A Novel Technique for Data Extraction from Hidden Web Databases

Anuradha
Department of Computer Engineering
YMCA University of Sc. & Technology
Faridabad, India

A.K.Sharma
Department of Computer Engineering
YMCA University of Sc. & Technology
Faridabad, India

ABSTRACT
The large amount of information on web is stored in backend databases which are not indexed by traditional search engines. Such databases are referred to as Hidden web databases and extraction of this hidden web content is a potential research area as the pages are dynamically created through search query interfaces. However, direct query through this search interface is laborious way to search. Hence, there has been increased interest in retrieval and integration of hidden web data with a view to give high quality information to the web user. This paper proposes a novel approach that identifies Web page templates and the tag structures of a document in order to extract structured data from hidden web sources as the results returned in response to a user query are typically presented using template generated Web pages.

Keywords: Hidden web, Deep web, Global interface, Hidden web crawlers, Surface web.

1. INTRODUCTION
Hidden web data, stored in structured or unstructured databases [4], is inherently hidden behind search forms. It is qualitatively and quantitatively different from the surface Web. The quality content of the deep Web is 1,000 to 2,000 times greater than that of the surface Web whereas overall the hidden web contains approximately 7,500 terabytes of data and 550 billion individual documents in contrast to the surface Web, which is reported to about 167 terabytes [4]. Since the hidden web is the biggest source for structured data and is not publically indexed yet, accessing the same is a challenging task especially when the pages are created dynamically through search interfaces.

The traditional search engines use inverted index as a data structure to index the web data and keyword interface to retrieve the data. But, surfacing the Hidden Web is more difficult task in many respects. First, the index structures for the hidden web deal with the structured data as well as the large volume of data. Second, the search query interfaces often have more than one attribute and requires their respective values to be submitted. For instance, suppose a user wants to search some information hidden behind a search interface on a site. The user needs to follow the following steps.

1. He/ She has to discover the URLs of the hidden websites.
2. Visits homepages of these web sites.
3. Send queries through HTML forms.
4. Extract the relevant information from the result web pages.
5. Compare or integrate the results from multiple sources.

Therefore, there arises the need for new information services that can help users to find the information in integrated form. To minimize user effort, the problem of automatically interaction with information sources in the hidden web is explored in this paper.

The paper has been organized as follows: Section 2 describes the related work in the area of hidden web data extraction. Section 3 describes the proposed work i.e a method to extract hidden web data in integrated form. Section 4 draws the conclusion.

2. RELATED WORK
This section discusses the work that is closely related to the proposed work. Although there are many proposed hidden web crawling techniques, a very little work is done on the indexing of hidden web data.

[2] automatically detects the domain specific search interfaces by looking the domain word in the URLs, then title and after that attributes of the source code. Feature space model classified the web pages into a set of categories using domain ontology. The large-scale collections of query interfaces of the same domain are then integrated into integrated search interface (ISI). This interface will permit users to access information uniformly from multiple sources of a given domain.

[3] also presents new index structures for querying the hidden web. But this study again considered only single attribute. Here, clustering of data is done to compress the index. But this technique is inefficient for storing the multi-attribute based hidden web data.

In [11], Raghavan and Garcia-Molina presents an architectural model for extracting hidden web data. The main focus of this work is to learn Hidden-Web query interfaces, not to generate queries automatically. Their approach is not automatic and requires human input. [11] addressed only the problem of crawling.

HiddenSeek [15] uses a keyword based indexing and searching technique for single-attribute hidden web sites. This approach uses the inverted index for indexing and searching method the hidden web data. HiddenSeek takes a term frequency of keyword as a factor for ranking the results i.e, whether the keyword appears in the URL of a page.

Siphon++ [16] proposed a strategy for automatically retrieving data hidden behind keyword-based form interfaces. It considers the query generation and selection techniques by detecting features of the index. Unlike multi-attribute forms, keyword-based interfaces are simple to query, since they do

not require detailed knowledge of the schema or structure of the underlying data. It is thus easy to create automatic and effective solutions to crawl these interfaces.

[17] states that there are a growing number of pages that can't be indexed by search engines and stay invisible to other surfers, despite the fact that they contain a lot of relevant content. Pages with dynamic content i.e the pages that are the result of a submitted query and consequently do not have a static URL can be impossible for search engines to find, since the crawlers cannot replicate the query submission carried out by human beings. Generally traditional search engines follow links to the index page on a site and then crawl from there to other pages by following links. Search engine crawlers will therefore have more difficulties seeing a page that is not linked to from any other page(hidden web pages).

The GlOSS system [18] supports data source discovery for text documents. This approach constructs source discovery servers for boolean text databases. Based on compact collected statistics, these servers can provide very good hints for finding the relevant databases for a given query. The Niagara system [19] uses an index structure that identifies a superset of data sources relevant to a user query. However, these systems do not support structured queries i.e multi-attribute queries.

3. PROPOSED WORK :

The goal of this research 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.

3.1 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. This kind of restriction is shown below in fig 1.

