HIDDEN WEB EXTRACTOR DYNAMIC WAY TO UNCOVER THE DEEP WEB

DR. ANURADHA
YMCA,CSE, YMCA University
Faridabad, Haryana 121006,India
anuangra@yahoo.com
http://www.ymcaust.ac.in

BABITA AHUJA
MRCE, IT, MDU University
Faridabad, Haryana 121004, India
Babita.spark@gmail.com
http://www.mrce.ac.in

Abstract - In this era of digital tsunami of information on the web, everyone is completely dependent on the WWW for information retrieval. This has posed a challenging problem in extracting relevant data. Traditional web crawlers focus only on the surface web while the deep web keeps expanding behind the scene. The web databases are hidden behind the query interfaces. In this paper, we propose a Hidden Web Extractor (HWE) that can automatically discover and download data from the Hidden Web databases. Since the only "entry point" to a Hidden Web site is a query interface, the main challenge that a Hidden Web Extractor has to face is how to automatically generate meaningful queries for the unlimited number of website pages.

Keywords: WWW; Hidden Web; Surface Web; Query Interface; Crawler; Semantic Web; XML;RDF.

I. INTRODUCTION

The hidden web databases are escalating every day. This had made the Hidden Web a centre of attraction to the researchers. Since data residing in Web databases is guarded by search interfaces and inaccessible to conventional Web crawlers. This portion of the Web is commonly referred to as "deep Web" or "hidden Web". According to many studies, the size of the Hidden Web increases rapidly as more organizations put their valuable content online through an easy-to-use Web interface [1]. In [2 Chang et al. estimate that well over 100,000 Hidden-Web sites currently exist on the Web. Moreover, the content provided by many Hidden-Web sites is often of very high quality and can be extremely valuable to many users [1]. For example, PubMed hosts many high-quality papers on medical research that were selected from careful peer-review processes. The surface web contributes 1% only as the part of the total web space whereas the hidden web makes a share of 99% of the total web space as shown in Fig. 1.

II. TYPES OF INVISIBILITY IN WWW

There are certain kinds of material on the web which cannot be accessed via the internet as they are not indexed by the search engines. WWW has four types of hidden web or Invisible content [7]. The four types of invisibility are:

- The Opaque Web
- The Secret Web
- The Proprietary Web
- The Proper Invisible Web

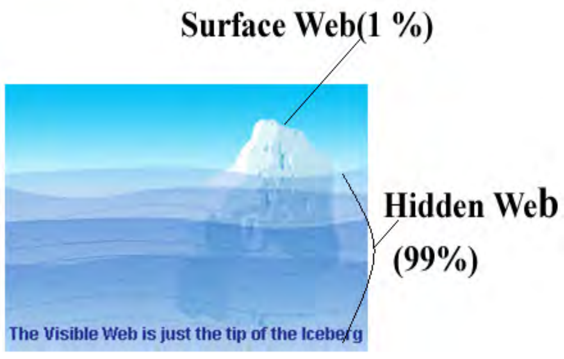


Fig. 1. The hidden web and the surface web

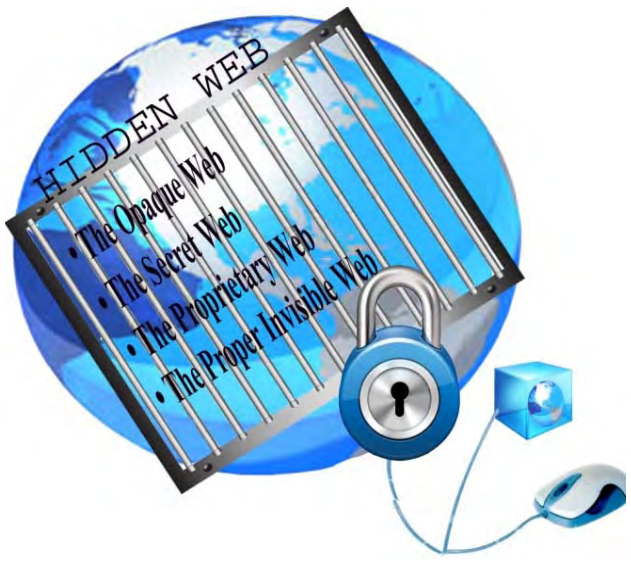


Fig. 2. Types of Hidden Web

A. Opaque Web

This web forms that part of the web that can be indexed by our web crawlers but they are not indexed by Web Crawler because of the following reasons:

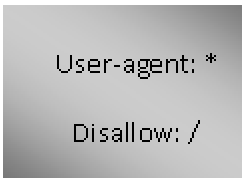
- *Depth of Crawl*
  It costs more to the search engines if they increase their depth of crawling and crawl each and every page of all websites. If they index every web page then the indexing cost also increases. Limiting the depth of crawl also reduced the cost of indexing a particular Web site.

