IJCSE International Journal of Computer Science and Engineering Open Access
Research Paper Volume-2, Issue-3 E-ISSN: 2347-2693

An Approach to Design Personalized Focused Crawler

Hardik P. Trivedi[1*], Gaurav N. Daxini[2], Jignesh A. Oswal[3], Vinay D. Gor[4], Swati Mali[5]

Student, BE Computers, K. J. Somaiya College of Engineering, Mumbai, India [1*,2,3,4]

Assistant Professor, M Tech Computers, K. J. Somaiya College of Engineering, Mumbai, India [5]

www.ijcseonline.org

Received: 1 March 2014 Revised: 14 March 2014 Accepted: 26 March 2014 Published: 31 March 2014

***Abstract*—** The amount of data and its dynamicity makes it impossible to crawl the World Wide Web (WWW) completely. It's a challenge in front of crawlers to crawl only the relevant pages from this information explosion. Thus a focused crawler solves this issue of relevancy to a certain level, by focusing on web pages for some given topic or a set of topics. Also a focused crawler with a page change detection policy can help in narrowing down the search to only newer pages, and thus eliminates risk of redundancy and missing updated data. This paper proposes a policy for design of a focused crawler with web page change detection policy.

***Keywords*—** Web Crawler, Focused Crawler, World Wide Web(WWW), Content Analysis, Link Scoring, Change Detection.

I. Introduction

A web crawler is an automated program that methodically scans WWW and downloads pages that can be reached via links. With the exponential growth of the web, fetching information about a particular topic is very important. A focused crawler is one that attempts to download only those web pages that are relevant to a predefined topic or set of topics instead of crawling the entire WWW. In order to determine whether a web page is relevant or not, focused crawler uses various classification techniques. The web is dynamic and it keeps on changing, so it is important to keep track of web pages that change very frequently.

All the search engines track the users' digital footprints and use the data for user profiling, analysis and personalization. Also, most of the results are based on in general user preferences, rather than being very personalized. The motivation behind developing a Personalized Focused Crawler is to provide targeted information to user i.e. providing information based on user's interest solely. Along with this feature, this application also focuses on user's privacy, for instance user details and their log activities are not forwarded to any third party applications.

This paper is organized in four sections with introduction preceding the literature review in section II. In section III, we propose our approach for Personalized Focused crawling. We follow this by Conclusion in section IV and the last part contains Acknowledgement.

II. Literature Review

There are various kinds of crawler architectures as in, parallel crawler, distributed crawler, intelligent crawlers, etc. Amongst all, the common issues are information retrieval and result ranking as per the relevance.

Hardik Trivedi, hardiktrivedi0612@gmail.com

Basic Working

A crawler initially starts with a set of seed URLs. It fetches the first page from crawl queue and checks whether it is relevant or not based on its relevance score with respect to search string provided by the user. If that page is relevant it is stored into a database. Then URLs from that page are extracted and inserted into crawl queue. Now the crawler fetches another page from crawl queue and repeats the process.

Relevance Calculation

There are two approaches through which relevancy of a page can be calculated.

1) Content Analysis: For a given page, title-text is more important and descriptive than the body-text. Thus title-text should have higher weight than the body-text. The weight of topic words [1,4,7] in different positions calculated as:-

$$Wk = ct * fk \text{ for title text and}$$
$$Wk = cb * fk \text{ for body text,}$$

where ct and cb are meta and body coefficients respectively. We assume ct=2 and cb=1. Wk is the weight of a topic word k and fk is the frequency of word k in the related text.
This weight variable is used to calculate the page relevance with cosine similarity measure as follows [1]:-

$$\text{Cosine}(d_1,d_2)=(\sum w_{1i} * w_{2i})/\sqrt{((\sum w_{1i}^2)* (\sum w_{2i}^2))}$$

Here, $d_i = (w_{i1}+w_{i2}+\ldots\ldots..+w_{ij})$ where w_{ij} represents the weight assigned to term t_j in document d_i. If the retrieved page is similar to the query terms given in the topic words weight table, then it is called relevant, otherwise irrelevant.

