ISSN:0975-9646

Understanding the Technique of Data Extraction from Deep Web

Manoj D. Swami#, Gopal Sonune#, Dr. B.B. Meshram*

#*M.Tech (NIMS)*
VJTI, Matunga, Mumbai.

**HOD, Dept. of Computer Technology*
VJTI, Matunga, Mumbai.

Abstract— **World Wide Web is developing rapidly, there are large number of Web databases available for users to access. This fast development of the World Wide Web has changed the way in which information is managed and accessed. So the Web can be divided into the Surface Web and the Deep Web. Surface Web refers to the Web pages that are static and linked to other pages, while Deep Web refers to the Web pages created dynamically as the result of specific search. This literature paper focuses on querying the Deep Web.**
Deep Web refers to the databases accessible through query interfaces on the World Wide Web. A Deep Web query system presents to users a single interface for querying multiple Web databases in a domain such as airline booking and extracts the relevant information from different web databases sources, and then returns results for users.

Keywords— **Deep Web, Surface Web, Deep web tool query, HiWe system.**

I. Introduction

The Deep Web refers to the accessibility of different web databases through query interfaces on the World Wide Web. A Deep Web query tool presents a single interface to users, and upon submission of a query via its interface, the tool submits equivalent queries to many hidden databases via front-end query interfaces and then extracts and merges the results received from different web-sources. The advantage of this tool in the airline domain for example, is to prevent the users querying from each airline among many airline websites which is time consuming; the second advantage is that the tool will present a simple and easy query interface to users and collect data from hidden airlines databases and then return a single interface of results for user-processing.

A Deep Web Tool usually has three components, Interface interpretation, Query formulation and Result interpretation.

Interface interpretation: this component produces an integrated interface over the query interfaces of web databases and analyzes the different web pages, concentrating on identifying the sections of the web pages that contain the relevant form (e.g. booking services, Payment services). Once the relevant sections are identified, relevant page attributes (or HTML tags) need to be identified. Different page attributes then need to be semantically mapped. A database or file template can be created to store all those page attributes.

Query formulation part will involve schema integration, formulating the query to be sent to the various web resources. The query formulation part can be developed separately from the result interpretation part.

Result interpretation extracts the results from pages returned by different web databases then merges them together into global interface for the utilization by users. This part requires the appropriate methods for data extractions and merging. The Deep Web domain is vast; this paper concentrates on result interpretation.

II. Overview

The remainder of this literature paper is organized as follows. Section 3 discusses the related work of response page processing including data error and duplication management. In section 4 we discuss the result processing approach in order to extract the relevant pieces of information out of returned pages. Section 5 presents methods use for data extraction in the Deep Web. Section 6 discusses data integration into a unified interface. Finally, section 7 concludes the paper and gives an overall summary.

III. Discussion

3.1. Response Page Processing from the Deep Web

Once a query is sent to relevant websites, the next step is to retrieve information from those target sites. Several cases are possible.

3.1.1 Results display by pieces

Handling of results displayed by piecemeal is discussed in [1]. In this case, the web site returns a bit at a time, showing perhaps 2 or 4 per page. The system will provide a button or a link URL to get to next page until the last page is reached. One approach [1] treats all the consecutive next pages from the returned page as part of one single document by concatenating all the pages into one page. The system activates this process if the returned page contains a button or link indicating next or more. In this way, the system constructs a logical page containing all the data.

3.1.2 Retrieving all results with default query in the case of small database

In the default query, the system may have retrieved all or least of significant percentage of the data before submitting all queries; the reason behind is, many forms have a default query that contains all data available from the website. Stephen W. Liddle, David W. Embley, Del T. Scott and Sai Ho Yau [1] discussed this issue in depth. The problem found in a default query (with a default query the user is not necessarily selecting or filling fields with information) is that, sometimes it does not retrieving all data and every set of data returned may be some particular subset of the overall database, in this case the problem is solved by sampling the database and finding the data not already returned by the initial default query, the user will continue the process of submitting the query until as much data as possible is retrieved. If the additional queries all return data that is equal to or subsumed by the data returned for the initial default query, we need not query with all combinations.

