ISSN: 2321-7782 (Online)
Volume 2, Issue 9, September 2014
International Journal of Advance Research in Computer Science and Management Studies
Research Article / Survey Paper / Case Study
Available online at: www.ijarcsms.com

Fuzzy Ontology Based Schematic Web Crawling with Self Adaptive Clustering on Extraction

Dr.K.Swathi[1]
Professor and Head
Department of Computer science &Engineering
Cauvery College of Engineering & Technology
Trichy (TN) – India

A.Gnana Prabha[2]
Scholar
Department of Computer science &Engineering
Cauvery College of Engineering & Technology
Trichy (TN) – India

R.Vijayanathan[3]
Senior Librarian and Head
Department of Library and Information Science
Cauvery College of Engineering & Technology
Trichy (TN) – India

Abstract: ***Due to the rapid growth of web pages in internet, discovering relevant content from the web is one of the main challenges in deep web crawling. Web crawlers play the vital role in search engines. Most of the web crawlers use the hyperlinks only for crawling. The html form urls and JavaScript based urls in the web pages can also have the relevant content for the keywords. While adding these urls also for crawling, the number of pages to be crawled will be increased which will increase the crawling time. This is one of the major challenges in deep web crawling. This research lie in the design for vocabulary based ontology learning to fetch the relevant content from the web by overcoming these challenges.***

Keywords: Hidden web, EIT Path, Multithread Crawler, Page Classification, Deep Web, ITF regex, Ontology.

I. INTRODUCTION

A web crawler is a program or automated script which browses the World Wide Web automatically in a methodical manner. This process is known as Web crawling or spidering. Search engines use spidering to provide the updated data. Web crawlers are mainly used to create a copy of all the visited pages for later processing by a search engine, which will index the downloaded pages to provide fast searches [1].

Internet has a wide expansion of information. Finding relevant information from web requires an efficient mechanism. Web crawlers provide that scope to the search engines. Most of the crawlers follow hyperlinks. In many cases, the important and relevant information will be hidden in forms [6][7][8]. Nowadays hosting the data in databases is increased. Those data will be queried using HTML forms. By simply following the hyperlinks [4][5], these contents can not be retrieved and they are hidden to the crawler point of view and they are referred as deep web.

How Crawler Works:

The workflow of the crawlers is as follows

1. Crawling starts with the seed url to visit and downloads the Web page.

2. Parse through the downloaded page and retrieve all the links.

3. For each link retrieved, repeat the process until the maximum number of pages reached.

II. System Components

The differences of the existing system and the proposed system can be summarized as follows.

1. In our previous work, the downloaded pages may be not the relevant one and the vocabulary of the ontology to be enriched [2]. In our research, the relevant content will be fetched for the keyword by using appropriate algorithms.

2. In existing system, the crawler will collect all the links and download all the pages and process them ie) post-query. In our research, we are using pre-query method ie) Downloading only the relevant pages with multithread downloader.

3. Our previous research work utilized the ontologies designed for health care, transport and mining service domains [2]. In this research, we design ontology's for other service domains in order to obtain the wider searching scope of the crawler.

Resource Locator) as input. It starts searching or fetching the information of that URL by initiating the threads process. Threads will be running continuously to get all the URI's information and stores them in a queue.

At the time of downloading each URI, it puts in threads view after completion of download process, it just transfers the completed URI into the request phase. If any errors occur while fetching any of the URI corresponding to the URL, they just listed in error view phase. The information like internet connection status, how many URI's downloaded how many errors occurred, how much memory available, what's the CPU usage these things are displayed in the status bar.

