A New Hidden Web Crawling Approach

L.Saoudi
Computer science department
Mohammed Boudiaf University
M'sila, Algeria

A.Boukerram
Computer science department
Abderrahmane Mira University
bejaia, algeria

S.Mhamedi
Computer science department
Mohammed Boudiaf University
M'sila, Algeria

***Abstract*—Traditional search engines deal with the Surface Web which is a set of Web pages directly accessible through hyperlinks and ignores a large part of the Web called hidden Web which is a great amount of valuable information of online database which is "hidden" behind the query forms.**

To access to those information the crawler have to fill the forms with a valid data, for this reason we propose a new approach which use SQLI technique in order to find the most promising keywords of a specific domain for automatic form submission.

The effectiveness of proposed framework has been evaluated through experiments using real web sites and encouraging preliminary results were obtained

Keywords—Deep crawler; Hidden Web crawler; SQLI query; form submission; searchable forms

I. INTRODUCTION

The World Wide Web is a global information medium of interlinked hypertext documents accessed via computers connected to the internet. Most of the users rely on traditional search engines to search the information on the Web. These search engines deal with the Surface Web which is a set of Web pages directly accessible through hyperlinks and ignores a large part of the Web called hidden Web which is hidden to present-day search engines. It lies behind search forms and this part of the Web containing an almost endless amount of sources providing high quality information stored in specialized databases, only accessible through specific search interfaces created by using CGI and HTML forms or JavaScript etc [1]. which need to be filled manually by the user. A search interface consists of different form elements like text boxes, labels, buttons etc. User must provide an entry in at least one of them, and submit the form to obtain response pages containing the results of the query.

The hidden web crawler must also perform a similar filling process either by selecting suitable values from the domain of each finite form element or by automatically generating queries. The challenge is how to equip crawlers with the necessary input values for use in constructing search queries to obtain the optimized response pages without errors?

To address these challenges, we adopt a task-specific based SQLI approach to crawl the hidden Web.

The rest of the paper has been organized as follows: Section II describes different concepts related to hidden web crawler; section III describes the proposed work i.e. design of a Domain-specific hidden web crawler and explains the functionality of different components of crawler; section IV presents the progress of the experiment and its phases, section V describes the experimental results that is done over book domain which are discussed in section VI and finally, section VII draws the conclusion and describes the future research.

II. HIDDEN WEB CRAWLERS

A. Generality

A web crawler (also known as a robot or a spider) is a system for the bulk downloading of web pages. Web crawlers are used for a variety of purposes. Most prominently, they are one of the main components of web search engines, systems that assemble a corpus of web pages, index them, and allow users to issue queries against the index and find the web pages that match the queries. [2]

The whole web is divided into two types: the public web and the hidden web. The public web normally deploys by common use search engine, hidden web represents information, stored in specialized databases, only accessible through specific search interfaces created by using CGI and HTML forms or JavaScript etc. [3]

However, a number of recent studies [4,5,6,7] have observed that a significant fraction of Web content in fact lies outside the PIW(Publicly Indexable Web). Specifically, large portions of the Web are 'hidden' behind search forms, in searchable structured and unstructured databases (called the hidden Web or deep Web [6]). Pages in the hidden Web are dynamically generated in response to queries submitted via the search forms.

B. General Hidden web crawling strategy

The basic actions of a deep web crawler are similar to those of other traditional crawlers. A traditional web crawler selects URL's, retrieve pages, process the pages and extract links from the retrieved pages. The traditional crawlers do not distinguish between pages with and without forms[8]. Whereas, a Hidden web crawler performs additional sequence of actions for each form on a page [9]:

1) Form detection: the search form extractor looks for any <FORM> tags in the HTML web page to extract the associated search forms.

2) Form Analysis: Parse and process the form to build an internal form representation.

3) Value assignment: Use approximate string matching between the form labels and the labels in the database to generate a set of candidate value

4) Form Submission: send the filled form to the web server.

5) Response Analysis and Navigation: Analyze the response to check if the submission yielded valid search results. Crawling the hypertext links in the response page to some prespecified depth.

III. Our Proposed Approach

Our crawler is a domain specific Hidden Web crawler, Fig.1 depicts the proposed architecture of our crawler HiWC, having : a Web Page Analyzer, a Form structure and content classifier, the Form filler that uses a Domain Specific data repository and a response page analyzer.

