A NOVEL APPROACH TO INTEGRATED SEARCH INFORMATION RETRIEVAL TECHNIQUE FOR HIDDEN WEB FOR DOMAIN SPECIFIC CRAWLING

Manoj Kumar[1], James[2], Sachin Srivastava[3]

[1]Student, M. Tech. CSE, SCET Palwal - 121105, MD University, Rohtak, (India)

[2]Department of Computer Science and Engineering,
SCET Palwal - 121105, MD University, Rohtak, (India)

[3]Department of Computer Science and Engineering,
SCET Palwal - 121105, MD University, Rohtak, India

ABSTRACT

The traditional web crawlers retrieve contents from only the "Surface web" and are unable to crawl through the hidden portion of the Web containing high quality information which is dynamically generated through querying databases when the queries are submitted through a search interface. For Hidden web, most of the published research has been done to identify/detect such searchable forms and make a systematic search over these forms. One approach is based on a Web crawler that identifies, analyzes search forms and fills them with appropriate content to retrieve maximum relevant information from the deep web. A critical problem with this is the search interface integration followed by the integration of the results obtained as a result of parallel request quest. This paper proposes a technique to detect and construct an integrated query interface that integrates a set of web interfaces over a given domain of interest. It provides users to access information uniformly from multiple sources. The proposed strategy does that by focusing the crawl on a given topic; by judiciously analyzing the domain specific matching, mapping information/key attributes which leads to web pages containing domain specific search forms in integrated manner. The interface mapping and matching library, semantic knowledgebase specific to the domain play a very key role here and demand for continuous evolution, improvement to make hidden web crawling effective and usable.

Keywords*: *Hidden Web, Search Form Identifier, Search Interface Parser, Domain Specific Interface Mapping and Matching library, Crawler

I. INTRODUCTION

The amount of information present on web is huge of the order of around 800 million webpages and still growing[1]. Due to its vastness most of the information is gathered by us through search engines.A search engine classifies the Search results by keyword matches, link analysis, or other mechanisms perhaps not entirely clear to a front end user. These initial search results are later on fine-tuned by individual users through supplying additional keyword phrases or restricting selection to few Web pages for the results. While people usually rely on search engines to search information from the Web, the results obtained can be variable and often additional browsing must be done within the results. Even powerful commercial search engines such as Google, Yahoo,

AltaVista etc., can have problems finding relevant results for the users. About 80% of the Web users uses search engine toretrieve information from the Web[2].

The WebCrawler of these search engines are expert in crawling various Web pages to gather huge source of information. However, majority of the WebCrawler of these search engines crawls only what we call as surface Web. Whenever any WebCrawler visits a Web server, they gather information of all the Web pages been stored on that Web server. These traditional search engines cannot retrieve contents from the deep Web since these pages do not exist until they are created dynamically as the result of a specific search. They are not able to penetrate deep into the Web server to access their databases and use the Web services served by that particular Web server.Because traditional search engine crawlers cannot search beneath the surface, the deep Web largely remains hidden. This database information is often hidden placed behind HTML forms [9] and Web services interfaces, and it is only published, on demand, in response to users' requests [8]. Lawrence and Giles estimated that 80% of all the data in the Web could only be accessed via form interfaces[3]. According to studies conducted in 2000, deep Web contains 7500 terabytes of information in comparison to 19 terabytes on surface Web[4]. As per studies, there are more than 200,000 deep Web sites[4].Public information on the deep Web is currently 400 to 550 times larger than the commonly defined World Wide Web[4].Estimates – based on extrapolations from a study done at University of California, Berkeley – show that the deep Web consists of about 91,000 terabytes[5][6]. By contrast, the surface Web (which is easily reached by search engines) is only about 167 terabytes[5]. Besides the larger volume than the surface web, the deep web also has a higher quality and a faster growth rate[4]. Currently, there exist a number of standard tools, such as search engines and hierarchical indices that can help people in finding information. However, a large number of the web pages returned by filling in search forms are not indexable (hence also known as outside the Pre-indexable Web) to most search engines since they are dynamically generated by querying a back-end database.

