of a URL. α is *URL Words Score* which is the relevancy between topic keywords and the words extracted from the URL. β is *Anchor Score* which is the relevancy between topic keywords and the words extracted from the anchor text of URL. γ is *Parent Score* which is the *Page Score* of a web page from which URL was extracted. ω is *Surrounding Text Score* which is the relevancy between topic keywords and text surrounding the URL. The URLs whose *Link Score* is greater than or equal to the threshold is considered as relevant URLs. Relevant URLs and their score are stored in relevant URLs Set. After that, the relevant URLs are sorted by their *Link Score* and Relevant URLs Set is merged into the URLs Priority Queue.

E. Dealing with Irrelevant Pages

During focused crawling process, a special case may occur regarding the processing of irrelevant pages. An irrelevant page may lead to a relevant page. To handle this situation, we have also extracted URLs from the irrelevant pages and their *Link Score* is calculated. A *Link Score* is greater than or equal to the threshold is considered as relevant URLs and merged into the URLs Priority Queue for further processing. URLs whose *Link Score* is less than the threshold is considered as irrelevant URLs and discarded. Discarding of irrelevant URLs, avoided the downloading of off-topic pages, thereby leading to a better efficiency of a focused crawler.

F. Convert Relative URLs to Absolute URLs

Relative URLs are the incomplete URLs present in the web pages, with some missing portion like Protocol Name, Host Name or the directory path. Relative URLs need to be converted into the Absolute URLs with the help of Base URL under crawling [17]. Suppose, a web page *http://csepedia.com/terms/data.html* is under crawling, Following Table shows some examples of Relative URLs present on this web page and the corresponding Absolute URL after conversion.

Table 3. Types of Relative URLs

Description	Examples
URL begins with /	**Relative URL:** /page.html **Absolute URL:** http://www.csepedia.com/page.html
URL begins with //	**Relative URL:** //www.csepedia.com **Absolute URL:** http://www.csepedia.com/
URL begins with ./	**Relative URL:** ./page.html **Absolute URL:** http://www.csepedia.com/terms/page.html
URL begins with ../	**Relative URL:** ../page.html **Absolute URL:** http://www.csepedia.com/page.html
URL begins with #	**Relative URL:** #section1 **Absolute URL:** http://www.csepedia.com/terms/data.html# section1

IV. Experimental Results

We have implemented our proposed focused crawler, an earlier focused crawler (without the capabilities of Priority URLs Queue, Optimization of URLs Queue and Expansion of Topic Keywords Weight Table) and a BFS Crawler (that simply downloads every page) for the comparison purpose. All three crawlers are implemented using PHP and DOM (Document Object Model). Target domain is “Computer Science”. Crawlers start with the 10 Seed URLs collected from top three Search Engines (Google, Yahoo, and Bing) and crawled about 2000 web pages. To evaluate the performance of these crawlers, the precision metric (7) is used. Table 4, shows the precision rate after crawling 2000 web pages.

$$Precision\ Rate = \frac{\#\ of\ Relevant\ Pages}{\#\ of\ Downloaded\ Pages} \quad (7)$$

Table 4. Precision Rate of different Crawlers

No. of Crawled Pages	BFS Crawler	Earlier Focused Crawler	Proposed Focused Crawler
100	0.5	0.75	0.86
500	0.35	0.67	0.81
1000	0.26	0.64	0.78
2000	0.2	0.65	0.77

The Precision Rate of different crawlers shows that our proposed crawler has increased the precision rate as compared to the Earlier Focused Crawler and BFS Crawler. Our proposed crawler gives 0.77 precision rate after crawling 2000 web pages. Fig. 5, Fig. 6 and Fig. 7 show the comparison of our Proposed Focused Crawler with the Earlier Focused Crawler and BFS (Breadth-First Search) Crawler for the “Computer Science” domain.

