Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 125 (2018) 584–590

Procedia Computer Science

www.elsevier.com/locate/procedia

6th International Conference on Smart Computing and Communications, ICSCC 2017, 7-8 December 2017, Kurukshetra, India

Keyword query based focused Web crawler

Manish Kumar*[a], Ankit Bindal[a], Robin Gautam[a], Rajesh Bhatia[a]

PEC University of Technology, Chandigarh, India 160012

Abstract

Finding information on Web is a difficult and challenging task because of the extremely large volume of data. Search engine can be used to facilitate this task, but it is still difficult to cover all the webpages present on Web. This paper proposes a query based crawler where a set of keywords relevant to the topic of interest of the user is used to shoot queries on search interface. These search interfaces are found on webpage of the website corresponding to seed URL. This helps crawler to get most relevant links from the domain without actually going in depth of that domain. No existing focused crawling approach uses query based approach to find webpages of interest. In the proposed crawler, list of keywords is passed to the search query interfaces found on the websites. The proposed work will give the most relevant information based on the keywords in a particular domain without actually crawling through many irrelevant links in between them.

Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications.

Keywords: Web crawler; Information retrieval; Focused Web Crawler; Query based crawler.

1. Introduction

A search engine can be defined as a program designed to find information from the World Wide Web (WWW). The search engine produces a result by searching indexed database as per the user query. Typically, the criteria are specified in term of keywords or phrases. The results retrieved are presented in an ordered manner that matches the specified criteria. At the back end, search engines use regularly updated indexes to operate quickly and efficiently. Search engines maintain their database index by searching a large portion of Web. Search engines are different from Web directories as directories are maintained by human editors on the other hand, search engines use crawlers.

* Corresponding author. Tel.: +91-9041858682

E-mail address: manishkamboj3@gmail.com

1877-0509 © 2018 The Authors. Published by Elsevier B.V.

Peer-review under responsibility of the scientific committee of the 6th International Conference on Smart Computing and Communications

10.1016/j.procs.2017.12.075

A Web crawler is also known as a Web spider or Web robot. These are the automated computer program that browses the WWW recursively by following the hyperlinks [1]. The process of getting data from Web by a crawler is called web crawling or spidering. Web crawlers download the visited webpages so that an index of these webpages can be created. A Web crawler starts with a list of Uniform Resource Locator (URLs) to visit, called the seed URLs. As the crawler starts it get all the hyperlinks in the webpage adds them to a list of URLs to be visited further [2].

This paper proposes a query-based focused crawler using searchable interfaces on webpages. These interfaces expose the backend databases of the website whose seed URL is provided. The proposed work is better than the existing approaches as it does not require following a path to reach to webpages of interest. The proposed crawler shoots a set of queries on seed webpages using our dynamic keyword list. We maintain and optimize keyword list by a learning mechanism and update the list dynamically. The rest of the paper is organized as follows: section 2 represents the literature review of existing work. Section 3 discusses in detail motivation behind the work, design and architecture and implementation details of the proposed work.

2. Background and Related Work

Rather than collecting and indexing all the webpages over Internet, focused Web crawler [3] knows its crawl boundaries. It selectively seeks out webpages that are relevant to a pre-defined set of topics. It finds those links on the webpages that are likely to be most relevant while avoiding the irrelevant region of the Web. An upto date review of the various crawler is presented in [2]. The purpose of a focused Web crawler is to collect all the information related to a particular topic of interest on Web [4]. The study [5] discusses execution plans for processing a text database either using a scan or crawl. The method chosen have a great impact on the execution time and precision. Finding the query interfaces for hidden Web is an active area of research [10]. These interfaces are not used for focused crawling.

The keyword query based focused crawler guides the crawling process using metadata. The keyword data set is used for creating effective queries and the result obtained are feedback to the system. An Indian project for tourism and health named Sandhan [6] which is a multilingual platform is an example of the same. This project aims at identifying the language of a webpage using N-gram method. For the training purpose regional, non-regional and health queries are used. Tang et al. [7] proposed a focused crawling for medical information relevance and quality of the webpages retrieved by the query. They use relevance feedback crawler using query by example. Altingovde et al. [8] constructed a query engine that allows keywords and advanced query on the extracted data. A Web portal that is domain specific is finally made that can extract information from the backend database.