Consider a use case where the user is looking for information about used cars before he can actually buy it online or by directly visiting the dealership/individual. He would want to know the price of car, make, color, kilometers/miles driven already etc… however this above information only exists in back end databases which is also updated very dynamically as compared to the general information based static web pages (e.g. technical specifications of the Car model). So, in order to get this information the user will have to visit different used car websites, send queries through the HTML search forms, extract the relevant information from the resulting web pages. The user will have to repeat this for multiple sources and then at the end, comparison of information received from multiple sources will be required. Therefore, there is a strong need for new ways of retrieving such hidden information that can help users to find the information in an integrated manner from multiple sources automatically without burdening the user to visit multiple web sites. To minimize user effort in such an information retrieval process, the problem of automatically detecting the search interface of particular domain is explored.

There are tremendous technical challenges for designing a crawler for the hidden web:

1. One of the critical challenges is that the non indexable web allows access to user using a user friendly interfaces instead of computationally friendly interfaces i.e. it is easier to use these interfaces directly by a user rather than by an automated process to retrieve the hidden data. Hence to begin with, the crawler must be designed to automatically parse, process, and interact with form-based search interfaces that are designed primarily for human consumption.

2. Second, unlike the pre indexable web crawlers which merely submit requests for URLs, hidden Web crawlers must also provide input in the form of search queries (i.e., "fill out forms"). This raises the issue of how best to equip crawlers with the necessary input values for use in constructing search queries.
3. Another major problem is that the web forms do not have the same structure. For e.g. any user looking for books on the Web will encounter different types of forms ondifferent Websites. Hence, understanding, handling of different types of forms adds to the complication. According to recent study, there are like 307,000 deep Websites [6]and an average of 4.2 query interfaces with the help of formsfor those Web sites [6][12]. Hence, it requires search of a huge number (of the order of thousands) of forms.

To address these challenges, a task-specific, human-assisted approach can be considered for hidden web crawling. Basically we can target a particular task / domain e.g. finding out information about the used cars available in market for buying. At a very high level, the user can provide comprehensive task-specific information for searching the information and the crawler does the rest of complex work by retrieving the relevant information from multiple web sites and presenting it to user after significant level of processing to facilitate best possible information.

The form-based query interfaces are designed for user interaction, providing a programmatic way to access web databases under query interfaces and integrate search system over Web databases has become an application in high demand. Such an interface would provide uniform access to the data sources of a given domain of interest. Currently, such integrated user interfaces exist, but they are manually constructed or generated using techniques that are not fully automated. Though one might pursue a manual creation of a unified interface for a few query interfaces (four or five), it is hard to do so for a large number of interfaces without errors and in a periodical manner[7]. Aim of this paper is to construct an integrated query interface which contains all distinct fields of all schemas and arrange them in a way that facilitates users to fill in the desired information.

Due to the sparse nature of www, even if we try to find information specific to a particular domain, it is found to be very sparsely distributed e.g. a focused crawler was able to retrieve only 0.001 % of search forms out of 100,000 web pages related to movie domain. Therefore it is essential to first of all develop a strategy for visiting web pages and identifying the pages that are more likely to have a search form.

This paper has been organized in 3 main sections:

Section 2 describes a high level view of the domain specific web crawler considered here for the integrated search information retrieval technique.

Section 3 proposes a novel method to detect the search form interface by looking at the web page HTML tags, key tags on the web form, filter out the interfaces which are generic and not specific to searching the hidden databases e.g. login, registration, subscription, query, contact, postings etc. It then describes how the searchable forms are classified w.r.t. search domain.

Section 4 touches upon the conclusion and describes the future research options in the same area.

