IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 2, Ver. VII (Mar-Apr. 2014), PP 45-50
www.iosrjournals.org

Fetching the hidden information of web through specific Domains

Usha Gupta
(Assistant Professor, Computer science department, Aravali College of Engineering and Management/ Maharishi Dayanand University, India)

Abstract: *Now days, most of the people find the information over the internet by visiting various number of web pages and also by following the links under those web pages. A user has the ability to browse for data through the web pages he/she requires by following these web links. Even though, if the web page requested is not made public. There are various resources over the internet; most of them are hidden behind the database. Because of it, the user is only able to access it after making a query in to the database. With the help of search engines, this problem of finding required and necessary data is been resolved. This paper presents a method to retrieve such hidden data from web.*

Keywords: *DSIM, Request URL generator, Response agent, SFE, SFI.*

I. INTRODUCTION

According to some researches, there is a very large amount of data (say millions of data) in the hidden web [1] and this number is still increasing over the time. Which means that, there is a need for a mechanism to retrieve such information [2] from hidden web for the user? The entry point to such hidden webs is through the forms [3, 4]. So whenever a crawler encounters such a form, then it needs to know the domain in which it acquired the form and the type of content that is required for filling up the form [1, 5, 6, 7].

There are many factors which makes this task of fetching hidden information quite complicated [1, 5, 6, 7] are as the web is constantly changing its resources. New resources are being added to the web and old ones are being modified or removed as per requirements. So there required a solution to such problem for crawling the data deeply inside the web by letting the method to find the hidden web sources automatically.

Another issue is that the web offers different form structures means no web forms has same structure. For example, if a user is searching for a book over the web, he/she will encounter different types of forms on different web sites. Which in turn will have different schemas in the form structure which ultimately added to the complexity of the form [1, 5, 6, 7]. According to recent study, there are like 307,000 deep websites [1] and an average of 4.2 query interfaces with the help of forms for those websites [1, 8]. Hence to look for data on such websites needs a very sophisticated and efficient method.

So there rises a need for a search process that must be efficient enough by possessing the capability of avoiding visits to the unwanted portions of web [9, 10]. and also the search process should search the web by keeping the focus on data to be searched also known as focused crawling, which tries to retrieve only a subset of links on the web that are relevant to the query's topic or data. Hence requires better indexing. for example: in a shopping website, user enters the keyword in form presented to him/her when the search is made for the first time, and there are another forms that can be used to sort the results of search by price values or popularity of item and so on.

The approach presented in the paper will look for the first page on the shopping website and will then extract all the forms presented over there and will make searches accordingly in order to eliminate other unwanted forms.

1.1 ASSUMPTION

Here in this regard, there are taken certain assumptions before designing the algorithm as given below:

i. The domain is been specified earlier.
ii. The set of websites found in that specific domain is written down initially. These websites were all from that single specified domain.

II. SYSTEM ARCHITECTURE

The prototype of domain based web crawling consists of five major components: *The Search Form Identifier, The Search Form Extractor, Domain Specific Internet Mapper, Request URL generator and Response Agent* components. Fig.1 illustrates the system architecture to crawl domain based Hidden web.

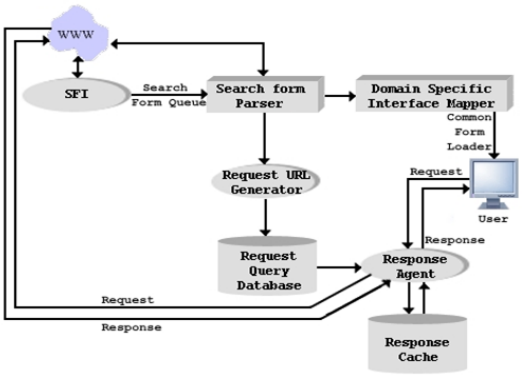


Fig: 1 system architecture to crawl domain based hidden web

2.1 SEARCH FORM IDENTIFIER (SFI)

As the World Wide Web have millions of Web pages. Some of which will have forms. These forms will act as an entry point for the extraction of information hidden behind them. This component will be scanning millions of Web pages to find such kinds of form. However, different domain Websites will have different types of forms in it. For e.g. a hospital Website might have a form used to find information about the patients and other health diseases. A Website related with weather information, will have a form used to get the weather and other related information for that particular area. Search form identifier (SFI) will be analyzing most of the Websites and will extract the forms that are related with shopping. The part of going to the Web and looking for maximum URLs and then identifying search forms related to specific domain. The basic architecture of SFI is show in Fig.2

