ISSN:0975-9646
Juhi Sharma et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (6) , 2014, 8239-8242

An Improved Extraction Algorithm from Domain Specific Hidden Web

Juhi Sharma
MTech Research Scholar ,
Department of CSE
MIET Meerut

Mukesh Rawat
Assistant Professor,
Department of CSE
MIET, Meerut

Abstract- The web contains a large amount of information which is increasing by magnitude every day. The World Wide Web consists of Surface Web (Publicly Indexed Web) and the Deep Web which consists of Hidden Data, also- referred to by different names such as Hidden Web, Deepnet or the Invisible Web. A user can directly access the surface web through a Search Engine but to access the hidden data/information, the users have to manually feed a set of keywords in a typical search interface to access these hidden web pages from source web sites. The problem area we are working on is devising efficient mechanisms to extract this information automatically beforehand since "crawlers" cannot access it otherwise. In this paper we present a mechanism to extract search forms from HTML pages spread over the web, automatic filling and submission of those forms at their source sites to download the Hidden Web pages in a repository for further use by web crawlers.

General Terms-Query Interface for data extraction from Hidden Web.

Keywords-Hidden Web, Query Interface, Data Mining

1. INTRODUCTION

It has been widely researched that only a frugal percentage of web content can be reached by web crawlers from the following hyperlinks [7,12]. As per many researchers, the size of hidden web has increased exponentially with time since organizations have put a lot of valuable information which can be accessed through search forms [12].It has been estimated that the hidden web contains information to the order of petabytes. This information is of very high quality. It also contains important information for the users. The World Wide Web consists of Surface Web (Publicly Indexed Web pages) as well as Deep Web (Hidden data). Over the years the number of Publicly Indexed Web pages has grown to billions and also 'deepened' i.e. the Hidden Data has increased. The Hidden Data is accessed by filling Query Forms of the web-data sources. These Query Forms consist of attributes that need to be filled by the user in order to gain access. For example, if we need to access research papers from a university website the Query Form consists of attributes like Author, Year of Publication, Journal name etc. So the problem is that filling these fields manually is a tedious process and therefore we want the Hidden Data to be available without the need to fill these Query Forms.

1.1 RELATED WORK

In the book published by Chris Sherman and Gary Price [2], "the paradox of invisible web"is discussed at length. They have discussed the reasons why crawlers can not see the hidden information but the methodology of uncovering this has been omitted.

Similarly Sriram Raghavan and Hector Garcia-Molina [3] have also discussed the limitations of current day crawlers. In their paper a model called Hidden Web Exposer (HiWE) is described and a technique called Layout-based Information Extraction Technique (LITE) is also discussed. The researchers have also discussed design challanges for a hidden web crawler. Their research proposed a "task specific, human assisted approach" to extract data from the hidden web. The key challenges in discovering hidden web data is the mechanism involved to automate it in processing form based interfaces, provide input to fill out forms and the issue of how best to equip crawlers with the necessary input values for use in constructing search queries.

1.2 PROPOSED WORK

As of now a user needs to fill the Query form in order to access hidden data from web-data sources manually which is a time consuming process. Our aim is to automate this process, so that the user can get this hidden data without the need to fill each field of the Query form manually. Automation of this process will definitely speed up the search process which is desired. In this method we will help user access the hidden data directly from the database containing the hidden data fetched from web-data sources previously which will reduce the search time considerably. Whenever a user enters a search query to access the hidden data the search engines will directly go to the Hidden Database created by us and fetch the matching results instead of going to the query form page of web-data sources.

The following section discusses the functional components of the modules we have implemented.