3. Proposed Work

This section discusses the motivation behind the work, design and architecture of the proposed work in detail.

3.1. Motivations

This work can be considered as an extension of our previous work [9]. The last developed crawler involved developing a URL ordering based focused Web crawler. The crawler takes input as the files containing: Indian surnames list, Indian cities and Indian premier institutes names list along with the seed URLs. The basic architecture of our last work is shown in figure 1.

Initially, a DFS crawling technique was applied where the crawler started from the seed URL and keeps on crawling the next URLs linked to the webpage blindly until a certain depth is reached. The number keywords matching the keyword databases present on the webpages are counted. The webpages having maximum number of matched keywords was considered as the most relevant.

In this paper, the above mentioned work has been extended. The top 10 most relevant webpages from each domain collected by the above crawler were chosen. From these webpages a list of priority keywords was generated from the words occurring a maximum number of times in these URLs. The list of the priority keywords thus generated was used to prioritize URLs at each step of crawling. It is used to order seed URL and further the URLs obtained to match the relevance of webpages with the topic of interest of the user.

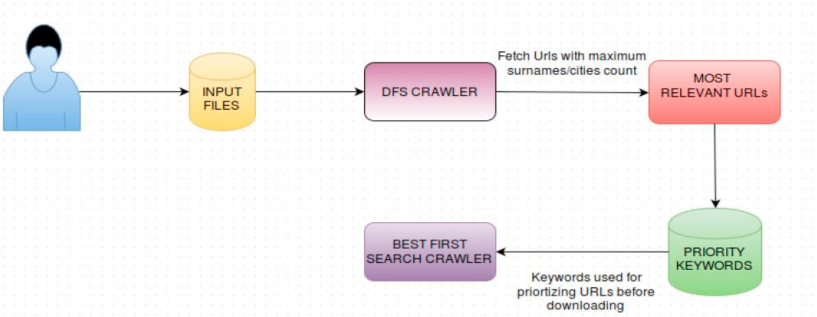

INPUT FILES
DFS CRAWLER
Fetch Urls with maximum surnames/cities count
MOST RELEVANT URLs
PRIORITY KEYWORDS
Keywords used for priortizing URLs before downloading
BEST FIRST SEARCH CRAWLER

Fig. 1. A DFS based focused Web crawler

3.2. Design and Architecture

Figure 2 shows the flowchart for the proposed crawler and its working. It starts when the user gives a set of seed URLs and one of the seed URL is chosen for exploration. The results are obtained using google application programming interfaces (API) and search interface on the seed URL, keywords dataset is used for drafting query in both the cases. Results are collected and merged, the fitness value of each webpage based on the weighted tag is calculated. The webpages are prioritized based on fitness value and explored by the Web crawler. Next, we will discuss the main components in detail.

a) **Initial Seeds:** The seed URLs are those URLs from which crawling process starts. The initial seeds database contains a set of URLs.

b) **Search Interfaces:** From the chosen seed URL, next we try to find the search interface in the corresponding webpage. It consists of finding any search boxes present along with radio buttons, checkboxes, textboxes etc.

c) **Shoot queries using the keywords:** For this purpose, a tool called Selenium is used. After finding search interface on the webpage, queries are drafted for each word in the keyword list.

d) **Google API calls:** If there is no search interface present on webpage corresponds to seed URL or otherwise, the keyword queries are passed for each keyword on Google. The domain of the query is restricted to the seed URL using the advanced search option of the Google. This was done mainly because of the observation that first; there may be no search interface present on some website. Second, the results from the seed URL webpage search and Google search with the domain restricted differ, even if the search interface uses Google-powered search. Thus, to incorporate all the results Google API is used.

e) **Top results collected:** The top results of all the keyword queries are collected for both seed URL and Google API search.

f) **Merge both results:** The results collected from both seed URL keyword search and Google API search are then merged into one list for each keyword. The duplicate links are removed from the list thus giving the actual relevant links from the search.

g) **Fitness Value Calculation:** The fitness value of each webpage in the merged list is then calculated by first creating a Document Object Model (DOM) tree of the webpage based on its tag structure as depicted in figure 3. Weights of each tag are assigned and provided as input to the crawler at this step. There are two methods implemented for calculation of fitness value that is discussed next.

h) **Prioritize URLs:** Based on the fitness value of the web page, the URLs are then prioritized based on their fitness value in a priority queue.

i) **Crawl based on priority:** Now based on the priority, crawler get the webpages in the order of the priority of the URLs.

