International Journal of Engineering & Technology, *7 (3.29) (2018) 550-553*

International Journal of Engineering & Technology

Website: *www.sciencepubco.com/index.php/IJET*

Research paper

Effective approach to crawl web interfaces using a two stage framework of crawler

Samiksha M. Nakashe [1] *, Dr. Kishor R. Kolhe [1]

[1] Department of Information Technology, MIT College of Engineering, Pune, India
**Corresponding author E-mail: samikshanakashe4@gmail.com*

Abstract

Nowadays, internet is important part of our life. User can explore answer to different queries according to his requirement using internet. The nature of these web resources is dynamic and they are present in huge amount. So it becomes challenge to search quality results of required query efficiently as well as personalized search is also a major challenge in Information retrieval. To handle these challenges, a two-stage framework of web crawler is proposed. In first stage, crawler performs “Reverse searching” that matches user searched query with the URL of link from site database. In second stage, crawler performs “Incremental prioritizing” that matches the searched query content with web document. Then crawler classifies relevant and irrelevant pages according to match frequency of searched keyword and ranks these pages. Proposed crawler performs searching through personalized searching according to user point of interest which is based on profession profile of user. The crawler performs the domain classification which helps user to know the contribution of standard resources of searched query. A separate log file is maintained by crawler considering the issue of searching time. While entering cursor in search box, user will get pre-query result based on past search results. Our objective is to design a Focused Crawler to effectively search the site database and provide quality result to the user.

Keywords: *Focused Crawler; Incremental Prioritizing; Information Retrieval; Reverse Searching; Web Crawler*

1. Introduction

Information retrieval (IR) is finding material of an unstructured nature, usually text that satisfies an information need from within large collections. Crawling process is an important part of internet for information retrieval. Web crawler plays an important role in crawling process. Web crawler also known as robot or spider is a massive download system for web pages. A web crawler is a program or automated script which browses the World Wide Web in a methodical, automated manner. This process is called Web crawling or spidering.
There are two types of crawlers, Generic Crawler and Focused Crawler [6]. Generic crawlers‘s is used to take care of basic client’s demand. While focused crawlers are solution to coverage problem that is they select those URLs that are similar to specific topics and remove the irrelevant data. Main components of web search engines, systems that assemble large web pages, point to them and allow users to publish queries in the index and find web pages that match queries [7].
In web there is growing interest in techniques that help you to locate the web interfaces efficiently. However, due to the large volume of web resources and due to its dynamic nature, reaching a broad coverage and providing efficient result is a challenge [4 - 6].
We propose a two-stage framework for efficient searching of web interfaces. In the first stage, Crawler performs Link based searching for centre pages with the help of search engines, avoiding visiting a large number of pages. In second stage, crawler matches form content and then it classify the links as relevant and irrelevant links. Here, we developed personalized search for efficient results according to user interests and separate log file is maintained by crawler for efficient time management.

2. Literature review

Web crawling is most important method to search the data on internet. It helps to discover web page as well as to download that documents. On internet reaching a broad coverage web interface and finding efficient and quality links are major a challenges. To take insight in reference to this problem following papers were referred:
S. Chakrabarti [1], described the concept and process for web crawling. In this two hypertext mining programs are given that guide crawler: a classifier that evaluates the relevance of a hypertext document with respect to the focus topics, and a distiller that identifies hypertext nodes that are great access points to many relevant pages within a few links, which helps to achieve effective performance of crawler.
C. Sheng [2], proposed the algorithms for searching in hidden database. Generalized crawler cannot effectively index hidden databases. There is rapid growth in the amount of such hidden data, which limited the scope of information accessible to ordinary Internet users. These issues are solved by the algorithms which extract all the tuples from a hidden database and allow user to crawl a hidden database in its entirety with the smallest cost.
L. Shou [3], proposed a method to generate online profile for searching to provide security to user’s identity and its confidential information. This paper presented a client- side privacy protection framework called UPS for personalized web search.
D. Kumar [4], proposed a process which is done by crawler by indexing deep web sites for efficient access. In this Deep Web public access contents that are hidden data indexed in such way that it can be efficiently crawl by general search engine crawler.