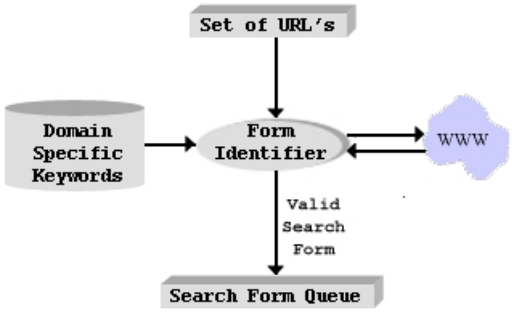


Fig: 2 search form identifier

SFI will be analyzing each and every Web page that the crawler comes across. It will identify the valid search page, that can be used to retrieve information or not. Like for e.g. product name or type of product looking for. Also, SFI will help us to find the difference between a registration page and a query page also. This is one of the most important components while looking into the shopping Web pages since most of the shopping Web pages also have registration forms.

The main job of the SFI is to look for html forms on a Web page and see if those forms are actually the search forms and not registration forms. Based upon the survey been made on large number of Websites, it is been observed that these forms which are designed for search function has specific type of format. Normally, the submit buttons of these forms are found to be named as "GO" or "Search". Besides this, they will contain one common structure of like having a text field named as "Keywords" or "ks" (Keyword

Search) or no name at all and a submit button. However, this find of format of having text field with that particular name and a submit button are not found in all the Websites, but they are the most common structures found and it also results into large number of successful search results.

Once we get the list of forms, we will start iterating through it one after another. While iterating through each and every form, we will do the following actions.

1. Look for a keyword "search" or "go". If we encounter any of these keywords, it will mean that we found the submit button of the form. If the form does not have any of these keywords, it will mean that the form does not possess any submit button. Hence we can go to step 2.
2. Look for images that have keywords like "search" or "go". Shopping Websites are now more interactive and attractive. Hence, in place of using traditional looking submit button, it might be possible that they have used an image button with name search or go. And looking at the number of URL's found on the Web for shopping,

around 60 % of the Websites will use image or any other method rather than a plain submit type button. Once, we find such kind of input type button, we can proceed to step 3 or we can pick up the next form and proceed to step 1.

3. After finding the submit button, now it's time to look for text field. The text field will be helpful in filling up the keywords that user has specified. Once we find the text field we are all set to fill it up with the user keywords (done using text field set Attributes ("user keywords") and clicking on submit button. It is done through submit_Button_click ( ) the function submitButton.click ( ) will return a new Web page object that will have the result page from that particular Website.

After the result page is been obtained, we pass it on to Result Analyzer to analyze the results from that particular Web page.

Algorithm SFI ( )

step 1: Initialize d_s_K hash with domain search form keywords
step 2: repeat steps 3 to 7 while(URL Queue not Full**)**
step 3: Extract initial Webpage;
step 4: Check if (d_s_k[webpage Keyword] == 1**) then**
step 4(a): Webpage_linkQuese = Extract Links from Webpage;
step 5: while(Webpage_link not empty**)**
step 6: Identify Valid Search form
step 7: Insert valid search form into Search_form_queue;

2.2 SEARCH FORM EXTRACTOR (SFE)

A form can have various kinds of input fields. Once the Web page is detected from where the form can be filled and made a search; extractor is called. The main job of the Search form extractor (SFE) is to look for various types of forms found on that Web page. Once a suitable form is found from where the query can be made; that particular form is been separated out and passed it on to the domain specific Internet Mapper and Request URL generator.