- *Disconnected URL's*
  Some URL's are disconnected as they don't have links pointing to them from other Web pages.

- *Frequency of Crawling*
  Most search Engines Crawl the web at a lesser rate (after a month or more). As the web is very dynamic, so new web pages are added and removed at very frequent rate. These pages make a major the part of invisible web.

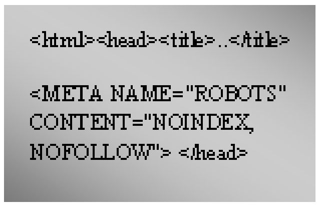
B. Secret Web

These web pages can be technically indexed but they are not indexed by the search engines as the Web Administrator doesn't wish this to happen. Web Crawler is not able to index these pages because of the following reasons:



```
User-agent: *
Disallow: /
```

Fig. 3. The Robot.txt file



```
<html><head><title>..</title>
<META NAME="ROBOTS"
CONTENT="NOINDEX,
NOFOLLOW"> </head>
```

Fig. 4. A Web page using the “Noindex Tag”

- *Robots Exclusion Protocol*
  The web Administrator can prohibit a web page to be indexed by search engine by placing the links of those pages in a robot.txt file. The robots.txt file gives instructions about their site to web robots; this is called The Robots Exclusion Protocol. If a crawler wants to visit a URL, http://www.mywebsite.com/welcome.html then before it does so, it firsts checks for robot.txt file shown in Fig. 3.

  The "User-agent: *" means this section applies to all robots. The "Disallow: /" tells the robot that it should not visit any pages on the site.

- *Meta Tag- Noindex*
  The Web Administrator uses the “noindex” Meta tag. When the crawler reads this Meta tag in the head section then they cannot read further and index the body as shown in Fig. 4.

- *Authentication*
- *Password protected web pages*

C. *The Proprietary Web*

This web has a proprietor and it is controlled as a property. If users desire to view such pages then they must register first and only then they can view such kind of pages. The registration procedure can be paid or it can be free of cost.

Web Crawler is not able to index these pages because of the following reasons:

- *Registration Procedure*
  The web Crawler cannot index such kind of web pages as the procedure of registration cannot be accomplished by the crawlers.

D. *Proper Invisible Web*

Web Crawler is not able to index these pages because of the following reasons.

1) *Format of Files*

There are some file formats which cannot be handled by today’s search engines. Ex:

- Audio Files
- Video Files
- PDF Files
- Flash Files
- Executable files

2) *Frequently changing web Pages*

There are some dynamically generated Web pages. These pages are generated by some server side scripts on the fly. So these pages can be indexed but they are not by the Search Engines. Such kinds of pages are changed within minutes, so if they are indexed then they will provide the stale information. So they are not indexed. Ex:

Share market values change very frequently so stale information will be very harmful there so they are not indexed.

3) Pages behind Query Interface:

Certain kind of information can only be accessed by the help of the query interfaces. User fills the query interface and submits the form. The information from the databases is then extracted and displayed to the users. Crawlers aren't programmed to understand either the database structure or the command language used to extract information.

III. TYPES OF QUERY INTERFACES

Some search forms list all possible values for a query (e.g., through a drop-down list), the solution is straightforward for such kind of query interfaces. We exhaustively issue all possible queries, one query at a time as shown in Fig. 5. When the query forms has text field input, then , an infinite number of queries are possible, so we cannot exhaustively issue all possible queries as shown in Fig. 6.

IV. RELATED WORK

A thorough survey of approaches for automatic schema Matching has been done [6]. They distinguish between schema-level and instance-level, element-level and structure-level, and language-based and constraint-based matchers. The MetaQuerier was designed [3]. The goal of MetaQuerier is twofold– First, to make the deep Web systematically accessible, it will help users find online databases useful for their queries. Second, to make the deep Web uniformly usable, it will help users query online databases. This system focuses on the query interface processing and the processing of the query results is not involved in detail.