3.1.3 Query submitted with a field missing or No-Result Found

For a query submitted with a field missing, or for no-result found, the system must automatically detect the problem and solve it. In the case that a required field is missing, the system will search for a message such as "Required field is missing" this kind of error requires the intervention of the user using the system. The user will be required to fill the relevant fields of the interface and submit again the query to the system. In the case that no result can be displayed to the user query, the system could search for message like "No matching result could be found". Both error cases have been discussed in [1]. It is more reliable to observe that the size of the information returned after removing miscellaneous header and footer information is normally very small if there was an error-usually a constant small value for all queries that return no result [1].

3.1.4 Errors Handling

During the response page processing from the Deep Web, the following errors may be encountered:

• In the case of network failure, a server down, or HTTP errors, the system will notify the user by an error message and the type of error and then abort the current operation

• The errors that might be in a HTML page result might be easily recognized automatically like HTTP 404. Other error messages are hard to recognize, this may be embedded within a series of tables, frames or other types of HTML division. Users can sometimes understand the messages, but automated understanding is very hard.

• The results coming from a HTML page may contain duplication of information, which we should discard. Section 3.2 shows how the system detects and solves the duplication error

• In certain circumstances, the server may require authorization information for logging on to the system.

3.2. Detection and removing of Result Duplication

Once the query is sent to the relevant web-sources, the data retrieved is placed into a repository discussed in [1, 3]. Data retrieved for multiple submissions of a form may contain duplication; the system eliminates this duplication of data before placing the result in the repository by using the detection mechanism described in [2], which is highly effective for finding duplicate sentences over a large set of textual documents. The system analyses systematically the data returned from a Deep Web query then calculates the hash value for each result and then removes the duplication [1]. The data retrieved from behind web forms is usually displayed as paragraphs separated by the HTML paragraph tag <p>, as rows in a table separated by <tr> </tr> tags, or as blocks of data separated by the <hr> horizontal rule tag.

Stephen W. Liddle, David W. Embley, Del T. Scott and Sai Ho Yau in [1] proposed a way of dealing with a special tag called the sentence boundary separator tag in order to adapt the copy detection system for collection of records. During the duplication detection process, the system inserts this special tag into a retrieved web document around certain HTML tags that most likely delimit the duplicate record.

The tags chosen for this treatment include </tr>, <hr>, <p>, </table>, </blockquote> and </html>. If none of the above tags except </html> appears in the document, the whole document is considered to be a single record. The idea above of handling duplicate recognition and elimination has been discussed in more detail in [1].

IV. Results Processing

This section deals with how the results are being processed from a web form once the query has been submitted by the user. A parser [6] will analyze different formats of data page returned by web databases in order to extract the relevant pieces of information out of forms. Once extraction of the data from different web-sites is done, next step is to merge those data into a single response page; this idea is detailed in section 5 called "Data Integration" Result processing can be split into three components, which are:

• *Result extraction*: this component will identify and extract the relevant results from the response pages returned by web databases

• *Result annotation*: this component will append the proper semantics for the extracted result

• *Result merging*: merge results extracted from different web databases into a single response page.

V. Web Data Extraction

Web data extraction from the Deep Web has been tackled by many people in related work e.g. [5]; it seems that more work must still be done in this area. Raghavan, S. and Garcia Molina at Stanford University [3, 4] developed the "Hidden Web Exposer (HiWe)" system that builds a Deep Web crawler that automatically parses, processes and interacts with form-based search interfaces. Because of the formidable challenges to a fully automatic process, HiWE assumes that crawls will be domain specific and human assisted. Although HiWE must start with a user filling in the form for a search task, HiWE learns from successfully extracting information and updates the task description database as it crawls.