Configuration Module:

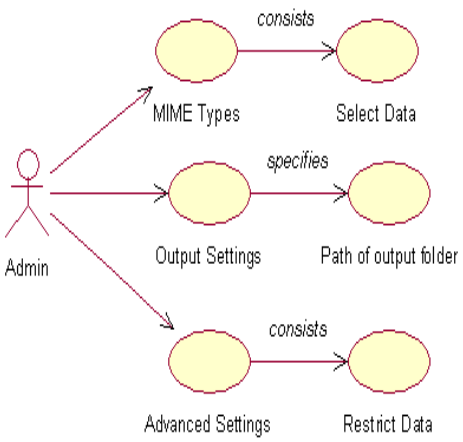

Figure 3: Configuration

Mime Types:

In this section, we can set what kind of data we need to extract from the particular URI like whether we need string data, boolean data and images information or not.

Output Settings:

In this section, we mention the output folder name where we need to store the content about the website is fetched.

Advanced Settings:

These are the settings made by the user in order to restrict some kind of website like with domain name as .NET, .AC.IN like this.

Multithreaded Downloader:

Here the multithreaded downloader is responsible for starting threads and obtaining the information about the website to be fetched. Multithreaded downloader starts threads with the given url and it collects all URI's and placed them in one queue. Each

and every thread starts with one Uri in the queue. After completing the URI, it moves to the next URI's in the queue. The URI's will be arranged in the queue based on some criteria like most visited pages and page rank. In this module one folder creates in the user desired path and the files created with the URI names having the static information.

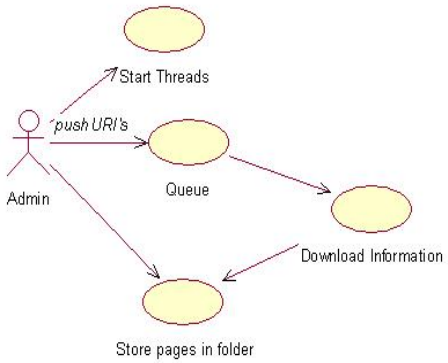

Figure 4: Multithreaded Downloader

III. PROCEDURE

Step 1: Start crawling from the root url

Step 2: Collect all hyperlinks, form urls and javascript based urls from the web page and filter the urls which are relevant to the keywords by using Stemming algorithm and Stop Word.

Step 3: The collected urls to be placed in the queue (Visiting URL).

Step 4: If any of the url in the queue is already visited, remove the url from the Visiting URL list.

Step 5: Sort the urls in the queue based on the criteria (most visited page and page rank) by using Best First Search algorithm.

Step 6: Multithread Downloader initiates threads from the urls in Step 4

Step 7: Extract the required contents and cluster them based on the type using regular expressions and place them in the output folder location .

Step 8: Mark the URL as Visited and remove it from the Visiting URL list.

Step 9: For each url in the thread, follow the Steps 2 to 4.

Step 10: Terminate the process when the maximum no of pages to be crawled is reached.

Step 11: Calculate the Harvest Rate, Precision and Crawling Time.

Stemming and Stop Words:

Stemming algorithm is used to check the content relevancy for the keyword.

Information Retrieval is essentially a matter of deciding which documents should be retrieved in a collection to satisfy the need of the information. Keyword contains one or more search terms. Hence, the retrieval decision is made by comparing the terms of the query with the index terms (important words or phrases) appearing in the document itself. The decision may involve estimating the degree of relevance that the document has to the keyword. For example the words Connecting, connection and connects can be stemmed to the root word "connect".

Stop words do not have important significance in the search queries. Those words should not be indexed and should be filtered out for before processing. Example for the Stop Words are "a", "the", "is" and "to".