II. INFORMATION FLOW ARCHITECTURE IN DOMAIN SPECIFIC WEB CRAWLER

In this paper, high level information flow architecture for the domain specific hidden web crawler is briefly explained. The following fig. illustrates the information flow between various components of hidden web crawler.

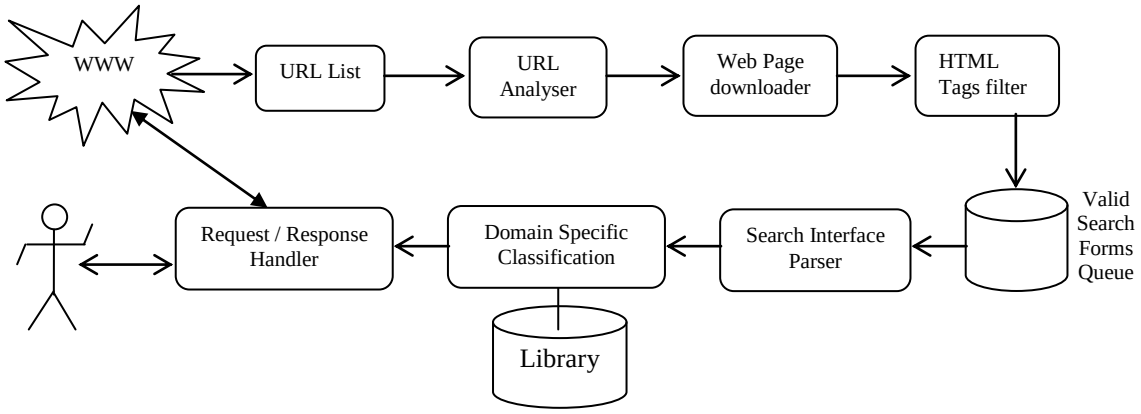


Figure 1: Flow for Automatic Detection and Integration of Web Search Query Interfaces

The flow will be as per the following details:

1. Primarily, specific to a particular domain, a common integrated base search form is presented which is prepared after extensive study of the domain specific hidden web sites.
2. User fills in the comprehensive attributes in the common form and sends the query to the hidden web crawler
3. Internally, the crawler automatically searches for the domain specific valid search form interfaces. Valid search interfaces are filtered in a separate search form queue while the generic web pages like login, registration etc. along with the complex web pages not handled by the hidden web crawler can be filtered out here.
4. Search form interfaces are further parsed to extract the domain specific attributes from the web pages
5. Domain specific mapping, matching library and the knowledge base is used to analyze the semantic relationship between the attributes of the identified search forms and the attributes from the 'common' integrated base search form filled by the user (as per the crawler). This analysis is used for generating the separate requests for individual websites with the user entered keywords in the request URLs. This mapping and matching of the keywords, attributes are critical and level of relevance of matching is given utmost consideration before generating the request URL.
6. The request URLs are prepared with the user provided keywords/inputs using a simple concatenation of attributes in the URL and submitted using the HTML tags. The request URLs are then sent to www using native libraries for the information retrieval.
7. Erroneous responses are filtered out (these can be used for further analysis and improvement of the web crawler as a feedback to the system architecture). In addition to the erroneous responses, complex responses which are not handled by the crawler are also filtered out here. It shall be noted here that the responses are also in the form of URLs.
8. The responses received from the above request quest are collated and presented to the user after one more level of evaluation – ranking etc for the relevancy most appropriate to the user.
9. The user can then retrieve the desired information by clicking on the identified response URLs.

III. PROPOSED WORK: SEARCH INTERFACE IDENTIFICATION

Significant research has been done in this area of hidden web crawling. Some more specific examples are HiWE, Deep Bot andother domain specific attempts for the same like AKSHR [11]. A number of recent studies[1][3][4] have observed that a significant fraction of Web content in fact lies outside the PIW[10].

Specifically, large portions of the Web are 'hidden' behind search forms, in searchable structured and unstructured databases. Pages in the hidden Web are dynamically generated in response to queries submitted via the search forms.