Besides an operational model of a Hidden Web Crawler, other more interesting contributions are:

• *The label matching component* used for matching labels entered on the form to those labels in the Label Value Set table.

• *Internal form representation*, the crawler breaks up a query form into several information pieces. The form is represented by F= ({E1… En}, S, M) where {E1… En} represents a set of n elements, S is the submission information associated with the form, and M is metadata information about the form. Each element of the set E has two pieces of information called domain Dom (Ei) and

Label (Ei). Domain refers to a set of values Ei can take on and Label is the description associated with a domain value.

• *Task-specific database*, the HiWE crawler uses a task specific database. This database stores all relevant information that helps the crawler to formulated search queries relevant to particular task.

• *Response analysis*, this component stores the page result in the crawler's repository.

Similar work has been discussed by S. W. Liddle et al [1]. They proposed a way to extract the data behind web forms. Their main contribution reveals how to retrieve the data behind a particular HTML form; how to process a result page returned by a form submission. This includes, for example, error detection.

VI. Data Integration

Data integration is the problem of combining data from various web databases sources, and providing users with a unified view of data [15, 16, and 17]. One of the main tasks in designing a data integration system is to establish the mapping or relation between the web database sources and a global schema, which must be taken into account in formalizing a data integration system.

Many people have explored this point of deep web data integration and many solutions are discussed. However they state that the challenging part remains "schema matching" for discovering semantic correspondences of attributes across heterogeneous sources. Bin He and Kevin Chen-Chuan Chang [7] addressed the "problem of automatic matching process" which integrated the DCM (Dual Correlation Mining Algorithm) framework with an automatic interface extractor. Such system integration turns out to be nontrivial– As automatic interface extraction cannot be perfect, it will introduce erroneous extraction, which challenges the performance of the subsequent matching algorithm. However, Stephen W. Liddle, David W. Embley, Del T. Scott and Sai Ho Yau [1] proposed that "it is necessary to automate extraction and integrate information data from different web databases".

Related work of data integration has been discussed by S.Raghavan and H. Garcia- Molina [3, 4]; their Crawling the Hidden Web gives more significant contribution to the data integration in the development of Deep Web data integration.

Maurizio Lenzerini [15] discussed in his "theory of data integration" the main components of a data integration system that are a Global Schema, web databases Sources and mapping. He formalizes a data integration system I in a triple (G, S, M) where G is the global schema, expressed in a language LG over an alphabet AG; S is the database sources, expressed in a language LS over an alphabet AS; and M is the mapping between G and S, constituted by a set of assertions of the forms {qS, qG} or {qG; qS} where qS and qG represent two queries respectively over the source S and over the global schema G.

VII. Suggested Work

Here, the goal is to extract the data from various hidden web databases and this data in integrated form will be stored in large repository with no duplicate records.

Search Query Interface is considered as an entrance to the websites that are powered by backend databases. User can find the desired information by submitting the queries to these interfaces. These queries are constructed as SQL queries to fetch data from hidden sources and send it back to user with desired results. The proposed approach is presented in four phases. Firstly, different query interfaces are analyzed to select the attribute for submission. In the second phase, queries are submitted to interfaces. Third phase extracts the data by identifying the templates and tag structures. Fourth phase integrates the data into one repository with all duplicate records removed. There can be various methods to submit queries.

7.1 Different Query Methods:

Blank query:

Blank query means no field is selected while submitting query to the interface form. This will extract the whole database at once. In this case, we can leave all the fields blank and press the submit button. But it is seen that most of the sites don't accept this kind of input. Many sites contain restrictions like "please select city" or "please select any one option". Here, in this case city is mandatory field.

Query with all combinations:

Second type of query selection can be selection of specific values of all fields. For example, in case of car domain (make="maruti", model="alto", city ="New Delhi") is selected and then this query is submitted. This kind of input gives us very accurate result. But this needs all combinations to be done prior to submission and there can be million of such combinations. Because we are dealing with Query interfaces that have multiple attributes and each attribute contains large number of values. So, this would be tiresome task.