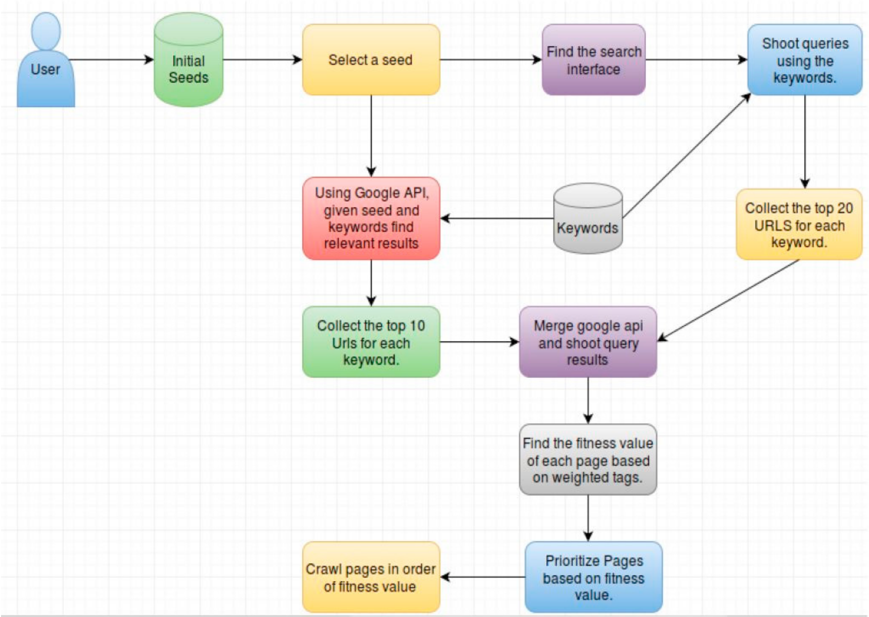

User
Initial Seeds
Select a seed
Find the search interface
Shoot queries using the keywords.
Using Google API, given seed and keywords find relevant results
Keywords
Collect the top 20 URLS for each keyword.
Collect the top 10 Urls for each keyword.
Merge google api and shoot query results
Find the fitness value of each page based on weighted tags.
Prioritize Pages based on fitness value.
Crawl pages in order of fitness value

Fig. 2. Flowchart for the proposed crawler working

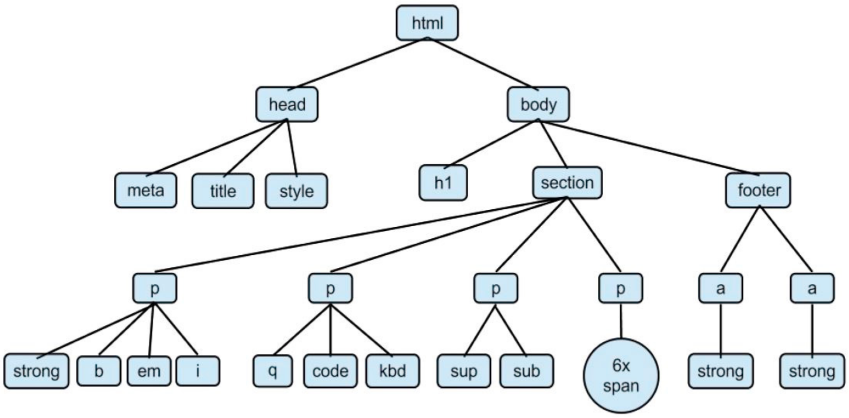


Fig. 3. DOM tree created for a sample HTML webpage

Max Weight Method: The maximum weight on the path from the root to leaf is then assigned as the path score. The sum of all the path scores on a webpage is the fitness value of webpage.

$$Path\ Score = \max\left(weight\ of\ 1st\ Ancestor\ , weight\ of\ 2nd\ Ancestor, weight\ of\ 3rd\ Ancestor\ \ldots weight\ of\ nth\ Ancestor\right)$$