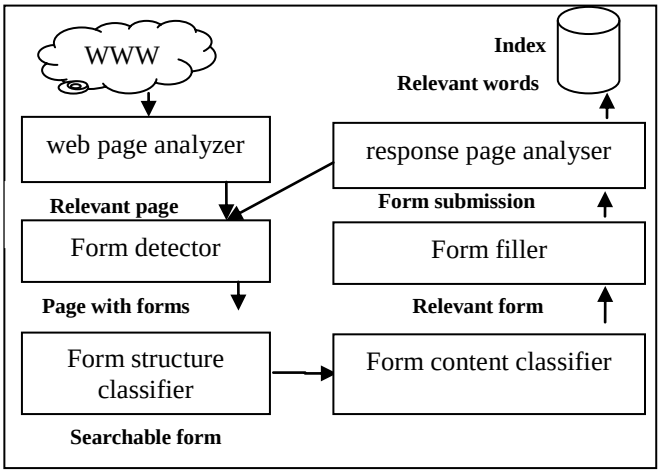

Fig. 1. general crawling process

Our crawling approach is divided in two phases:

A. Domain Definition phase:

In this phase (Fig2), we implement the domain definitions used to define a data-collection task. A domain definition is composed of two tables.

1) Labels table:

This table is an object relational table; it has three attributes label, alias and score. Each label *ai* has an associated name, a set of aliases {ai _alias1,…, ai _aliask}, and a specified score *si*.

A label represents a field that may appear in the search forms that are relevant to the domain.

The alias represents alternative labels that may identify the attribute in a query form. For instance, the attribute AUTHOR, from a domain used for collecting data about books, could have aliases such as "writer" or "written by".

The score is a number between 0 and 1 which represent the weight of each label in the domain, for example the label ISBN should have the higher score in the book domain.

We collect search labels from training web sites, and then we find manually the alias of each label and give its score from its frequency in the search form in all the training sites.

2) Repository table:

This table has *m* attributes, each one take a value of the attribute label in the labels table (for example in the book domain, we take: ISBN, author, title...etc as attributes) we add an attribute which represent the weight of the object in its domain.

We obtain the initial repository values from 10 training search web sites.

we observed in several cases that the majority of search web page are vulnerable of SQLI queries, for this reason our crawler sends different SQLI queries to each search site to extract its response pages, for each obtained response page the crawler extract object values ,store its relevant information into database and give their weights , the weight is a value from 1 to 10, if the same object exist in the 10 web sites then its weight=10 which signify that this object is very important and should be among the first selected key words in the form submission process.

B. Crawling phase:

This phase passes through several steps as it shown in fig3:

1) Form detection

A standard HTML Web form consists of form tags, a start tag <form> and an end tag </form> within which the form fields reside. Form detector looks for any <FORM> tags in the HTML web page to extract the associated form.

2) Form structure classification

It was observed in several forms that there are content differences between searchable and non-searchable forms, For instance, forms with a password field, username, email or with a file upload field are non-searchable. The goal of the classification performed automatically was to exclude non-searchable forms.

3) Form content classification

To classify the search page into relevant or not relevant page, we extract form labels and use them in the classification process. The method we use to determine if a form is relevant to a domain consists of adding the frequency of each label, pondered by its predefined score, and checking if the sum exceeds the relevance threshold μ.

4) Form Submission

Once the system determines that a form is relevant to a certain domain d, the crawler select the promising objects with height weights to fill the form according to the matching process between the form labels and its equivalent on the labels table to provide a successful form submission.

5) Response page analysis

The response page to a form submission is received by a response analyzer module to distinguish between pages containing search results and pages containing error messages, different relevant values are extracted from successful response page. This feedback can be used to update the repository table. The obtained values are assigned scores that vary with time. The score of an object gets a positive (negative) boost ever time it is used in a successful (unsuccessful) form submission.

6) Indexing dynamic page

Indexes are data structures permitting rapid identification of which crawled pages contain like particular words or phrases, it has two types:

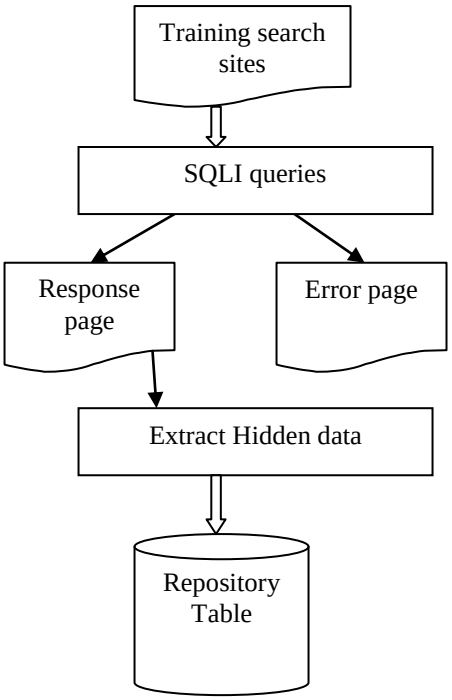

Fig. 2. Initial phase

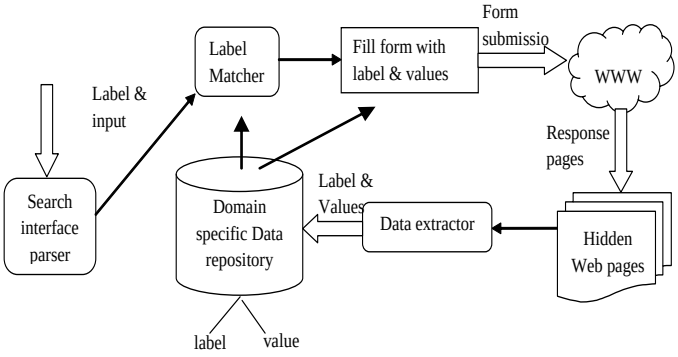

Fig. 3. crawling phase

- Inverted index: the inverted index stores a list of the URLs containing each word
- Forward index: The forward index stores a list of words for each URL

The rationale behind developing a forward index is that as URL are parsing, it is better to immediately store the words per URL. The forward index is sorted to transform it to an inverted index.

IV. EXPERIMENTATION

To evaluate the performance of our approach, we test it on Books domain, our experimentation passes through two phases.

A. Training phase:

In this phase our crawler tries to collect initial data to implement the repository and labels tables.

The process for creating the domain definition was the following:

For book domain, we manually explored 10 sites at random as it shown in table1, and used them to define the attributes and its aliases. The specificity weight and the relevance threshold were also manually chosen from our experience visiting these sites.

TABLE I. TRAINING SITES

Site name	URL	SQLIA
Book Depository	https://www.bookdepository.com/	query :)or'(1)=(1
ISBN Search	http://www.isbnsearch.org/	'or'1'='1
Paperback Swap	http://www.paperbackswap.com/	'or'1'='1
Tattered Cover Bookstore	http://www.tatteredcover.com/	'or'1'='1
AbeBooks	http://www.abebooks.com/	'or'1'='1
Free-eBooks.net	http://www.free-ebooks.net/	'or'1'='1
Green Apple Books&Music	http://www.greenapplebooks.com/	'or'1'='1
goodreads	https://www.goodreads.com/	'or'1'='1
Alibris	http://www.alibris.com/	'or'1'='1
Thomson Gale	http://www.cengage.com/	'or'1'='1

After having injected these sites, these latter will be analyzed for extracted all data of each book and put them in the table of labels that contains the label, aliases, and score as it shown in table2.

TABLE II. LABELS TABLE

label	alias	score
Title	'Name', 'title of book'	0.6
Author	'By', 'Written by', 'Author', 'author's name'	0.7
ISBN	ISBN-13, ISBN-10	0.95
Publisher	Editor	0.8
Format	'binding type'	0.25
Category	Genre, sort	0.05
Price		0.05

B. Testing phase:

To evaluate the performance of our approach, we test it on Books domain. Once the domain was created, we use our crawler to crawl 12 websites called test websites. The list of websites visited by HiWC (Hideen Web Crawler) is shown in Table 3.

To check the accuracy of the obtained results, we manually analyzed the websites and compared the results with those obtained by our crawler. We measured the results at each stage of the process: associating texts with form fields, associating form fields with domain attributes, establishing the relevance of a form to a domain, and executing the queries on the

relevant forms. To quantify the results, we used standard Information Retrieval metrics: precision, recall. The metrics defined to measure the performance of HiWC, make use of the following variables:

- FieldAttributeAHiWC: set of the associations between form fields and domain attributes discovered by HiWC.

- FieldAttributeAManual: set of the associations between form fields and domain attributes discovered by the manual analysis.

- FormDomainAHiWCt: set of the associations between forms and domains discovered by HiWC.

- FormDomainAManual: set of the associations between forms and domains discovered by manual analysis.

- SubmittedFormsHiWC: set of forms successfully submitted by HiWC.

TABLE III. TESTING SITES

Name	URLs
Blackwell's Bookshop	http://bookshop.blackwell.co.uk
The American Book Center	http://www.abc.nl
Strand Book Store	http://www.strandbooks.com
Dymocks Booksellers	https://www.dymocks.com.au
eCampus.com	http://www.ecampus.com
Powell's Books	http://www.powells.com
Barnes&Noble	http://www.barnesandnoble.com
Bookjetty	http://www.bookjetty.com
Listal	http://www.listal.com
Library thing	https://www.librarything.com/
BookFinder.com	http://www.bookfinder.com/
IWC Schaffhausen	http://www.iwc.com/en/collection/portugieser/

We defined the following metrics:

Metrics for associating labels and form fields.