Query selection with mandatory field:

Third type of query selection can be selection of only mandatory field. It is observed that most of the sites have one compulsory field that should be selected and if values of this field are filled and submitted, it will give us the whole database and this retrieved database can be used for later searching. To maintain the uniformity, same field is selected in all local interfaces for submission. The selected field should be field which is seen as mandatory field in most of the sites.

7.2 Architecture for data extraction and integration approach:

An interface in integrated form would provide uniform access to the data sources of a given domain of interest. Because some sites have restriction over the inputs, we cannot submit the blank form. So, Crawler submits the values of mandatory field and extracts the results. Mandatory field is selected and all the option values are filled to Global Interface which is formed by schema matching of all the local interfaces. These values are now submitted to local interfaces of all sites and results are then extracted. Every local site will send its result into local database (table). Now, the large repository will be made to fetch all the data from local databases and make it global.

In HTML, tables are defined with the <table> tag. A table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. The data stored in backend databases is in structured form (table form). So, we can extract the table data from the database by looking at the tags. So, we will extract results which lies inside the <tr>….</tr> tags and all the rows are extracted which lies inside, <td>….</td>

7.2.1 Removing Duplicate Records:

Data is extracted from all the local databases by submitting same query to respective local interfaces. However, it is very much possible that many of these sites contain same results or same tuples. Hence, data repository should be made in such a way that duplicate records are removed while merging. To remove duplicate records, Sql query is fired and all the distinct tuples are inserted into data repository as shown below in fig 9. Sql query is shown below.

insert into table3 select * from table1 union select * from table2;

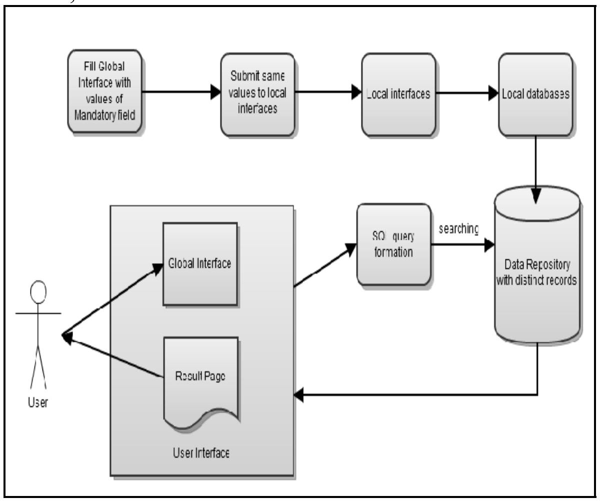
Fill Global Interface with values of Mandatory field
Submit same values to local interfaces
Local interfaces
Local databases
Global Interface
Result Page
User Interface
User
SQL query formation
searching
Data Repository with distinct records

Fig 7.1: Architecture for data extraction and integration

7.2.2 Query Formation:

One template is made for construction of query. When user gives the keyword in search box of the search engine, Search engine responds with the Global search interface form for specific domain which contains certain attributes and will be filled by the user. If it is partially filled, then it will be filled with all the permitted values. SQL query is made automatically using attribute-value pairs in global interface by query generator. This query is fired on data repository as shown below.

Select * from from table3 Where a1='x1' and a2='x2' and a3='x3';

Where a1, a2 ,…. are the attributes and x1, x2, x3 are the values filled in the form.

VIII. CONCLUSIONS

In this paper, querying the deep web for web database sources has been discussed. Such a system contains three components, Interface integration, Query formulation and Result interpretation.

Interface integration produces a unified interface over the query interface of the web databases from a single domain such as airline booking and analyzes the different web pages.

Query formulation involves schema integration and formulates the query to be sent to the various web-databases sources.