For example figure 3 show one of the path scores will be

$$Path\ Score = \max\left(\ weight\ of\ 'strong'\ tag, wieght\ of\ 'paragraph'\ tag, weight\ of\ 'h1'\ tag, weight\ of\ 'body'\ tag, weight\ of\ 'html'\ tag\ \right)$$

Level K Weight: In this method, the weights of all the tags in the path from root to leaf are taken into consideration for calculating the fitness value of a page. The path score is calculated as:

$$Path\ Score\ = (weight\ of\ 1st\ parent\ *\ k) + (weight\ of\ 2nd\ parent\ *\ 2k) + \ldots (weight\ of\ nth\ Parent\ *\ nk)$$

Example of path score:

$$Path\ Score\ = (weigth\ of\ strong\ tag\ *\ k) + (weight\ of\ paragraph\ tag\ *\ 2k) + (weight\ of\ section\ *\ 3k) + (weight\ of\ body\ tag\ *\ 4k) + (weight\ of\ html\ tag\ *\ 5k)$$

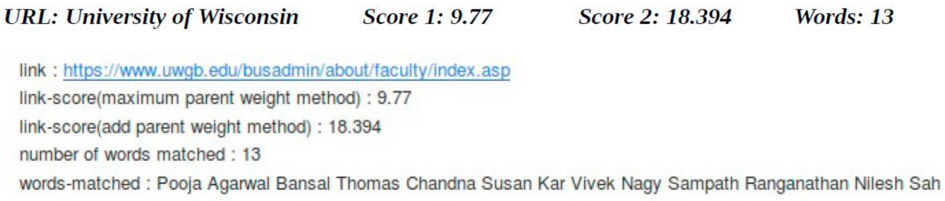
$$Totalscore/Relevancy\ = Path\ Score\ 1\ +\ Path\ Score\ 2\ +\ Path\ Score\ 3 + \ldots Path\ Score\ i\ ;\ where\ i\ is\ total\ no\ of\ words\ matched\ in\ the\ given\ page.$$

3.3. Implementation details

The proposed keyword query based Web crawler is implemented using Python. The main libraries used are Beautiful Soup, Selenium client API and WebDriver, Google Search API, Regular Expressions Module, Urllib2.

4. Results and Discussions

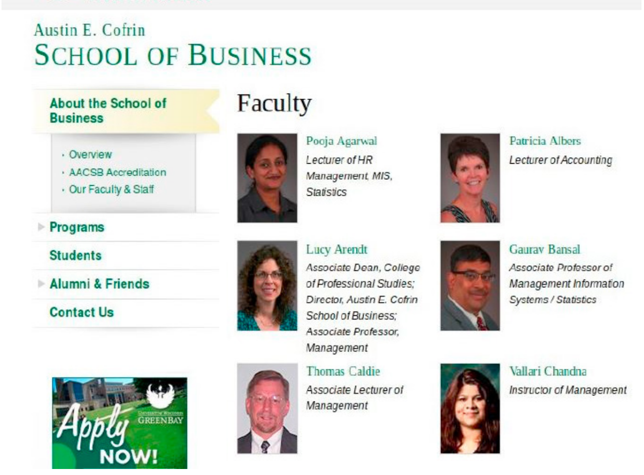
This section presents the discussion and analysis of test runs on various websites. As a part of our main project, we tested the proposed Web crawler to find the webpages of Indian origin academicians working outside India. As input to the crawler, a keyword list is prepared. The crawler was run on 25 foreign universities websites to find Indian origin academicians. For a sample, we are representing the result for the University of Wisconsin.



URL: University of Wisconsin *Score 1: 9.77* *Score 2: 18.394* *Words: 13*

```
link : https://www.uwgb.edu/busadmin/about/faculty/index.asp
link-score(maximum parent weight method) : 9.77
link-score(add parent weight method) : 18.394
number of words matched : 13
words-matched : Pooja Agarwal Bansal Thomas Chandna Susan Kar Vivek Nagy Sampath Ranganathan Nilesh Sah
```

Fig. 4. Sample results for University of Wisconsin



Austin E. Cofrin
SCHOOL OF BUSINESS

Faculty

Pooja Agarwal
Lecturer of HR Management, MIS, Statistics

Patricia Albers
Lecturer of Accounting

Lucy Arendt
Associate Dean, College of Professional Studies; Director, Austin E. Cofrin School of Business; Associate Professor, Management