2.3 DOMAIN SPECIFIC INTERFACE MAPPER (DSIM)

The *Domain Specific Interface Mapper* (DSIM) finds the semantic mappings between the components of different web interfaces of the same domain i.e. all the interfaces belong to the same domain such as airline domain. The main inputs to this Interface mapping system, are two interfaces A and B comprising of a number of components i.e. $\{n_1, n_2 \ldots n_p\}$ and $\{n'_1, n'_2 \ldots n'_q\}$ respectively.

DSIM uses a *Search Interface Repository* that is the repository for domain-specific search interfaces. It also provides an extensible domain-specific matcher library to support multi-strategy match approach. The multi-strategy match approach uses different matching strategies like *fuzzy matching, domain-specific matching* etc that are executed independently. The SVM (support vector machines) Generator in DSIM is used to provide matrices of mapping that are identified by the matching library. The *SVM Selector* generates the appropriate mapping which can be used as the output. The DSIM also uses a *Mapping Knowledgebase* which stores the important semantic mappings so that they can be used further when after sometime our search interface repository would be updated.

The multi-strategy interface matching is carried out by the DSIM in three phases: *parsing, semantic matching and semantic mapping generation*.

This component is responsible for loading Common search interface at user-end from search interface repository.

2.4 REQUEST URL GENERATOR

The real time search mode is when the WebCrawler has to find the relevant Web pages based on the users query. The key thing in such kind of search mode is to follow the URLs from the Web pages that are similar to what the user wants, and thus these URLs will lead to more relevant Web pages. Such a kind of approach roughly captures the way people navigate the Web: they find a Web page about a topic related to what they are looking for and follow links from there.

A form can have various kinds of input fields. Once the search form is parsed by the SFP; this component is called. The Request URL Generator creates the request query or set of request queries to extract information from the hidden web. Once the suitable request query or set of request queries is created, then the combination request query get stored into databases. The Reponses Agent runs the user's query against its index.

In order to create the request query, request URL Generator first identify the protocol used to access content, including HTTPS, HTTP, FTP and Gopher. Request URL Generator provides the base URL of all the possible requests for submission. The request query stored in Request Query Database has the following format:

Request Query Format:

<Protocol><Base URL>? attribute1=<a1_value>& attribute2=<a2_value>

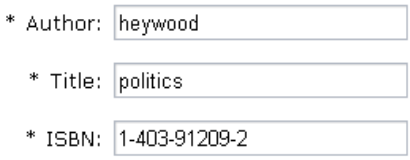
Algorithm Request_URL_generator ( )
step 1: repeat steps 2 to 6 while (Search Form queue not Empty**)**
step 2: Request_query = "<Protocol><baseURL>?";
step 3: Set I=0;
step 4: repeat step 4 while(Form Attribute**)**
step 4(a): Request_query=Request_query."attribute [i]."=<attribute_value>."&";
step 5: Set i = i++;
step6 : Store request_query into Database;

2.4.1 USER

User fills the request search form and sends it to Response Agent, and fills each value corresponding to its attribute weather compulsory or optional.

2.5 RESPONSE AGENT

Once the user fills up the common forms generated by DSIM and send the request to Response Agent, the results of the forms are been obtained in the form of Web pages. For every possible URL and its request query format generated by the Request URL Generator is substituted for original value corresponding to its attribute, for example, URL: http://www.abebooks,comfinal query may take the following format.
http://www.abebooks.com/servlet/SearchResults?tn=politics&sortby=2&isbn=1403912092&sts=t&an=heywood&vci=837705&bi=0&bx=off&y=7&ds=30&x=68
in the above example, al the fields of the link will be concatenated by the web browser in one string and will be appended at the end of URL string without any blank space.