V. Shukla [5], described Pre/Post query processing approach and site-based searching approach which can be combine together in order to pre-process the user query. By integration of different processing approaches and link ranking approaches a lot of valuable user time is saved. Post query system filter out all irrelevant information which is not necessary according to the query which is been fired, and gives the expected results.

F. Zhao [6], proposed a two stage crawler to harvest deep web interfaces. The smart crawler in this proposed system performs site- based searching and in- site searching. To achieve more accurate results by focused crawler, Smart crawler ranks websites to prioritize highly relevant ones for a given topic. Smart crawler achieves fast in-site searching by excavating most relevant links with an adaptive link-ranking. A link tree data structure allow crawler to achieve wider coverage for a website as well as to eliminate bias on visiting some highly relevant links in hidden web directories. This paper is referred as base for proposed system. Existing crawler system works on hidden database. It uses Incremental site prioritizing that calculates out of site link of pages. It uses Form Focused crawler which doesn’t allow users to perform personalized search. It becomes difficult to efficiently search the result in less time, due to rapid growth in data on web. Proposed system works on site database. Ranking is calculated on the basis of the importance of pages using incremental site prioritizing that directly check query word on page content. It focuses on specific topic and performs domain classification for links which helps user to walkthrough all the relevant links. It also allow user to personalized search and they will get the result of query in their area of profession.

3. Dataset

In this system, we have used Google API as our site database from where we can perform link collection for our searching. This API allows developer to get web or image search results data in JSON or Atom format. In this system, we have taken link collection count as 20. We can take maximum count as per our requirement; but it will consume more time to classify links as relevant and irrelevant and delays the searching time.

4. Proposed system

The main idea of this system is to crawl a web page effectively to obtain relevant results for searched query. In first stage, Crawler performs “Reverse searching” and in second stage “Incremental prioritizing” is performed to match the query content within form. Then according to match frequency, crawler classifies pages as relevant and irrelevant pages and ranks these pages.

Site locating stage starts with a seed collection of sites from a site database. Collected links are given to system to start crawling. Query processing unit get user’s query as input on which it will decide the query is of which type, that is personalized query or normal query.

Link collection unit will collect link from site database and then it will perform reverse searching it match user query content in URL, then crawler classifies the links as relevant and irrelevant links, then Incremental prioritizing unit uncovers the collected site forms and matched the content of query on form, depends on matching, system is going to classify relevant and irrelevant links.

Page ranking is performed and on the basis of frequency of searched query in document, it will display high ranked results on result page. Domain classification is performed to show contribution of standard sources. Here personalize search perform searching according to user profile so it is easy to get accurate result to user.

If query is normal query, then reverse searching algorithm is performed to find out relevant and irrelevant links. Then this relevant and irrelevant links passed to incremental site prioritizing unit to rank the links according to its relevance. This results of ranking is passed to domain classification unit to present it in more understanding and useful manner. We get results in form of relevant links.

If query is personalized query, then reverse increment searching algorithm is performed. This personalized search is performed by considering user’s profession profile to obtain the relevant links. These results are passed to domain classification unit and we get results in form of relevant links.

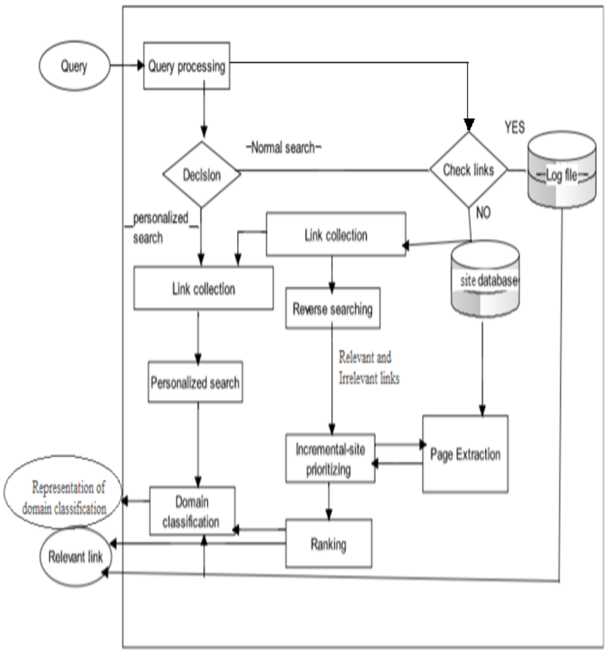

Fig. 1: Proposed System.