2) Link Scoring: In link scoring for web pages, the crawler also checks the links included in irrelevant pages and it keeps on crawling through them up to a certain threshold. Then, the crawler calculates link scores using the following equation and decides whether to fetch pages specified by links or not. The link score can be calculated as [1,4]:

$$\text{LinkScore}(u) = \text{URLTextScore}(u) + \text{AnchorTextScore}(u) + \text{NumberOfParentPages}(u) + [\text{Relevance}(p_1) + \text{Relevance}(p_2)+\ldots+\text{Relevance}(p_n)].$$

Thus crawler spends less time on irrelevant pages and does not miss relevant pages, which are children of an irrelevant page.

In a crawling process, the effectiveness of the focused crawler does not just rely on the maximum amount of relevant pages to be fetched but it also depends on the speed of the crawling process. The speed of a crawler depends on the number of relevant URLs inserted into the URL queue. Therefore, a mechanism, that we call URL optimization, is required for the crawler to select links that are more likely to be relevant. The URL optimization enables the removal of certain links whose link scores are below a predefined threshold point. For URL optimization, we used two different link score evaluation methods for relevant pages.

- Same as above equation which is used for irrelevant pages.
- Naive Bayes (NB) classifier to compute link scores[1,5].

Change Detection

Change detection for a web crawler is the technique by which the web crawler decides which web pages to re-crawl, by detecting the changes that have incurred in the web pages. There are two approaches through which changes in a page can be detected.

1) Freshness Tuning: Freshness Tuning[2] provides four categories of crawl behaviors. To apply a crawl behavior, specify URL patterns for the behavior.

- Crawl Frequently[2]: Use Crawl Frequently patterns for URLs that are dynamic and change frequently. Any URL that matches one of the Crawl Frequently patterns is scheduled to be re-crawled after a certain period based on frequency of updating.

- Crawl Infrequently[2]: Use Crawl Infrequently Patterns for URLs that are relatively static and do not change frequently. Any URL that matches one of the Crawl Infrequently patterns is not crawled very frequently, regardless of its page relevance or how frequently it changes.

- Always Force Re-crawl[2]: Use Always Force Re-crawl patterns to prevent the crawler from crawling a URL from cache.

- Re-crawl URL Patterns [2]: Use Re-crawl URL Patterns to submit a URL to be re-crawled. URLs that you enter here are re-crawled as soon as possible.

2) Dom Tree: A clear way to decrease the load time of a site is to decrease the file size of the HTML documents[8]. And one of the methods to do this is by creating a DOM tree. In this method, the HTML pages are transformed into XML files so that their later access becomes faster. Also another advantage of the DOM Tree method[3,6] is during the process of re-crawling, the matching of two web pages for similarity becomes very easy. This method works in two steps. In the first step document tree is generated for the downloaded web page while in the second step, level by level comparison between the trees is performed. The downloaded page is stored in the repository in search/insert fashion as given below[3]:-

- Search the downloaded document in the repository.
- If the document is found then compare both the versions of the document for structural changes using level by level comparison of their respective document tree.
- Else store the document in the repository.

III. PROPOSED SYSTEM

This paper proposes an approach for a multi-user personal focused web crawler with the facility of detecting the changes in the search result page since the last access.

Basic Working

The main crawler logic will get triggered, when the user provides the search string. A crawler initially has a crawl queue that contains set of seed URLs.
The seed URL can be provided in two ways:

- Pre-defined seeds based on user preferences.
- Seeds that the user provides.

The crawler program will fetch the first page from crawl queue and check whether it is relevant or not based on its relevance score with respect to search string provided by the user. If that page is relevant, then only is it stored into database. Then URLs from that page are extracted and inserted into crawl queue. Now crawler fetches another page from crawl queue and repeats the process.
The system is further divided into two modules:-

- Relevance calculation to retrieve relevant pages.
- Change Detection to detect any changes in content of pages.