3.1 Study of Search form Interfaces

Based upon the survey done on little number of Websites, it is observed that these forms which are designed for search function have specific type of format. Normally, the submit buttons of these forms are found to be named as "GO" or "Search"[9]. Besides this, they 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.

The World Wide Web has millions of Web pages. Some of them will have forms. These forms will act as an entry point for the vast 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.

The following are some examples for different type of search form interfaces:

Figure 2: Keyword Based Search Interface Examples

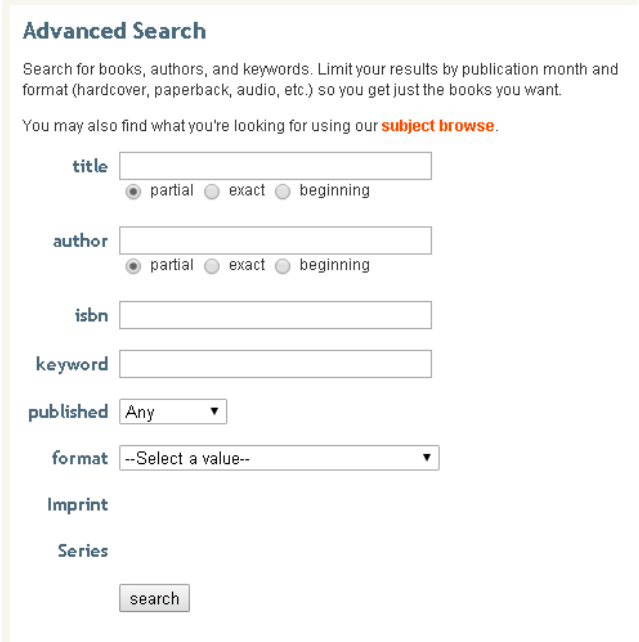


Figure 3: Multi Attribute Based Search Interface for www.penguin.com.

3.2 Search Interface Identification

A search interface generally allows the user to perform a keyword based query. The user enters the query by entering the keywords, selection out of predefined list items on the web pages designed in lot different ways. The majority of such interfaces are designed using HTML form tags (other advanced technologies like Ajax etc are not considered as part of this work and are candidates for future research work). The knowledge about HTML based page creation is used during this paper to propose the method for automatically identifying the more relevant search forms for the hidden web crawling.

This proposed method automatically detects the search interfaces by looking for the keywords in the URLs, page title, followed by key labels, attributes of the source code of the HTMP web page. The undesired web pages are discarded and desired interfaces are stored in a list/queue for further analysis.

3.2.1 IdentifyValidSearchForm()

Begin

While(URL Queue not Empty or the maximum iterations allowed limit reached**)**

Begin

1 Extract first URL from the URL queue

2 Check for the string login or registration or signup in URL

If (present) **then**

" it is a login/ registration form";

Discard URL and Go to next step.

else

Go to next step.

3 Download the source code for the web page

4 Check the source code for the web page for tag <form> and </form>

if (present) **then**

Go to next step.

else

" It is a simple web page". Goto step 1.

5 Check URL and the source code for login, registration or signup forms

if (input-type="login" or "registration" or" sign up") or

or (having password control) **then**

Go to step 1

else

Go to next step

6 Check URL and the source code for "submit", "search", "Advanced search", "submit query" or "find" – check for these in text as well as image titles as well.

if (present) **then**

Go to next step.

else

7 This is a valid web search interface, add the URL to the search interface queue;

Go to step 1.