In a recent study [5] present an architectural model for a Hidden Web crawler. The main focus of this work is to learn Hidden-Web query interfaces, not to generate queries automatically. The potential queries are either provided manually by users or collected from the query interfaces. The idea of automatically issuing queries to a database and examining the results has been previously used in different contexts. For example, Callan and

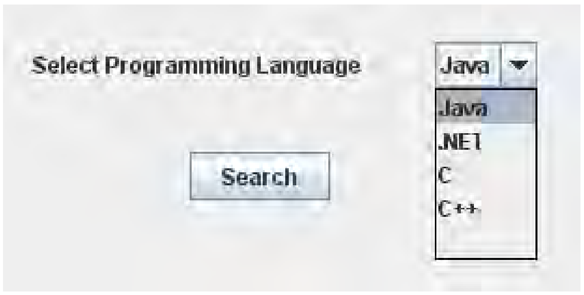
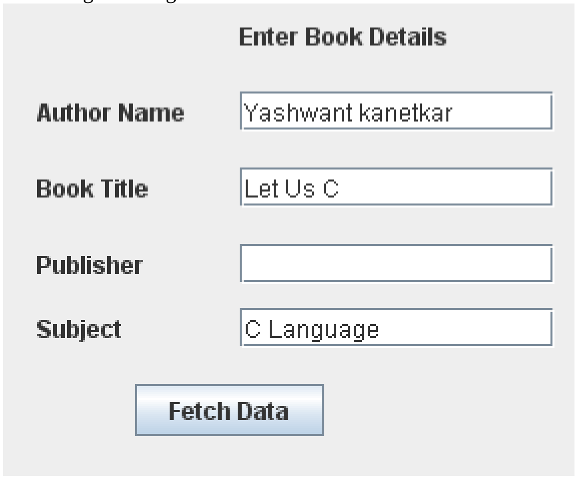


Fig. 5 A single finite value-attribute search interface



Author Name	Yashwant kanetkar
Book Title	Let Us C
Publisher	
Subject	C Language

Fig. 6. A Multiple infinite value attribute search interface

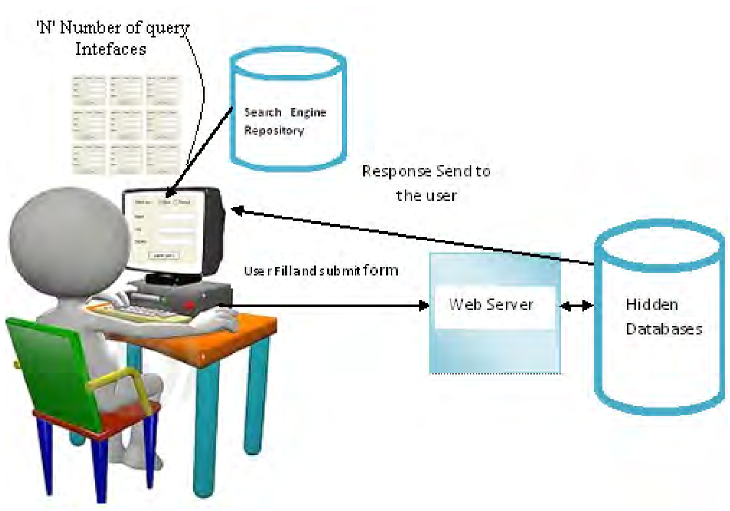

'N' Number of query Intefaces
Search Engine Repository
Response Send to the user
User Fill and submit form
Web Server
Hidden Databases

Fig. 7. Interaction of user with Query Interface

Connel [4] try to acquire an accurate language model by collecting a uniform random sample from the database their indexes.

V. PROPOSED WORK

The information of highest value contained in the deep web is behind the Web forms. The current interaction of the user with the query interface is shown in Fig. 7. The user fills the form retrieved from the page repository of the search engine, stored during the traditional crawling procedure, and submits the query which is processed at the server end. The hidden data from the web server is retrieved and the result is displayed to the user.
The proposed Technique enables the search engine to automatically fetch the information from the databases of the remote website and display them to the users. The proposed technique HIDDEN WEB EXTRACTOR (HWE) performs the basic actions fetching pages, parsing and extracting URLs and adding the URLs to a URL list are similar to those of traditional crawlers. However, whereas the latter ignore form, HWE performs the sequence of actions for each form on a page as shown in Fig. 8. HWE uses the semantics of the web pages and store their information in xml files.