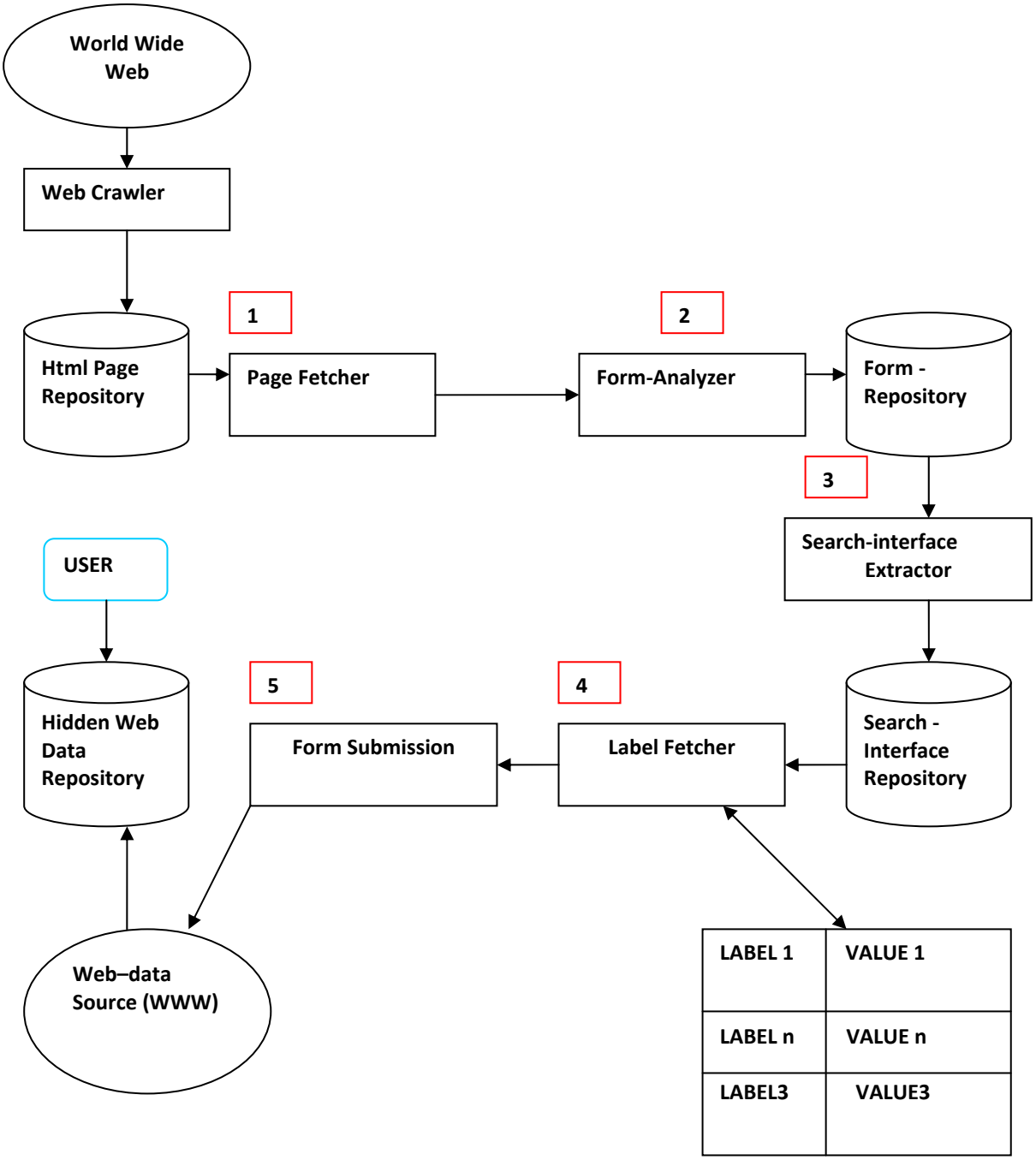

Fig 1: Architecture of Domain Specific Hidden Data Web Extractor

2. General Architecture Of Domain Specific Hidden Web Data Extractor

The general architecture of our proposed system consists of various components as shown in the figure above which help us to accomplish the task of Automatic Extraction of Domain Specific Hidden Web Data. The Search Engine uses our proposed system to directly return hidden web-data to the user without the need to fill Search Interface forms which prior to this is a very time consuming process.

The main components of our architecture are the modules. There are five modules in our architecture

1. Page Fetcher – The function of this component is to fetch HTML pages from the HTML page repository one by one.

2. Form Analyzer- The function of this component is to extract the form from the HTML page fetched by the Document Fetcher.