Input form of above string

Once the search request submitted by User to the Response Agent it will parse request in understandable form of different attributes (because earlier all attributes were concatenated in one string by User's web browser). Simultaneously, Response Agent is searching for all different attributes in Request Query Database and it fills all values submitted by User in each search form generated URL, it also checks for weather desired result is presented in Response Cache or not if present then return matched result to user, if not then it will send request to Web till all search form generated URLs processed. Now it is the job of the Response Agent to analyze each and every result on that particular Web page and then extract the necessary results from it. This can also be explained by following Flow-Chart:

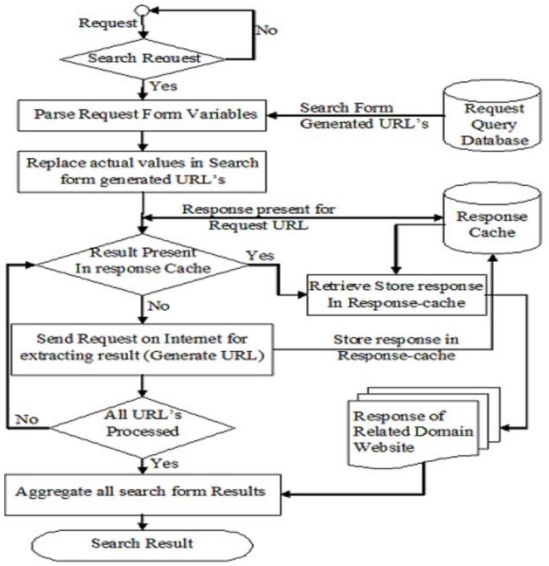


Fig: 3 flow chart for domain specific crawler architecture

These agents are all ran as separate process. Since there might be few issues like server been down or the network having a bottleneck. A typical WebCrawler contains we { by picking up a URL from the input file and submit it to Web Client. The Web Client will return an Html Page object.

The input to the Response Result is a request received from the common form loaded by DSIM. These results will be evaluated depending upon the type of search made. If a particular Web page has results that are more in number corresponding to the other ones then we can say that particular Web site is more likely to be the right one and hence, if similar keywords search will be made on that particular Web site then the search results will be more likely to be the right ones. Hence, Response Agent will play an important role to determine the accuracy of the WebCrawler. The result analyzer will keep the search result into Response Queue Database to achieve higher response time.

Algorithm Response_agent ( )
step 1: repeat steps forever
step 2: check if(User Search Request**) then**
step 2(a): Read request query databases for request queries;
step 2(b): Replace attribute values into request query;
step 3:check if (result present in Response Cache Database) **then**
step 3(a): Retrieve response from the Response Cache Database;
step 3(b): Return Retrieved response;
step 4: else
step 4(a):request to www using request query;
step 4(b): Retrieve response;
step 5: Store retrieved response into Response Cache Database;
step6: Return Retrieved response;

III. DEPLOYMENT

Similar in operation to search engines on the Web, the deployment of this architecture on the Web involves three tiers: the user tier, the Common Form Loader & Response Agent (CFL&RA) tier, and crawl tier.

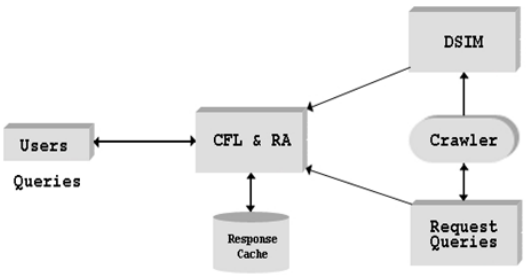


Fig: 4 deployment framework

In this particular framework crawler process the request in a decentralized fashion, the information tier consists of resource collections and added request Query and manage response cache.

The CFL&RA interface tier is the request & response tier, CLF unit is responsible for loading search form at user end and response agent unit is responsible for user request & response. The user tier posts the query and manages the results from the Request Agent interface system. The Response cache acquires the recent result for quick turnaround time against user queries.

IV. CONCLUSION

This paper concludes that the hidden Web as a system for the organization and discovery of hidden information behind search form. It emphasizes the values of extracting hidden resources mechanism which works behind common search form generated across one domain. What has been demonstrated is not a new approach to the hidden web. What is new is the use of an integrated information retrieval technique using request quest and domain-specific Interface Matching Library to achieve high performance and more accurate results.