End

8 Display the list of search interfaces from the search form queue.

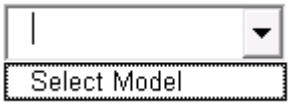
End

Algorithm 1: identifying valid search form interface

3.3 Search interface parser - attribute extraction

A form can have various kinds of input fields. Once the Web page is detected from where the form can be filled and search can be made (identified valid search form); interface parser is used for processing the page further. Once a suitable form is found from where the query can be made; that particular form is then separated out and further analyzed for the Label fields as well as Text fields (which accept user inputs). The main job of the parser is to look for various types of Labels, user enterable text fields found on that Web page. For example, different publishers or book store web sites have different (although related) fields using which the searches can be made. Each such field is combination of a user identifiable Label (field name) and Element (input text fields for accepting user enterable value).

A standard HTML Web form consists of form tags—a startform tag <form> and an end form tag </form> and thesetags contains form fields or attributes. Attributes may includeradio buttons, check boxes, selection lists, and text boxes.The <select> tag opens a selection list, and the </select> tag closes it. A sample HTML tag usage is shown here:




```
<select name=”modelone” id=”modelone”
disabled=”disabled”><option value=”0">Select Model</option></select>
```

Figure 4: Sample HTML Tag

All these type of attribute information is extracted from the search interface forms. The parses this way collects the label, attribute information from the search form. This information about the fields is then further used with the domain specific mapping, matching library and knowledge base to generate parallel requests (request quest).

3.4 Domain Classification (Matching and Mapping of Attributes)

In this phase, the semantic mappings between the components of different web interfaces of the same domain are obtained i.e. all the interfaces belong to the same domain such as book domain. For example, different publishers or book store web sites have different (although related) fields using which the searches can be made. Each such field is combination of a user identifiable Label (field name) and Element (input text fields for accepting user enterable value).

1. The main inputs to this Interface mapping system, are two interfaces A and B comprising of a number of components {n1, n2, n3… np} and {m1, m2, m3 … mq} respectively. Each of these components are then further comprising of a Label and Element attribute resulting into a combined set of i.e. Labels {L1, L2, L3 … Ln} and Elements {E1, E2, E3 … En}.
2. This component will compute semantic relationships between the fields in different interfaces in the same domain. In recent years, a considerable effort has been dedicated to finding these semantic mappings. The accuracies of these automatic methods to identify the semantic mappings between fields can be as high as 90%.
3. The library uses a repository for domain-specific search interfaces. It contains the information about each attribute e.g. data type, list of values, Yes/No type usability, attribute’s mandatory / optional characteristics

etc… to name a few. All this is fed after thorough study of a large number of the domain specific web sites. Additionally, it is essential that this is extensible as well as scalable.

4. It also provides an extensible domain-specific mapping - matcher library to support multi-strategy match approach. The multi-strategy match approach uses different matching strategies like relational matching, domain-specific thesaurus etc. The mapping – matcher library is responsible for creating the matrices of mapping (relevance percentage to begin with) or parent child relationship of the attributes using the defined knowledge base. The appropriate mapping among attributes from different interfaces is used for arriving a mapping to and from a common search interface and the individual interfaces of different web sites specific to the domain.
5. This component 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.

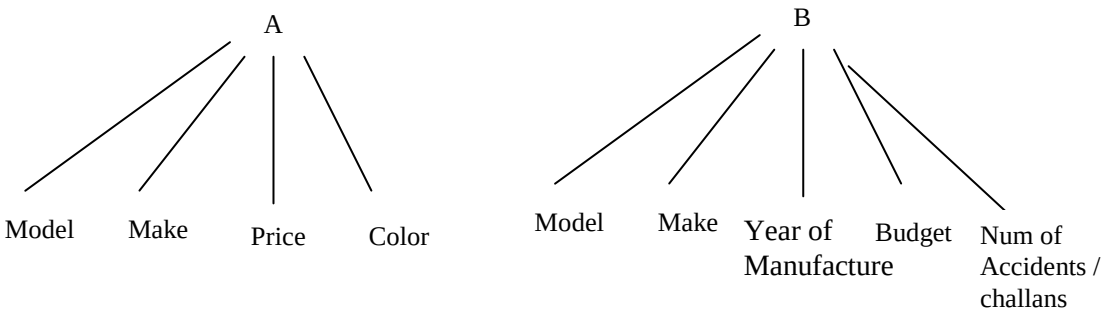


Figure 5: Hierarchical Representation of Two Query Interfaces in USED CAR Domain

In the fig. above the two interfaces of the Used Car domain are shown. The query interfaces for two websites take different attributes however some of these are common or highly related to each other. Some interfaces have additional attributes than the other interfaces to provide some extra information to the user.