5. Proposed approach

In this system, we have implemented three algorithms for improving our searching technique which are given as follow:

Reverse Searching: In reverse searching, crawler is checking for searched keyword in URL and description of link based on that, this algorithm classified the link as relevant and irrelevant link.

Algorithm 1: Reverse Searching

```
Input: Seed sites to harvest web pages
Output: Relevant sites
1    while #candidate sites do
     //Pick a website
2    Site = SeedSiteCollection(siteDatabase, seedSites )
3    Links = extractLinks(link)
4    Page= compareUrl(link)
5    Relevant = classify (page)
6    if Relevant then
     list.add(page)
7    return list
8    End
```

Fig. 2: Reverse Searching Algorithm.

1) Incremental Prioritizing: The output of reverse searching is passed to incremental prioritizing. This algorithm helps to rank the link results, for that it open the web page from relevant link queue and searched for query keyword in that web page. Ranking is based on the frequency of searched keyword in that particular web page.

Algorithm 2:Incremental Prioritizing

```
Input: List of links which we get after performing reverse searching
Output: High priority Link
1   hQueue=list.CreateQueue (relevantLinks)
2   lQueue= list.CreateQueue(irrelevant Links)
3   while list is not empty do
    if HQueue is empty then
    hQueue.addAll (lQueue)
    lQueue.clear ()
4   End
5   if Relevan t then
    contentExtraction (site)
    output pageContent
    siteRanker.rank (pages)
    End
6   if pages is not empty then
    hQueue.add (pages)
    Else
    lQueue.add (pages)
    //Add pages in lQueue
7   End
```

Fig. 3: Incremental Prioritizing Algorithm.

2) Personalized Search: This algorithm helps to personalized search based on profession of user. During registration, this crawler note down the profession of user, which can be combine with searched query if user want to get the result in its field of profession.

Algorithm 3: Personalized Search

```
1   Get query keyword to search
2   Get user profession
3   Get query related link
    //Get links by combining query keyword and profession of user
4   Perform reverse-increment searching.
    //To classify link as relevant and irrelevant
5   Display result of personalized search to user.
```

Fig. 4: Personalized Search Algorithm

6. Performance evaluation

The proposed crawling framework is tested over Google API dataset based on that its effectiveness is evaluated. The proposed crawler is implemented in Java and to evaluate an
extensive performance of proposed crawling framework, it is compared with existing SmartCralwer mentioned in [6].
Goals include:

- Evaluating the reliability of proposed crawler in obtaining relevant websites.
- Evaluating the performance of proposed crawler using log file.
- Analysing the contribution of different standard resources in context to search result based on domain classification.
- Evaluating the performance of crawler in terms of personalizing search.The description of SmartCralwer and proposed crawler is given as follow. Both the crawlers are compared on the parameter of searching time taken by crawler to search keyword.
- SmartCrawler: We implemented algorithm strategies used in SmartCrawler framework over Google API dataset.
- Proposed Crawler: Proposed crawler uses Reverse searching algorithm and Incremental prioritizing algorithm to search the websites efficiently. The log file performance of this proposed crawler is also evaluated in terms of searching time.

Table I shows the time taken by SmartCrawler and proposed crawler to search the links. Figure 5 shows the performance of both the crawlers.

Table 1: Performance In Terms of Time Taken By Existing and Proposed Crawler Framework

Number of links in Link Collection	Time taken by following crawler in minutes		
	Smart-Crawler	Proposed crawler	Using log file-Proposed crawler
10	1.77	0.43	0.07
20	1.16	0.99	0.072
30	1.94	1.91	0.14
40	2.77	2.23	0.075
50	3.49	3.2	0.085
60	4.47	3.51	0.056
70	5.5	3.68	0.067
80	5.67	3.62	0.055
90	7.72	4.87	0.05
100	7.54	5.89	0.048