Best-First Search Algorithm:

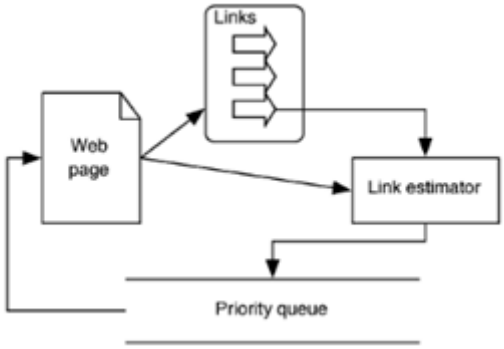

Figure 5: Best First Search

```
BFS(topic,starting_urls){
   foreach link(starting_urls){
      enqueue(frontier,link,1);
   }
  while(visited < MAX_PAGES){
      link := dequeue_top_link(frontier);
      doc := fetch(link);
      score := sim(topic,doc);
      enqueue(frontier,extract_links(doc),score);
      if(#frontier > MAX_BUFFER){
          dequeue_bottom_links(frontier);
      }
   }
}
```

Best-First crawlers have been studied by Cho et al. [1998] and Hersovici et al. [1998]. The idea of BFS is that given a frontier of links, the best link to be selected for crawling according to some estimation criteria. Different Best-First strategies of increasing complexity and (potentially) effectiveness could be designed on the bases of increasingly sophisticated link estimation criteria. The link selection process is guided by computing the relevancy between the topic's keywords and the source page for the link. Then the best estimated URL is selected for crawling. Minimum similarity score are removed from the frontier if necessary in order to not exceed the limit size MAX BUFFER. Best-First crawling is the most powerful one due to its simplicity and efficiency.

IV. EVALUATIONS

Harvest Rate:

Harvest rate is one of the important parameter to evaluate the performance of the system.

$$\text{Harvest Rate} = \frac{\text{No of relevant pages downloaded}}{\text{Total no of pages downloaded}}$$

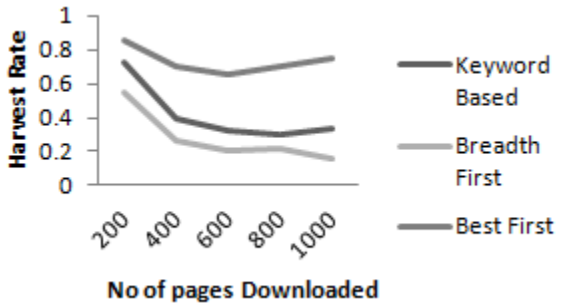

Figure 6: Harvest Rate

Harvest rate of the crawler which uses Best-First search is comparatively better than the other crawlers[3].

Precision:

Precision is measured based on the number of relevant pages downloaded against the total number of pages.

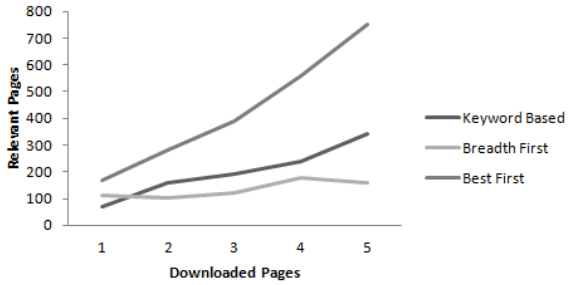

Figure 7: Precision

From the above graph, we observe that the number of relevant pages downloaded by our system is higher than the others.

Crawling Time:

The crawling time for the proposed system is comparatively less than the other crawlers. Since the proposed system consider the relevant links only for crawling by pre-query method. The irrelevant links will be ignored for crawling instead of identifying them after processing.

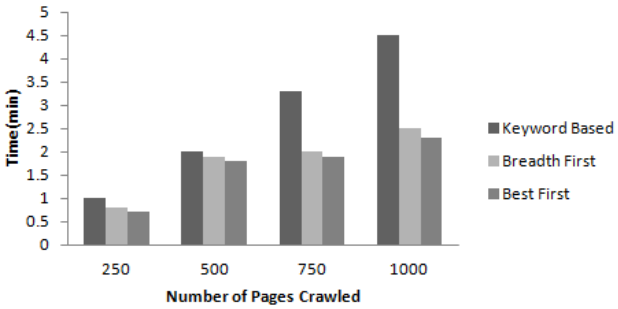

Figure 8: Crawling Time

V. Conclusion and Future Work

The conclusion is summarized as below

1. The proposed system has the capability to detect the Entry URL which increases the coverage.
2. It uses an emulator technique to detect even the JavaScript-based URLs and html form URLs which most of the existing Crawlers cannot do.
3. It uses a Freshness First Strategy for performing the online crawling which helps to retrieve newer pages prior to old ones which is advantageous in a situation where the available resources are limited.