Gaurav Bansal
Associate Professor of Management Information Systems / Statistics

Thomas Caldie
Associate Lecturer of Management

Vallari Chandna
Instructor of Management

Fig. 5. Webpage corresponding to the scores calculated

In figure 4, Score 1 represents the score using maximum parent method and Score 2 calculated using level K method. Figure 5 represents the corresponding webpage of the University

URL Ordering Based Vs Query Based Web Crawler

In our previous work URL ordering based Web crawler [9], the relevant URLs were found after some depth after crawling through many other webpages while in our query based crawler, we find those relevant links at first depth only as shown in figure 6 for various foreign university website.

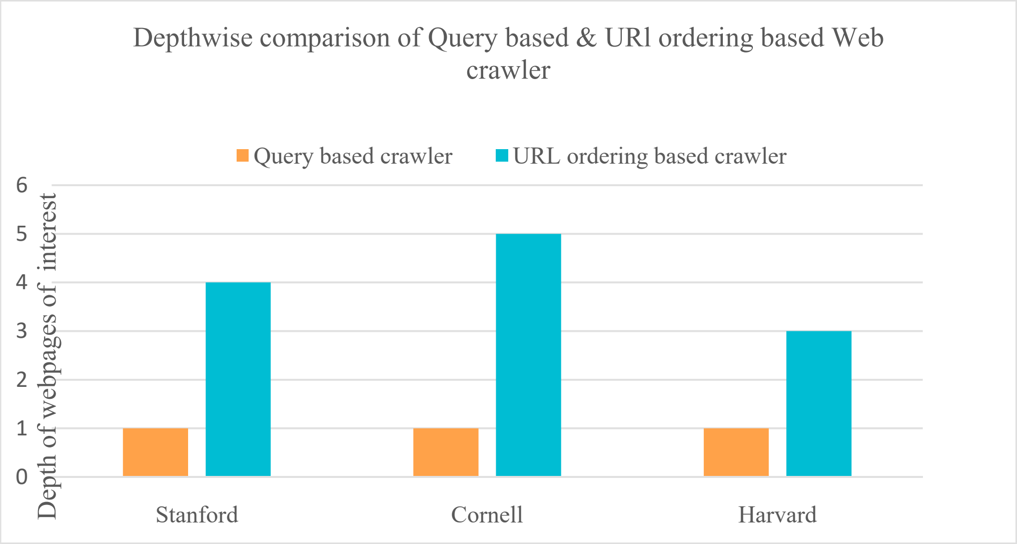


Fig. 6. URL Ordering Based Vs Query Based Web Crawler

Webpage relevancy calculation: K-Level method vs Max ancestor method

Webpage relevancy calculation is one of the most important aspects of any Web crawler. It gives us how important the webpage information is to our crawler. We structured two ways of doing the same and results are shown in figure 7. If we use only the occurrence of keywords as the webpage relevancy criteria, It doesn't give us the complete picture. Considering the HTML structure of the webpages gives us another parameter to compare these webpages in a structured way.