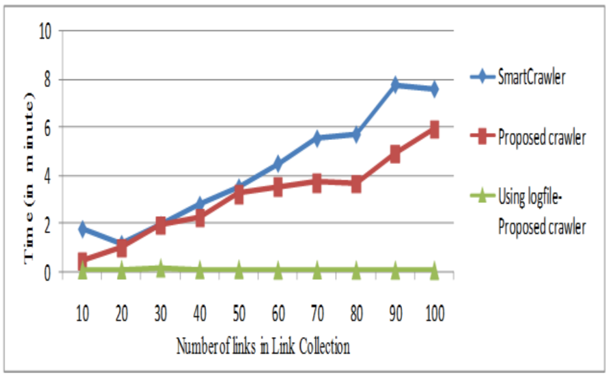

Fig. 5: Graph Showing Performance in Terms of Time Taken by Existing and Proposed Crawler Framework.

From above graph, we can conclude that proposed crawler performs better than existing crawler in terms of searching time. When proposed crawler is implemented using logfile, it performs much better than the SmartCrawler and proposed crawler. Table II shows the standard sources and their description which contributes in searching the links relevant to searched query. This are the six major sources which crawler preference during domain classification.

Table 2: Standard Sources and Their Description

Standard Sources	Description
Google	Web Search Engine
Wikipedia	Free Online Encyclopaedia
Linkedin	Business- And Employment-Oriented Service
Twitter	Online News And Social Networking Service
Facebook	American Online Social Media And Social Networking Service
Proprietary Or Undetectable	Open Source Site

Table 3: Shows the Contribution of Standard Sources in Percentage Based on Below Pie Chart, when "Class" Word is Searched

STANDARD SOURCES	CONTRIBUTION (IN PERCENTAGE)
GOOGLE	4.1%
WIKIPEDIA	3.4%
LINKEDIN	-
TWITTER	-
FACEBOOK	3.6%
PROPRIETARY OR UNDETECTABLE	88.9%

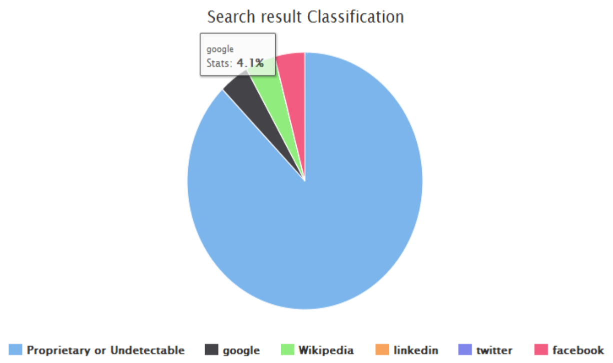

Fig. 6: Pie Chart, Which is Generated by, Proposed Crawler Showing Domain Classification, when “Class” Word, Which.

Helps to understand contribution of above-mentioned standard sources

Personalized search helps to customize the search engine and to provide results based on person’s interests and ranked the results accordingly. This crawler helps to personalize the search based on user’s field of profession. For example, when we search for “Class” word on any standard search engine, we will get output as links of definition of class, social class; but when we search same “class” word using this proposed crawler, we will output as link of Java class definition or related to computer background as registered user is Software Engineer. This helps to search result according to user’s profession.

7. Conclusion and future work

In this paper, proposed crawler efficiently searches relevant documents for user searched query. The crawler works in two stages i.e. Reverse searching and Incremental prioritizing. The ranking helps to get relevant documents from retrieved documents. Domain classification helps to know the contribution of links from standard resources for a particular searched query. Log file is maintained reduce search time of query result for previously searched query. Experimental results shows the effectiveness of proposed crawler, proposed crawler searches query results in less time than smart crawler and effectively personalized the search according user’s interest. When crawler is implemented using logfile, it shows much better performance. In future work, we can implement this application for E-learning: E-learning application can reduce the costs of education. Using E-learning application, we can increase productivity in terms of web searching as well as for training people. Domain classification can be improved to search and view results in efficient manner, so user can efficiently walkthrough all relevant links. Log file can be used in more efficient way for securing user’s confidential data as well as for maintaining its privacy.

References

[1] S. Chakrabarti, M. Berg and B. Dom, “Focused crawler: a new approach to topic-specific web resource discovery.” Computer Networks, 31(11):1623–1640, 1999.