4. By using pre-query method, the irrelevant pages are ignored for crawling. Multithreaded downloader reduces the crawling time and the content is extracted efficiently and clustered by its type. Future work includes discovering the new threads and refreshing the crawled threads in a timely manner. Conducting more comprehensive experiments to further verify our approach and improve upon it. It is necessary to enrich the vocabulary of the ontology by surveying those unmatched but relevant service descriptions.

References

1. http://en.wikipedia.org/wiki/Web_crawler
2. Hai Dong,Farookh Khadeer Hussain (2014) "Self-Adaptive Semantic Focused Crawler for Mining Services Information Discovery"-IEEE Transactions on Industrial Informatics Vol.10,May 2014
3. Debajyoti Mukhopadhyay, Arup Biswas, Sukanta Sinha – "A New Approach to Design Domain Specific Ontology Based Web Crawler" - 10th International Conference on Information Technology
4. Wei Huang, Liyi Zhang, Jidong Zhang, Mingzhu Zhu-(2009) "Focused Crawling for Retrieving E-commerce Information Based on Learnable Ontology and Link Prediction" - 2009 International Symposium on Information Engineering and Electronic Commerce
5. Rodrigo Campos, Oscar Rojas, Mauricio Marın, Marcelo Mendoza (2013)"Distributed Ontology-Driven Focused Crawling" - 2013 21st Euromicro International Conference on Parallel, Distributed, and Network-Based Processing
6. Michael K. Bergman, (2004)"The Deep Web:Surfacing Hidden Value", In Proc. Of BrightPlanet-Deep Content, Septermber, 2004.
7. Lee Ratzan, "Mining the Deep Web: Search Strategies That Work", Computerworld, 11th December, 2006.
8. K. V. Arya, K. V. Arya Baby Ramya Vadlamudi (2012) "An Ontology-Based Topical Crawling Algorithm for Accessing Deep Web Content" - 2012 Third International Conference on Computer and Communication Technology

AUTHOR(S) PROFILE

Dr.K.Swathi, obtained her under-graduation in B.E., (Computer Science & Engineering) from Bharathidasan University, Trichy in 1999. She obtained her M.E. degree in Computer and Communication Engineering from Anna University, Chennai in 2004.She obtained Ph.D degree in Faculty of Information & Communication Engineering from Anna University, Chennai in 2014. Presently, she is working as Associate Professor in Computer Science & Engineering, Cauvery College of Engineering and Technology, Trichy in 2008. She has 14 years teaching experience and also she had attended many workshops, seminars and conferences on Research issues in Image processing. She has published papers in international journals and presented papers in various Conferences. She is the life member of Indian Society for Technical Education (ISTE). Her areas of interest include Image processing, Data mining, network security and Software engineering.

A.Gnana Prabha, received her B.E in Electronics and Communication Engineering from Anna University, Sri Sai Ram Engineering College,Chennai in 2006 and doing her M.E in Computer Science Engineering in Cauvery College of Engineering and Technology,Trichy. Her area of interest includes data mining and open source technologies.

Vijayanathan.R., received his Master of philosophy Library and Information Science from Annamalai University in 1999 Also he obtained his post-graduate degree in Master of Economics in Bharathidasan University in 1996.He is working as a Sr. Librarian, Department of library and Information Science in Cauvery College of Engineering and technology, Trichy from 2009.

He has published 13 research papers in National and International journals. His areas of interested are networking; Cloud computing, IC Technologies, Environmental study, Library Automation, Webometric study, Scientometric study, Bibliometric analysis, and citation study.