This paper also described the challenges that must be overcome to integrate the information retrieval technique and domain-specific Interface Matching Library. The first problem is to identify the valid search form from the web and extract the search form from web pages. Then we must be able to integrate all the search form and create one common search form, which can be used for searching hidden web resource across various web resources related to that domain, but since the Web is a distributed system with dynamic information on a multitude of sources, a single solution is unrealistic.

V. FUTURE SCOPE

The architecture of crawler been presented here to crawl domain based hidden web. It is been designed to search within single specific domain. This architecture will execute queries on various Websites related to that domain and will gather the necessary results from them. Using this architecture, the Hidden Web Crawler can further be extended to enhance its performance by using a knowledge base.

It should be emphasized that the results obtained from by submitting search use of the data from a various websites and to simulate the request as the Single Webpage result. To fully realize the potential of this architecture, the crawler must be configured accurately by defining domain related keyword and by providing domain related URLS.

One of the major issues is the consistency of the resource over the hidden web is the template in which search page result is defined. In particular, when resources are obtained from different sources, consistency cannot be guaranteed. Information retrieved from one source could be inconsistent with another source. Although currently there is no standardized approach in maintaining consistency of the information retrieved, some general approaches like by define template or by reading RSS Feed file for extracting information from the listing/result page. Further it can be extended to support the web sites written in Ajax and JavaScript.

In conclusion, this paper has elaborated a framework to integrated information retrieval technique using request quest and domain-specific Interface Matching Library. The potential of the Web Crawler in such aspects have been explored in an integrated hidden web resources using a request queries set and common search form.

ACKNOWLEDGEMENTS

Words are often too less to reveal one's deep regards. An understanding of the work like this is never the outcome of the efforts of a single person. I take this opportunity to express my profound sense of gratitude and respect to all those who helped me through the duration of working on this theme.

First, I would like to thank the Supreme Power, the GOD, who guided me to work on the right path of life. Without his grace, this would not have been possible.

I take pride of myself being daughter of ideal great parents whose everlasting desire, sacrifice, affectionate blessing and help without which it would have not been possible for me to complete my studies. I also have been very thankful to my loveliest kushagra for his cheerful activities for enchanting me and also helping me in relieving the stress. Most importantly, I would also take the opportunity to thank my elder brother for guiding me so well and my di and sandy for their encouragement, support and love. Lastly I would also like to thank my guide of M tech dissertation for his support and guidance.

REFERENCES

[1] Barbosa, L AND Freire, J.Searching for Hidden-Web Databases, Eight International Workshop on Web and Databases, 2005
[2] Kobayashi, M. AND Takeda, K.Information Retrieval of the Web, ACM. Computing Surveys, 2000
[3] Barbosa, L AND Freire, J. Siphoning Hidden-Web Data through Keyword-Based Interfaces. In Proc. of SBBD, 2004
[4] Raghavan, S AND Garcia-Molina,H. Crawling the Hidden Web. In Proc. Of VLDB, 2001
[5] He, B AND Chang, K.Statistical Schema Matching across Web Query Interfaces. In Proc. of SIGMOD, 2003
[6] He, H, Meng, W, Yu, C AND Wu, Z. Automatic integration of Web Search interfaces with WISE-Integrator. VLDB Journal, 2004
[7] Wu, W, Yu, C, Doan, A, AND Meng, W.An Interactive Clustering- Based Approach to Integrating Source Query interfaces on the DeepWeb. In Proc. Of SIGMOD, 2004
[8] Chang, K., He, B., Li, C., Patel, M., AND Zhang, Z. Structured Databases on the Web: Observations and Implications. SIGMOD Record, 2004
[9] Shkapenyuk, V. AND Suel, T. Design and implementation of high Performance distributed WebCrawler, Proceedings of the 18th International Conference on Data Engineering, IEEE, 2002
[10] Fei, L., Fan-Yuan, M., Yun-Ming, Y., Ming-Lu, L. AND Jia-Di, Y.Distributed High-Performance Web Crawler Based on Peer-to-Peer Network, Parallel and Distributed Computing, Applications and Technologies, 2004