3. Search Interface Extractor- The function of this module is to identify whether the form extracted by the

Form Analyzer is a Search Interface or an Input Form. It discards the Input type forms and saves the search forms.

4. Label Fetcher- The job of this module is to fill the Search Form extracted by the previous module. The labels and their values are fetched from a table known as the Label Value Set and filled in the form.

5. Form Submission- This module goes to the website of the Web-data source mentioned in the Form coding submits this form and stores that data into a repository from where the search engine can directly access it

The modules in our system architecture reference other components like the databases and tables. There are four databases and one table in our system. The databases are used to store :

A. HTML Page Repository- As the name suggests this database stores the HTML pages crawled by the Web crawler component of the search engine.

B. Form Repository- This database stores the various forms extracted out of the HTML pages.

C. Search Interface Repository- The Search Forms are stored in this database.

D. Hidden Web Data Repository- The final output of our proposed system is stored in this database. It stores the hidden information fetched from the Web-data Source by submitting Search forms.

E. Label Value Set- This is a table that stores the set of label and corresponding values. This table helps in filling the Search Forms.

The system architecture works as follows; first we have a database of HTML Web pages in our HTML Page Repository then each HTML page is fed by the Document Fetcher into the Form Analyzer here the form is extracted from the HTML page and the rest of the code is rejected and the Form is stored in the Form Repository. Next the Search Interface Extractor module analyses the forms present in the Form Repository and select only Search forms and reject all the forms of other types. The Search Forms selected are stored in the Search Interface Repository. The Label Fetcher module picks the search form and finds the values of each field present in them and fills it. The Form Submission module submits the Search Form at the Web-data Source and fetches the hidden information and stores it in the Hidden Web-Data Repository. The user can now directly access this information from this repository using any search software.

3. DESCRIPTION OF THE MODULES

3.1 Page Fetcher

The first module in the system architecture is the Document Fetcher. The functionality of this module is to fetch the HTML documents present in the HTML Page Repository one at a time. The input to this module is a HTML document and this document is fed into the next module in our architecture which is the Form Analyzer. The algorithm of this module is implemented using the function **extract (URL of the HTML Page Repository).** The algorithm of this module is as follows in the Fig.2.

```
Extract (s1)
{
If (s1 is directory)
S1<<s1+"/"
St[] << files and directories in s1
For i =1 to length [st]
s3<<s1+st[i]
Extract (s3)
Else
Ex_form (s1)
}
```

Figure 2: Algorithm For Page Fetcher

3.2 Form Analyzer

The Form Analyzer module is the second module in our system architecture. The input to this module is an HTML page from the previous module and output of this module is the Form if present in the HTML page. The function of Form Analyzer module is that it looks for the Form tag inside that HTML page and if it finds a form tag then it copies the information within the form tag into a file with the name same as the name of the fetched file and store it into the Form Repository and discards the rest of the code. If there is no Form tag present in the HTML page fetched then it discards the complete page and moves on to the next page. The algorithm for the Form Analyzer is implemented by the function **ex_form (URL of the HTML file in the repository).** The algorithm used by the Form Analyzer is as shown in the Fig. 3.

```
ex_form (sr)
{
s_chk<<characters in file represented by sr
If ("<form" found in s_chk)
Write characters between "<form>" and
"</form>" in a file
}
```

Fig 3: Algorithm for Form Analyzer

3.3 Search Interface Extractor

The third module is the Search Interface Extractor. The input to this module is the Form extracted from the HTML page by the Form Extractor in the previous step. The function of this module is to identify whether the Form is a Query Interface or Input Form. In this module we look for the label of the button at the end of the form. If the label of the button is one of these 'Search', 'Go', 'Find' then the Form is definitely a Query Form and it is stored into the Search Interface Repository else the Form is of Input Type and it is discarded. The algorithm used by this module is as follows in Fig.4 .