These attributes are matched and mapped together to arrive at a common integrated search interface for the user (base interface). Additionally similar mechanism is followed during information retrieval, where the user provided inputs / keywords to the common search form are mapped to the individual site specific interface, the input values are mapped to the corresponding attributes in the individual site's query URL before the query is sent for processing.

IV. CONCLUSION AND FUTURE SCOPE

The approach described in this paper automatically detects the domain specific search interfaces by looking the domain word in the URLs, then the web page title, and after that attributes of the source code of the web page including the title of images in the web page. The task of presenting a comprehensive, integrated and common search interface is achieved by using the semantic mapping, matching of the attributes specific to a particular domain with the help of domain specific mapping, matching library and knowledge base. The input to this library and knowledge base is the information collected after careful study of the different most used websites specific to that particular domain. This common integrated base interface will allow the users to access information uniformly from multiple sources for that domain. The parallel request quest will be automatically triggered by the crawler on behalf of the user by generating the request URLs after carefully concatenating the input keyword values entered by the user.

The approach described here is on the basis of study of few specific domains however this can be further extended to other domains as well. The domain specific interface matching and mapping knowledgebase is one area which can be enhanced by using better and improved mechanisms using logic as fuzzy matching after more profound study of more and more web sites specific to the domain.

REFERENCES

[1] Steve Lawrence and C. Lee Giles, Accessibility of information on the web, http://nike.psu.edu/classes/ist597/2003-fall/papers/web-access.pdf, 1999

[2] Kobayashi, M. AND Takeda, K. Information Retrieval of the Web, ACM Computing Surveys, 2000

[3] S. Lawrence and C. L. Giles. Searching the World Wide Web. Science, 280(5360):98–100, 1998.

[4] Bergman, M. The Deep Web: Surfacing Hidden Value, The Journal of Electronic Publishing, 2001

[5] Resource link: http://www.newworldencyclopedia.org/entry/Deep_Web.

[6] Barbosa, L AND Freire, J. Searching for Hidden-Web Databases, Eight International Workshop on Web and Databases, 2005

[7] Dragut, E; UIC; Wensheng Wu; Sistla, P. ; Yu, C. ; WeiyiMeng, Merging Source Query Interfaces on Web Databases, Data Engineering, 2006. ICDE '06. Proceedings of the 22nd International Conference, 2006.

[8] Barbosa, L AND Freire, J. Siphoning Hidden-Web Data through Keyword-Based Interfaces. In Proc. of SBBD, 2004

[9] Hypertext markup language (HTML). http://www.w3.org/MarkUp.

[10] Raghavan, S AND Garcia-Molina, H. Crawling the Hidden Web. In Proc. Of VLDB, 2001

[11] Sharma, A.K. And Bhatia, K.K. DSIM-Domain Specific Interface Mapper

[12] Chang, K., He, B., Li, C., Patel, M., AND Zhang, Z. Structured Databases on the Web: Observations and Implications. SIGMOD Record, 2004

[13] Nupur Gupta, Shalini Kapoor. Extraction of Query interfaces for domain specific hidden web crawler, IJCSIT, 2014.

[14] Usha Gupta. Fetching the hidden information of web through specific domains, IOSR 2014.

[15] Resource Link: http://www.thinkpink.com/bp/WebCrawler/History.html.

[16] Pinkerton, B. Finding WhatPeople Want: Experiences with the WebCrawler, Second International WWW Conference, 1994