PrecisionFieldAttributeA:= | FieldAttributeAHiWC ∩ FieldAttributeM | / | FieldAttributeAHiWC |

RecallFieldAttributeA = | FieldAttributeAHiWC ∩ FieldAttributeM | / | FieldAttributeM |

Metrics for Global associations between forms and domains:

Precision FormDomainA:= | FormDomainA HiWC ∩ FormDomainAM | / | FormDomainAHiWC |

RecallFormDomainA = | FormDomainAIHiWC ∩ FormDomainM | / | FormDomainM |

Precision SubmittedForms = |SubmittedForms HiWC | /| FormDomainA HiWC ∩ FormDomainAM|

V. EXPERIMENTAL RESULTS

In this section, we summarize some of the more significant results from these experiments.

We now take the 12 testing sites to crawl them and extract the forms, this latest be classified into two categories those which is searchable form and non-searchable form, as, it shown in table4.

TABLE IV. EXTRACTED FORMS CLASSIFICATION

Number of sites from which forms were picked	12
Total number of forms	25
Numbers of search forms for books	12

TABLE V. EXPERIMENTAL RESULTS

	D1(10)	D2(12)	D1+D2(22)
Submitted Forms			
Precision	13/13 1.00	12/12 1.00	25/25 1.00
Form-Domain Associations			
Precision	13/13 1.00	12/12 1.00	25/25 1.00
Recall	13/13 1.00	11/11 1.00	24/24 1.00
Field-Attribute Associations			
Precision	27/28 0.96	45/50 0.95	72/78 0.92
Recall	27/28 0.96	45/52 0.87	72/80 0.9

We performed a number of experiments to study and validate the overall architecture as well as the various techniques that we have employed.

Table V summarizes the obtained experimental results.

For each book domain, it shows the values obtained for all the metrics in the Training dataset (D1, the sites used to define the domains), the test dataset (D2, the testing sites) and in the Global dataset (D1+D2, Training+ testing).

It is important to notice that, in order to calculate the metrics for form-domain and field-attribute associations, "quick search" forms have not been considered.

VI. DISCUSSION

The obtained results are quite promising: all the metrics show high values and some of them even reach 1.00

Recall in associating forms and domains reached 1.00 in every case

The precision values obtained for the associations between attributes and form fields exceeded 0.92 with a recall 0.9.

The majority of the errors in this dataset came from a single source (Blackwell's Bookshop). If we did not have into account this source, the metrics would take values similar to those reached by the other ones.

VII. CONCLUSION

In this paper we described the conceptual and experimental study of the proposed approach. Our approach is based on a domain definition, which describe a data-collecting task based SQL injection technique to extract the most promising keywords of a specific domain for automatic form submission. We presented a simple operational model of a hidden Web crawler that succinctly describes the steps that a crawler must take: relevant page extraction, form detection, form structure classification, form content classification, form submission and response page analysis.

We described the architecture and design techniques used in HiWC, a prototype crawler implementation based on SQLI to get the initial keywords values and fill the repository database in the training phase. The promising experimental results using HiWC demonstrate the feasibility of hidden Web crawling and the effectiveness of our different techniques to implement this crawler.

In the future, we propose to handle forms powered by Javascript , that can significantly improve HiWC performance, and to test our crawler with different task specific domains.

REFERENCES

[1] L. Barbosa, J. Freire. "An Adaptive Crawler for Locating HiddenWeb Entry Points", IW3C2 2007, Banff, Alberta, Canada, May 8–12, 2007.

[2] C. Olston, M. Najork, Web Crawling, now the essence of knowledge, vol. 4, No. 3 ,2010 ,pp. 175–246.

[3] K. K. Bhatia, A.K. Sharma, and R. Madaan, "AKSHR: A Novel Framework for a Domain-specific Hidden Web Crawler," Proceedings of the 1st International Conference on Parallel, Distributed and Grid Computing , 2010, pp. 307–312.

[4] J. Madhavan, D. Ko L. Kot, "Google's deep web crawl," Proceedings of the 34th Internation Conference on Very Large Data Bases (VLDB), August, Auckland, New Zealand, (2008),pp. 1241-1252.

[5] M. Soulemane, M. Rafiuzzaman ,H. Mahmud, "Crawling the Hidden Web: An Approach to Dynamic Web Indexing" International Journal of Computer Applications.,vol. 55, No. 1,pp. 7-15, October 2012.

[6] G. Z. Kantorski, V. P. Moreira, and C. A. Heuser, "Automatic Filling of Hidden Web Forms: A Survey," SIGMOD Record,Vol. 44, No. 1, , March 2015, pp. 44-35.

[7] B. Saharawat, A. Goyal, "Prefetching Data from Hidden Web with DSIM Architechture", International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, No. 4, April 2013, pp. 887-890.

[8] S.Bal Gupta, "Challenges in designing a hidden web crawler", International journal of information technilogy and system,vol.2,No.1, jan-jun 2013, pp.2277-9825

[9] S. Raghavan, H.Garcia-Molina, "Crawling the Hidden Web", Technical Report 2000-36, Computer Science Department, Stanford University, December2000.