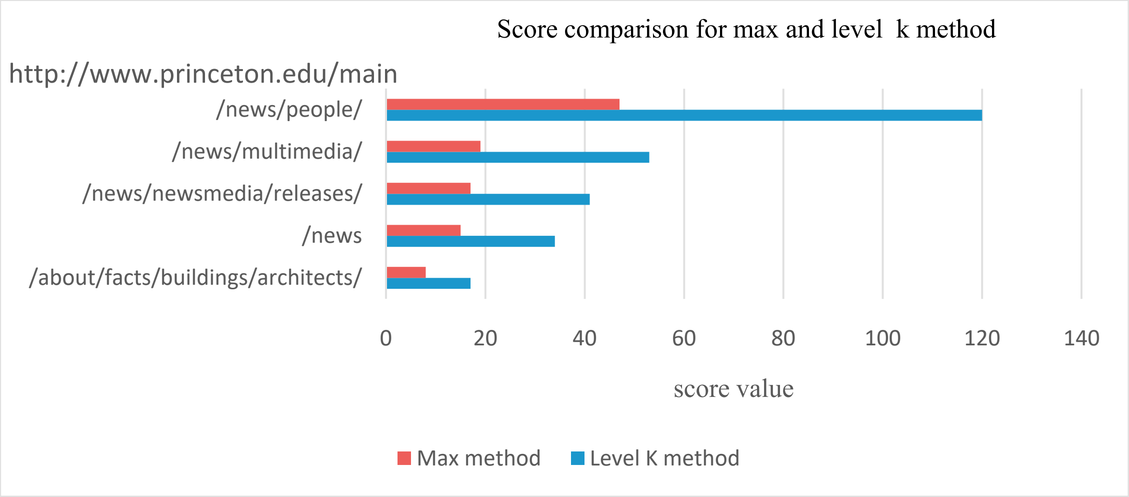


Fig. 7. Webpage relevancy calculation using two methods (domain: Princeton)

Interpretations

1. In Intra domain, as the structure of webpages are similar so k-level method is better than max method. Also along with the weight of each ancestor of a texttag field, it includes the depth of the text-field where it is present, giving a nice idea about the relevancy of the webpage.

2. In Inter domains, max method is better as the structure of two domains may not be similar, so considering the most weighted ancestor tag for each text field to calculate page relevancy gives a good idea of how much the page is relevant.

5. Conclusion

This paper discusses a keyword query based focused crawler in which the webpages are crawled fast. The webpages of interest are crawled independent of the level at which they are present in the website. Query based crawler is more efficient than the previous BFS crawler in terms of time taken and precision. Page relevancy calculator uses DOM structure of relevant webpages. This method takes the webpage along with the Meta tags into consideration while deciding the relevancy of a webpage. We discussed K level method and max ancestor method for calculating the relevancy of a webpage. From results and discussion, it can be concluded that K level method is better for intra domain and max ancestor method is better for inter domain.

References

[1] Brin, S., Page, L. (2012) "Reprint of: The anatomy of a large-scale hypertextual web search engine." *Comput. Networks*. **56 (18)**: 3825–3833. doi:10.1016/j.comnet.2012.10.007.

[2] Kumar, M., Bhatia, R., Rattan, D. (2017) "A survey of Web crawlers for information retrieval." *Wiley Interdiscip. Rev. Data Min. Knowl. Discov*. e1218. doi:10.1002/widm.1218.

[3] Shokouhi M, Chubak P, Raeesy Z. (2005) "Enhancing focused crawling with genetic algorithms." In *Information Technology: Coding and Computing, 2005. ITCC 2005. International Conference on 2005,* IEEE Apr 4, 2: 503-508.

[4] Chakrabarti, S., Van Den Berg, M., Dom, B. (1999) "Focused crawling: A new approach to topic-specific Web resource discovery." *Comput. Networks.* 31 (11), 1623–1640. doi:10.1016/S1389-1286(99)00052-3.

[5] Ipeirotis, P.G., Agichtein, E., Jain, P., Gravano, L. (2007) "Towards a query optimizer for text-centric tasks". *ACM Trans. Database Syst.* **32 (4)**: 21 doi:10.1145/1292609.1292611.

[6] Priyatam PN, Vaddepally SR, Varma V. (2012) "Domain specific search in indian languages." *In Proceedings of the first workshop on Information and knowledge management for developing region 2012 Nov 2, ACM*: 23-30.

[7] Tang, T.T., Hawking, D., Craswell, N. and Griffiths, K. (2005) "Focused crawling for both topical relevance and quality of medical information." *In Proceedings of the 14th ACM international conference on Information and knowledge management, October 2005, ACM:* 147-154.

[8] Altingovde IS, Ulusoy O. (2004) "Exploiting interclass rules for focused crawling." *IEEE Intelligent Systems.* 2004 Nov;**19 (6):**66-73.

[9] Kumar M, Bhatia R, Ohri A, Kohli A. (2016) "Design of focused crawler for information retrieval of Indian origin Academicians*." In Advances in Computing, Communication, & Automation (ICACCA)(Spring), International Conference on 2016 Apr 8, IEEE*:1-6.

[10] Zhao, F., Zhou, J., Nie, C., Huang, H., & Jin, H. (2016). SmartCrawler: A Two-stage Crawler for Efficiently Harvesting Deep-Web Interfaces. IEEE transactions on services computing, 9(4), 608-620.