Result interpretation extracts the page results from different web-databases sources after query submission and then merges the data into a global consolidated result. The discussion in this paper concentrated on the Result component of Deep Web query. Once the query to the Deep Web is submitted, the system must find the relevant fields of records and match them to fields of the global schema, then extract field values into a repository and then display as an integrated result. In addition problems to handle include duplication in the results; the system must provide a mechanism of handling duplications and errors before integrating the result into global consolidated result. In the case of a missing field error, user intervention is required.

REFERENCES

[1] Stephen W. Liddle, David W. Embley, Del T. Scott and Sai Ho Yau. Extracting Data behind Web Forms; Proceedings of the 28th VLDB Conference, pp. 2-11, Hong Kong, China, 2002

[2] D.M. Campbell, W.R. Chen, and R.D. Smith. Copy detection system for digital documents. In Proceedings of the IEEE Advances in Digital Libraries (ADL 2000), pages 78-88, Washington, DC, May 2000

[3] S.Raghavan and H. Garcia-Molina. Crawling the hidden Web. In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB), Pages: 129 - 138, Rome, Italy, September 2001.

[4] S.Raghavan and H. Garcia-Molina. Crawling the hidden Web. Technical Report 2000- 36, Computer Science Department, Stanford University, December 2000. Available at http://dbpubs.stanford.edu/pub/2000-36.[Thursday 17th July 2007]

[5] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In Proceedings of the ACM SIGMOD Conference on Management of Data, pages: 337 – 348, year of publication: 2003

[6] Web Group, Deep Web data Integration, Dataspaces, WAMDM lab, Institute of Data and Knowledge. http://idke.ruc.edu.cn/projects/web.htm [Monday 12th May 2008 10h30]

[7] Bin he and Kevin Chen-Chuan Chang. Automatic Complex Schema Matching Across Web Query Interfaces: A Correlation Mining Approach, pages: 346 – 395. University of Illinois at Urbana-Champaign Year of Publication: 2006, ISSN: 0362-5915.

[8] Wensheng Wu, AnHai Doan and Clement Yu. WebIQ: Learning from the Web to Match Deep-Web Query Interfaces. 22nd International Conference on Data Engineering (ICDE'06), pp.44, April 3-7, 2006. University of Illinois, USA

[9] Wu, W., Yu, C., Doan, A. and Meng, W. 2004. An Interactive Clustering-based Approach to Integrating Source Query Interfaces on the Deep Web. Proceedings of the 2004 ACM SIGMOD international conference, pages: 95 – 106, year of Publication: 2004.

[10] Wu, P., Wen, J., Liu, H. and Ma, W. Query Selection Techniques for Efficient Crawling of Structured Web Sources. Available online: http://research.microsoft.com/~jrwen/jrwen_files/publications/DeepWebCrawling.PDF. [2 May 2008 18h45]

[11] Qiu, J., Shao, F., Zatsman, M., and Shanmugasundaram, J. Index Structures for Querying the Deep Web. International Workshop on the Web and Databases (WebDB) San Diego, California, June12-13, 2003.

[12] He, B, Zhang, Z, Chang, KC. MetaQuerier: Querying Structured Web Sources On-the-fly. Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages: 927 – 929, year of Publication: 2005. University of Illinois, USA.

[13] Myllymaki, J. Effective Web Data Extraction with Standard XML Technologies Proceedings of the 10th international conference on World Wide Web, pages: 689 – 696, year of Publication: 2001. Hong Kong, Hong Kong.

[14] LIU Wei (1976-), Male, Ph. D. candidate, research direction: Web data integration

http://www.springerlink.com/content/t2322q344kv405l5/ [Monday 14th July 2008 18h44]

[15] Maurizio Lenzerini. Data Integration: A theoretical Perspective. Proceedings of the twentyfirst ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, pages: 233 - 246, year of Publication: 2002. Università di Roma La Sapienza, Via Salaria 113, I-00198 Roma, Italy.

[16] A. Y. Halevy. Answering queries using views: A survey. Very Large Database J., 10(4):270–294, 2001.

[17] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. On Principles of Database Systems (PODS'97), 1997.