[2] C. Sheng, N. Zhang, Y. Tao and X. Jin, “Optimal Algorithms for Crawling a Hidden Database in the Web”, Proceedings of the VLDB Endowment, 5(11), Pages: 1112–1123, Year: 2012.

[3] L. Shou, H. Bai, K. Chen and G. Chen, “Supporting Privacy Protection in Personalized Web Search”, IEEE Transactions on Knowledge and Data Engineering, Year: 2014, Volume: 26, Issue: two, Pages: 453 – 467.

[4] D. Kumar and R. Mishra, “Deep web Performance enhance on search engine”, International Conference on Soft Computing Techniques and Implementations (ICSCTI), Year: 2015.

[5] S. Shukla, “Improving the Efficiency of Web Crawler by Integrating Pre-Query Approach”, International Journal of Innovative Research in Computer and Communication Engineering, Vol. 4, Issue 1, January 2016.

[6] F. Zhao, J. Zhou, J. Zhou, C. Nie, H. Huang and H. Jin, “Smart Crawler: A Two-Stage Crawler for Efficiently Harvesting Deep-Web Interfaces”, IEEE Transactions on Services Computing, Volume: nine, Issue 4, Pages: 608 – 620, Year: 2016.

[7] S. Gupta and K. Bhatia, “A Comparative Study of Hidden Web Crawlers”, International Journal of Computer Trends and Technology (IJCTT), Volume 12 No. 3, Year: Jun 2014.

[8] M. Dincturk, G. Jourdan, G. Bochmann and I. Onut , “A model-based approach for crawling rich internet application”, ACM Transactions on the Web, Volume- 8(3):Article 19, 1–39, Year: 2014.

[9] Dr. S. Vijayarani1, Ms. J. Ilamathi and Ms. Nithya, “Preprocessing Techniques for Text Mining - An Overview”, International Journal of Computer Science & Communication Networks, Vol. 5(1), 7-16.

[10] P. Wu, J. Wen, H. Liu and W. Ma, “Query Selection Techniques for Efficient Crawling of Structured Web Sources”, 22nd International Conference on Data Engineering (ICDE'06), Year: 2006.

[11] J. Cope, N Craswell and D. Hawking, “Automated discovery of Search Interfaces on the web”, Conferences in Research and Practise in Information Technology, Volume: 17, Year: 2003.

[12] K. Chang, B. He, C. Li, M. Patel and Z. Zhang, “Structured databases on the web: Observations and Impliations”, ACM SIGMOD Record, Volume: 33, Issue: [3], Year: September 2004, Pages 61 – 70.

[13] L. Gravano, P. Ipeirotis and M. Sahami, “Query- vs. Crawling-based Classification of Searchable Web Databases”, Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, Year: 2001.

[14] S. Duamais and H. Chen, “Hierarchical classification of web content”, ACM Publication, Year: 2000.

[15] J. Jiang, X. Song; N. Yu and C. Lin, “Focus learning to crawl web forums”, IEEE Transactions on Knowledge and Data Engineering, Year: 2013, Volume: 25, Issue: 6, Pages: 1293 – 1306.

[16] J. Cho, H. Garcia-Molina and L. Page, “Efficient Crawling Through URL Ordering”, Jounal of Computer Networks and ISDN Systems, Volume: 30, Issue: 1-7, Year: April 1, 1998, Pages 161-172.

[17] R. Botafogo and B. Shneiderman, “Identifying aggregates in hypertext structures”, Proceeding HYPERTEXT '91 Proceedings of the third annual ACM conference on Hypertext, Pages 63-74, Year 1991.

[18] S. Liddle, D. Embley, D. Scott and S. Yau, “Extracting Data Behind Web Forms”, Proceedings of the 28th VLDB Conference, Hong Kong, China, Year: 2002.

[19] A. Bergholz and B. Childlovskii, “Crawling for domain- specific hidden web resources”, Proceedings of the Fourth International Conference on Web Information Systems Engineering, Year: 2003, Pages: 125 – 133.

[20] H. Dong and F. Hussain, “Self- Adaptive semantic focused crawler for mining services information discovery, IEEE Transactions on Industrial Informatics, Year: 2014, Volume: 10, Issue: 2, Pages: 1616 – 1626.