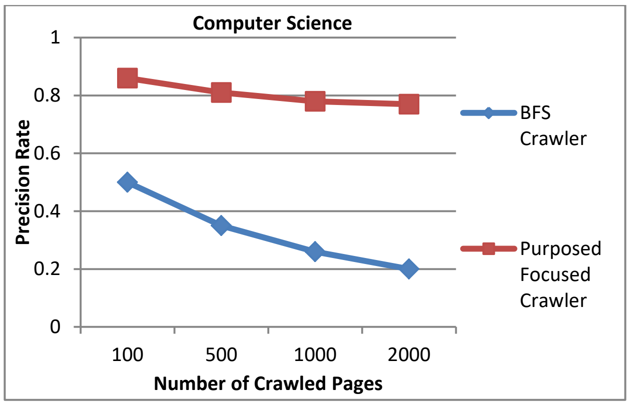

Fig.5. Comparison of Precision Rate of BFS and Proposed Focused Crawler

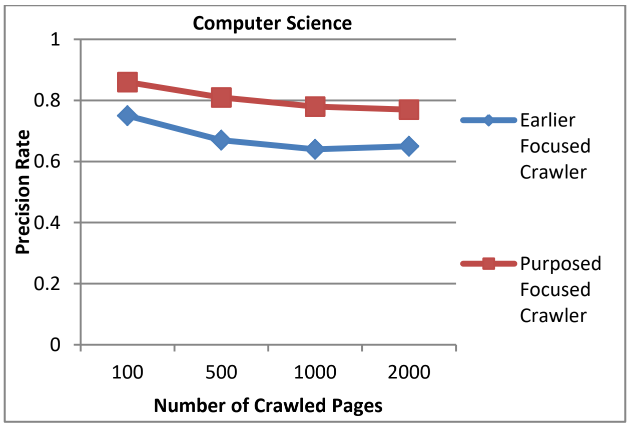

Fig.6. Comparison of Precision Rate of Earlier and Proposed Focused Crawler

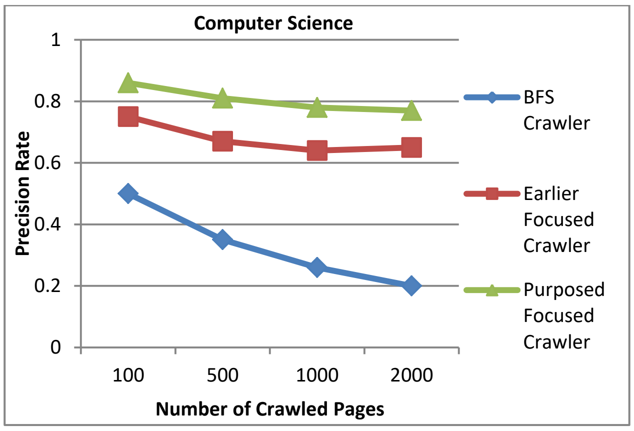

Fig.7. Comparison of Precision Rate of All Crawlers

V. Conclusions and Future Work

In this paper, we presented an improved architecture of focused crawling approach that has the capabilities like Priority URLs Queue, Optimization of URLs Queue and Expansion of Topic Keywords Weight Table. Experimental results show that our proposed focused crawler has better performance than the earlier focused crawler and BFS crawler. The above-said capabilities have optimized the crawling process and minimized the operation time, leading to a better efficiency of a focused crawler.

In our future work, we plan to apply classification and clustering techniques to categorize the retrieved relevant pages into subtopics. It will help search engine users to easily navigate the web pages of their interest.

References

[1] “Internet Live Stats - Internet Usage & Social Media Statistics.” [Online]. Available: http://www.internetlivestats.com/. [Accessed: 16-May-2017].

[2] M. Shokouhi, P. Chubak, and Z. Raeesy, “Enhancing Focused Crawling with Genetic Algorithms,” in *International Conference on Information Technology: Coding and Computing (ITCC’05) - Volume II*, 2005, p. 503–508 Vol. 2. DOI: 10.1109/ITCC.2005.145

[3] M. P. S. Bhatia and D. Gupta, “Discussion on Web Crawlers of Search Engine,” no. April, pp. 227–230, 2008.

[4] M. Levene and A. Poulovassilis, *Web Dynamics*. 2004.