Relevance Calculation

Whenever an input string has been entered by the user, the crawler must provide the links that are related to it. To do so, the crawler must start from a seed page and then select the corresponding outgoing links from it and then check whether a link selected is relevant or not.

Here in this system, relevancy of a page is calculated according to Word Frequency i.e. the number of times the search string is repeated in the web page. Word Frequency is counted and the links are sorted in descending order of count and results to the user are displayed according to it.

Following is the pseudo code for calculating word frequency:

```
1.  Extract page source.
2.  while (current_line!= EOF) // scanning each line
3.  {
4.          if(current_line contains search_string)
5.          {
6.                  word_frequency++;
7.          }
8.  }
```

Change Detection

To check whether the page has been changed or not, we compare the checksum of old page stored in database with the checksum of new page. If checksums are same then there is no change in the page, if checksums are different then we assume that there has been some change in the page and we must re-crawl that page.

To compute checksum:
Checksum (P) = ASCIISumOfPageContent / DistinctCharacterCount.

Algorithm

```
1. set of seed URLs is given as input to the crawler.
2. input search string provided by user.
3. select the seed pages based on user preferences and search string.
4. insert the seed pages into crawl queue of URLs.
5. while the crawl queue is not empty
6. {
7. get the first URL from the list.
8. Calculate word frequency based on search string provided
9. if page is relevant
10. {
11. insert current page details into corresponding user database
12. move the page to the already searched list i.e. (visited list).
13. fetch all links present in that page
14. while the page contains another link(linktobeinserted),
15. {
16. validate the linktobeinserted
17. if the linktobeinserted is present in visitedlist.
18. {
19. discard the link.
20. }
21. else
22. {
23. add it to the inserted list.
24. }
25. }
26. }
27. }
```

Architecture

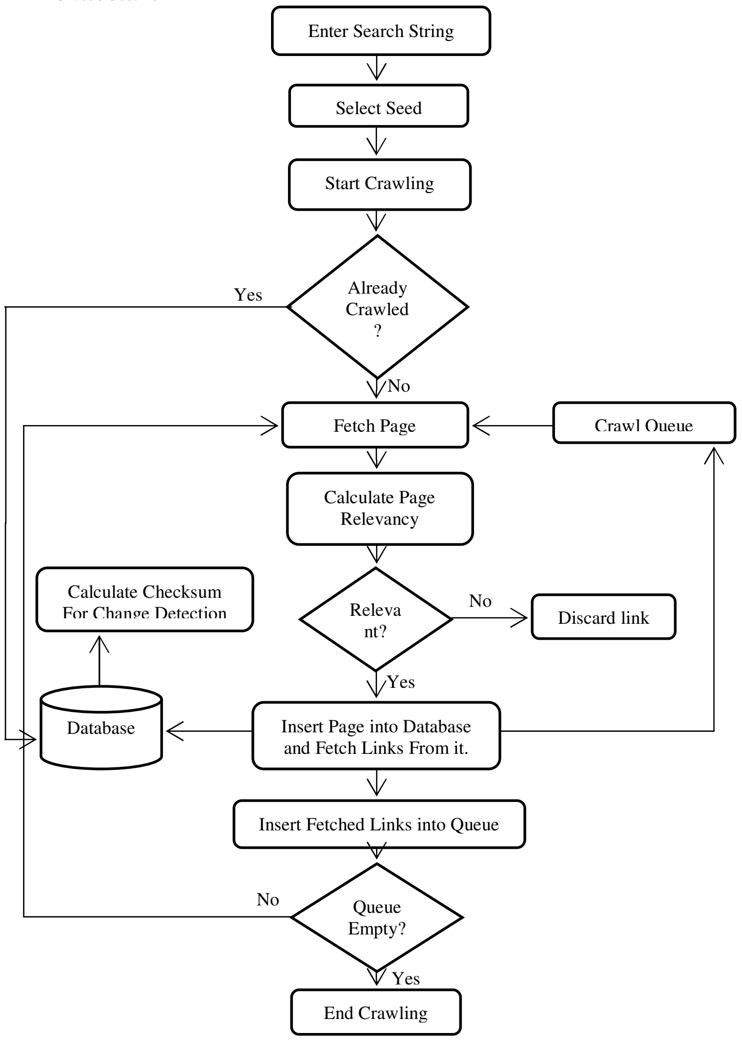

Fig.1: Crawler Architecture.