3.4 Label Fetcher

The Label Fetcher module is the fourth module in our system architecture. Search interfaces from the Search interface Repository is input into this module and the output of this module is the filled Search Interface. In this

module each field present in the Search Form is considered one by one and than matched using a table known as the Label-Value Set. Each fields label is matched with the list in the table and if a match is present than the corresponding value in the table is filled in the Form against the field. This is how the complete form is filled using the Label-Value Set table.

```
ex_form (String sr)
{
Var f1, f2
Var fin, fin1
Var ch, ch1
Var fou
F1 <<file represented by string "sr"
Fin <<connect inputstream with f1
While (ch ? read from fin! =EOF)
{
S1 ←ch
}
If (s1 contains any string such as "go,
submit, search, find book etc")
{

fin1 <<connect input stream with f1
f2 <<new file created in "d:/hdoc/searchpage/"+f1
fou <<connect output stream with f2

While (ch1 ←read from fin1! = EOF)
    {
     write ch1 in f2
    }

}}
```

Fig 4: Algorithm for Search Interface Extractor

3.5 Form Submission

This is the last module in our System Architecture. The input to this module is the filled Search Interface and the output of this module is the Hidden Web-data which is stored in a repository from where the user can directly access it through the search engine. In this module the Form is submitted at the Web-data Source on the World Wide Web and the hidden information is fetched from the servers and stored in the Hidden Web-data Repository.

4. CONCLUSION

Over the years, the surface web (Publicly Indexed Web) has grown to billions of HTML pages. Simultaneously, the web has been rapidly “deepened” by the massive web data-sources and a far more significant amount of information is hidden in the deep web, behind query forms of web data-sources like amazon.com. These data-sources are also referred as hidden or invisible web. These hidden sources allow the users to access the underlying information by querying through their query interfaces. Where in data-sources contain the attributes that tend to describe the information accessible through them. For example, the query interface of a source like amazon.com contains attributes such as author, title, ISBN etc. These query interfaces acts as the entry points to the hidden or invisible web and therefore became potential candidates to be identified to extract hidden data.Our work presents a mechanism to extract pages, forms, identifying search forms and extract out their labels which could be further used by the hidden web crawler. The hidden web crawler automatically fills these forms with the values available in the label-value-set table in the database and send these forms to the World Wide Web and extract out the deep information and analyze it. We propose to extend our work to support regional languages and further to optimally design mechanisms to index as well as extract such documents.

REFERENCE

[1] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. In ICDE, 2003.

[2] The Invisible Web: Uncovering Information Sources Search Engines Can't See by Chris Sherman and Gary Price (Cyber Age Books, 0-910965-51-X).

[3] Crawling the Hidden Web, Sriram Raghavan and Hector Garcia-Molina, Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

[4] E. Agichtein, P. Ipeirotis, and L. Gravano. Modeling query-based access to text databases.In WebDB, 2003.

[5] Article on New York Times. Old Search Engine, the Library, Tries to Fit Into a Google World. Available at: http://www.nytimes.com/2004/06/21/technology/21LIBR.html, June 2004.

[6] The Open Directory Project, http://www.dmoz.org.

[7] M. K. Bergman. The deep web: Surfacing hidden value,http://www.press.umich.edu/jep/07-01/bergman.html.

[8] S. Byers, J. Freire, and C. T. Silva. Efficient acquisition of web data through restricted query interfaces. In WWW Posters, 2001.

[9] J. P. Callan and M. E. Connell. Query-based sampling of text databases. ACM Transactions on Information Systems, 19(2):97–130, 2001.

[10] B. He, M. Patel, Z. Zhang, and K. C.-C. Chang. Accessing the Deep Web: A survey. Communications of the ACM, 50(5):95–101, 2007

[11] J. Madhavan, S. Jeffery, S. Cohen, X. Dong, D. Ko, C. Yu, and A. Halevy. Web-scale Data Integration: You can only afford to Pay As You Go. In CIDR, 2007..

[12] K. C.-C. Chang, B. He, C. Li, and Z. Zhang. Structured databases on the web: Observations and implications. Technical report, UIUC.