[5] S. Chakrabarti, M. Berg, and B. Dom, “Focused crawling: A New Approach to Topic- Specific Web Resource Discovery,” *Comput. Networks*, vol. 31, pp. 1623–1640, 1999. DOI: 10.1016/S1389-1286(99)00052-3

[6] A. Pal, D. S. Tomar, and S. C. Shrivastava, “Effective Focused Crawling Based on Content and Link Structure Analysis,” *Int. J. Comput. Sci. Inf. Secur. IJCSIS*, vol. 2, no. 1, p. 5, 2009.

[7] Meenu, P. Singla and R. Batra, “Design of a Focused Crawler Based on Dynamic Computation of Topic Specific Weight Table,” vol. 2, no. 4, pp. 617–623, 2014.

[8] M. S. Safran, A. Althagafi, and D. Che, “Improving Relevance Prediction for Focused Web Crawlers,” *2012 IEEE/ACIS 11th International Conference on Computer and Information Science*, 2012. DOI: 10.1109/ICIS.2012.61

[9] D. Hati and A. Kumar, “An Approach for Identifying URLs Based on Division Score and Link Score in Focused Crawler,” *Int. J. Comput. Appl.*, vol. 2, no. 3, pp. 48–53, 2010. DOI: 10.5120/643-899

[10] J. Choudhary and D. Roy, “Priority based Semantic Web Crawler,” *Int. J. Comput. Appl.*, vol. 81, no. 15, pp. 10–13, 2013. DOI: 10.5120/14197-2372

[11] P. Gupta, A. Sharma, J. P. Gupta, and K. Bhatia, “A Novel Framework for Context Based Distributed Focused Crawler (CBDFC),” *Int. J.CCT*, vol. 1, no. 1, pp. 14–26, 2009.

[12] M. Jamali, H. Sayyadi, B. B. Hariri and H. Abolhassani, “Method for Focused Crawling Using Combination of Link Struc ture and Content Similarity,” *Proc. 2006 IEEE/WIC/ACM Int. Conf. Web Intell.*, pp. 753–756, 2006. DOI: 10.1109/WI.2006.19

[13] X. C. and X. Zhang, “HAWK: A Focused Crawler with Content and Link Analysis,” *Int. J. Comput. Technol.*, vol. 2, no. 3, p. 2012. DOI 10.1109/ICEBE.2008.46

[14] S. Kumar and N. Chauhan, “A Context Model For Focused Web Search,” *Int. J. Comput. Technol.*, vol. 2, no. 3, 2012.

[15] “CsePedia - Encyclopedia of Computer and Internet.” [Online]. Available: http://www.csepedia.com.

[16] M. Yuvarani, “LSCrawler : A Framework for an Enhanced Focused Web Crawler based on Link Semantics,” 2006. DOI: 10.1109/WI.2006.112

[17] “PHP tip: How to Convert a Relative URL to an Absolute URL | Nadeau Software.” [Online]. Available: http://nadeausoftware.com/articles/2008/05/php_tip_how_convert_relative_url_absolute_url. [Accessed: 11-May-2017].

Authors’ Profiles

Bhupinderjit Singh received his M Tech degree in Computer Science and Engineering from Dr. B R Ambedkar National Institute of Technology, Jalandhar. His research interest includes Web Mining, Data Structures and Algorithms.

Deepak Kumar Gupta is Associate Professor at the Department of Computer Science & Engineering, Dr. B R Ambedkar National Institute of Technology, Jalandhar. He has total 29 years of experience including teaching and industries. His research interest includes Social Media, Data Analytics and Operating System.

Raj Mohan Singh received his M Tech degree in Computer Science and Engineering from Dr. B R Ambedkar National Institute of Technology, Jalandhar. His research interest includes Cloud Computing, Distributed Computing, Operating Systems and Data Analytics.

How to cite this paper: Bhupinderjit Singh, Deepak Kumar Gupta, Raj Mohan Singh, "Improved Architecture of Focused Crawler on the basis of Content and Link Analysis", International Journal of Modern Education and Computer Science(IJMECS), Vol.9, No.11, pp. 33-40, 2017.DOI: 10.5815/ijmecs.2017.11.04