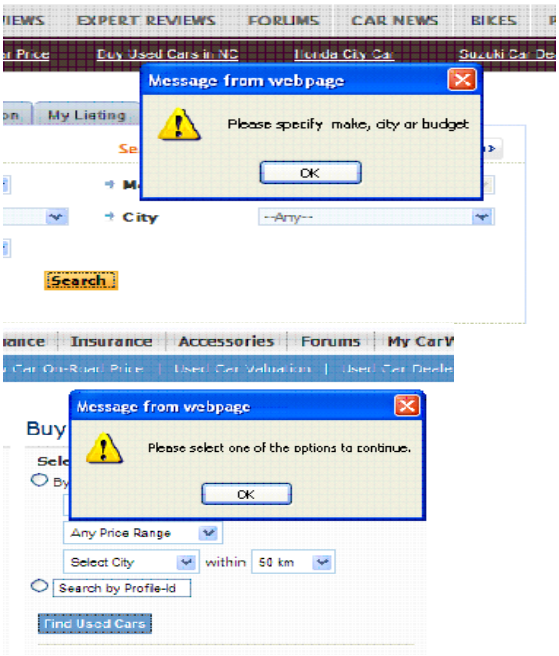

Fig1: Example of search interfaces with mandatory field

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.

3.2 Architecture for data extraction and integration approach:

An interface in integrated form [1] 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 [1]. 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>

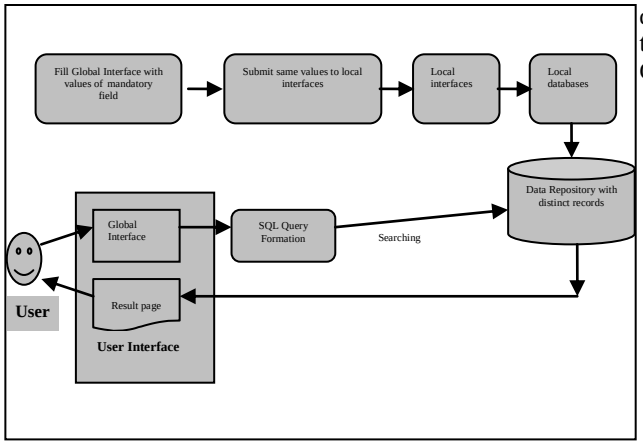

Fig2: Architecture for data extraction and integration

3.2.1 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 2. Sql query is shown below.

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

3.2.2 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 [1] 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.

4. CONCLUSION

In this paper we proposed a new information retrieval service that can help users to find the desired information in integrated form. To minimize user effort, the problem of automatically interaction with hidden web sources is explored. In this paper hidden web data extraction method has been discussed. This system contains four components, Interface analyses, query selection, query submission, result extraction. Repository is formed by the data extracted from various sites. In addition to this, duplicate records are also removed from repository and this repository is prepared for user search. Later on, When user fills the global form for searching, SQL query is fired o this repository to get the desired result. Hence, the user's effort is minimized by just filling only one form i.e Global interface.

5. REFERENCES

[1] Anuradha, A. K. Sharma, Komal Kumar Bhatia: "Optimized Merging of Query Interfaces for Domain-specific Hidden Web "Proc. Third International Conference on Advanced Computing and communication Technologies (ICACCT 2008) Volume 2, No. 2, pp. 196-199

[2] Anuradha, A.K.Sharma, "A Novel Approach for Automatic Detection and Unification of Web Search Query Interfaces using Domain Ontology" selected in International Journal of Information Technology and knowledge management(IJITKM), August 2009.

[3] Jian Qiu, Feng Shao, Misha Zatsman, Jayavel Shanmugasundaram, Index Structures for Querying the Deep Web, Workshop on the Web and Databases (WebDB), 2003, 79-86

[4] BrightPlanet Corp. "The deep web: surfacing hidden value."

[5] D. Florescu, A.Y. Levy, and A.O. Mendelzon. "Database techniques for the world-wide web: a survey," SIGMOD Record 27(3), 59-74, 1998.

[6] J. Wang and F. Lochovsky. "Wrapper Induction based on Nested Pattern Discovery," Technical Report HKUST-CS-27-02, Dept. of Computer Science, Hong Kong U. of Science & Technology, 2002 (submitted for publication).

[7] C.H. Chang, and S.C. Lui. "IEPAD: information extraction based on pattern discovery," Proc. 10th World Wide Web Conf. 681-688, 2001.

[8] V. Crescenzi, G. Mecca and P. Merialdo. "ROADRUNNER: towards automatic data extraction from large web sites," Proc. 27th Intl.Conf. on Very Large Data Bases, 109-118, 2001.

[9] D. Embley, Y. Jiang and Y.K. Ng. "Recordboundary discovery in web documents," Proc. ACM SIGMOD Conf., 467-478, 1999.

[10] He, K. Chang, and J. Han. Discovering complex matchings across web query interfaces: A correlation mining approach. In SIGKDD, 2004.

[11] S. Raghavan and H. Garcia-Molina. Crawling the Hidden Web. In Proceedings of VLDB, pages 129–138, 2001.

[12] B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In Proceedings of SIGMOD, pages 217–228, 2003.

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

[14] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In SBBD, 2004.

[15] Ntoulas, A., Zerfos, P., Cho, J. Downloading Textual Hidden Web Content Through Keyword Queries. In Proceedings of the 5th ACM/IEEE Joint Conference on Digital Libraries (JCDL05). 2005.

[16] L. Barbosa and J. Freire. Siphoning hidden-web data through keyword-based interfaces. In SBBD, 2004.

[17] L. Gravano, H. Garcia-Molina, A. Tomasic, "GlOSS: Text-Source Discovery over Internet", TODS 24(2), 1999.

[18] J. Naughton et al, "The Niagara Internet Query System", IEEE Data Eng. Bulletin, 24(2), 2001.

[19] Brin, Sergey and Page Lawrence. The anatomy of a large-scale hypertextual Web search engine. Computer Networks and ISDN Systems, April 1998