IV. CONCLUSION

This paper provides a method to develop a personalized web crawler. This web crawler is personalized by providing results and suggestions based upon the user profile, user defined seeds or links that have been visited by the user. Here, relevancy of a page is calculated using word frequency method. Now for every crawler, it is challenging to provide updated information regularly to user. Thus it is necessary to re-crawl pages that are changing very frequently. In this paper we have discussed the change detection method based on the content of web pages.

Future work includes improving the crawler efficiency to speed up the crawling process. A major scope for future work is to have extensive tests with a large volume of web pages and perform change detection of pages which include graphical information.

ACKNOWLEDGMENT

The research presented in this paper would not have been possible without the kind support and help of Prof. Swati Mali Deshpande (K. J. Somaiya College of Engineering). We also wish to thank all the other anonymous contributors for their valuable guidance and support.

REFERENCES

[1] Mahdi Bazarganigilani, Ali Syed and Sandid Burki, “Focused web crawling using decay concept and genetic programming”, published in International Journal of Data Mining & Knowledge Management Process (IJDKP), Vol.1, No.1, Page no(1-12), January 2011.

[2] 3Swati Mali and B B Meshram, “Focused Web Crawler with Page Change Detection Policy”, published in International Journal of Computer Applications (IJCA) proceedings on International Conference and workshop on Emerging Trends in Technology (ICWET), No 9 Article 9, Page No 51-56, 2011.

[3] 4DivakarYadav, AK Sharma, Sonia Sanchez-Cuadrado, Jorge Morato, “an approach to design incremental parallel webcrawler”, published in Journal of Theoretical and Applied Information Technology, Volume 43 No 1, Page no:(8-29), 15 September 2012.

[4] 6Anshika Pal, Deepak Tomar and S.C. Shrivastava, “Effective Focused Crawling Based on Content and Link Structure Analysis”, published in (IJCSIS) International Journal of Computer Science and Information Security, Vol. 2, no. 1, Page No: (1-5), June 2009.

[5] 7Ioannis Avraam and Ioanni Anagnostopoulos, “A Comparison over Focused Web Crawling Strategies”, published in Panhellenic Conference on Informatics(IEEE), Print ISBN 978-1-61284-962-1,Page No: (245-249), September 2011.

[6] 9Weicheng Ma, Xiuxia Chen and Wenqian Shang, “Advanced deep web crawler based on Dom”, published in IEEE Fifth International Joint Conference on Computational Sciences and Optimization, print ISBN 978-1-4673-1365-0, Page No: (605-609), June 2012

[7] Mejdl S. Safran, Abdullah Althagafi and Dunren Che, ”Improving Relevance Prediction for Focused Web Crawlers”, published in IEEE/ACIS 11th International Conference on Computer and Information Science, print ISBN 978-1-4673-1536-4, page no: (161-166), May 2012.

[8] Jatinder Manhas, “A Study of Factors Affecting Websites Page Loading Speed for Efficient Web Performance”, published in International Journal of Computer Sciences and Engineering (IJCSE), Vol-1, Issue-3, Nov 2013.

AUTHORS’ PROFILES

Hardik Trivedi is currently a student of K. J. Somaiya College Of Engineering, Computer branch, Mumbai, India.

Gaurav Daxini is currently a student of K. J. Somaiya College Of Engineering, Computer branch, Mumbai, India.

Jignesh Oswal is currently a student of K. J. Somaiya College Of Engineering, Computer branch, Mumbai, India.

Vinay Gor is currently a student of K. J. Somaiya College Of Engineering, Computer branch, Mumbai, India.

Ms. Swati Mali is working as Asst. Professor in the department of Computer Engineering, at K. J. Somaiya College of Engineering, Vidyavihar, Mumbai. She holds an M.tech degree from VJTI, Mumbai.