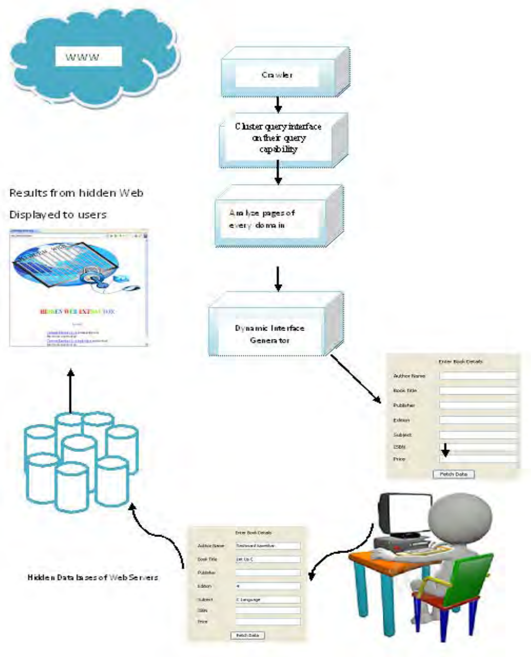

WWW
Crawler
Cluster query interface on their query capability
Analyze pages of every domain
Dynamic Interface Generator
Enter Book Details
Author Name
Book Title
Publisher
Edition
Subject
ISBN
Price
Fetch Data
Results from hidden Web Displayed to users
HIDDEN WEB EXTRACTOR
Hidden Databases of Web Servers

Fig. 8. Result of the Proposed System.

A. *Abbreviations and Acronyms*

All the query interfaces indexed by the traditional crawlers will be analyzed by the Interface parser (IP). The IP will look for the meta tags and the <h1> tags, form tags to identify the query capability of the web page. Depending on the query capability the query interface web pages will be clustered in different domains. Ex: Books, Cars, Home etc. In this paper we are taking the car domain for the purpose of discussion.

B. *Form Parsing*

The query Interface domains generated above will go through the form parsing steps discussed below.

- *Cloak Website Removal*

If a human reading the site would see different content or information than the search engine robot reading the site then that site is a cloak site. This is used to check if a document is relevant and it should be processed or not. HWE will categorize query interface (forms) web pages depending on their domain.

- *Form Tag Extractor*

This is used to pre-process the document. Generate the tokens, perform stemming, stop word removal and extract the useful attribute name from the HTML pages specific to a particular domain ex: car.

1) *Schema vocabulary of query interfaces of all domains*

Schema of every query Interface of cars along with its URL will be stored in RDF file. RDF is written in XML. This file will help in matching the schema at later steps of the algorithm. A file cardetails.xml will be created that will store the schema vocabulary of all query interfaces of car domain.

```
<? xml version="1.0"?>
<RDF>
< Description about=" http:// www.cardekho.com/ " >
  <Price></Price>
  <Brand></Brand >
  <Model></Model>
  <City></City>
<Description>
< Description about=" http://www.autocarindia.com/ ">
  <Manufacturer> </ manufacturer >
  <Model></Model>
  <Edition></ Edition >
<Description>
< Description about=" http://www.cars.com/ ">
  <Company> </Company >
  <Model> </Model>
  <Edition> </ Edition >
<Description>

<RDF>
```

2) *Schema vocabulary of words from query interface of all domains*

Let us say for car domain the list given below is generated ex: (Price, Brand, City, Model, Manufacturer, Edition, Company, Model, and Edition)

C. *Duplicate Removal*

The vocabulary prepared in step above contains the synonyms. The duplicate entries will be removed from the list and their synonyms information will be maintained in a separate file carsynonym.xml. Ex: List above contains brand, company and manufacturer which are synonyms so it will be removed from the list and at the same time synonym information will be stored in a file carsynonym.xml. The carsynonym.xml file given below will be created. The tag Brand is substituted in place of Manufacturer and Company. So the xml file created depicts that Manufacturer and Company are the synonym of Brand.

```
                             <?xml version="1.0"?>
<Brand>
      <synonym> Manufacturer</synonym>
      <synonym> Company</synonym>
</Brand>
```

The unique vocabulary will be created containing (Price, Brand, Model, City, and Edition) and will be stored in a file car.xml. Ex:

```
                           <?xml version="1.0"?>
                                 <catalog>
                                     <car>
       <Price></Price>
       <Brand></ Brand >
       <Model></Model>
       <City></City>
        <Edition></ Edition >
   </car>
</catalog>
```

D. Dynamic web Page Creation

The car.xml will be used to create a dynamic web page as shown in Fig. 9. Steps 5.2. to 5.4. will be repeated for every domain of query Interfaces.

E. User Assistance

When the user submits a query in the search Engine domains as shown in Fig. 10, the proposed system analyses the query and selects the relevant Dynamic Query Interface (DQI) from its list of domains as shown in Fig. 11. User will fill this DQI and submit this form. Some text boxes may not be of interest of the user. User can skip filling some attributes. If the DQI contains 15 attributes user can skip filling few of the attributes.

F. Query Translation

Now the query from DQI will be translated into queries of original sources as shown in Fig. 12. This translation will be done very effectively as we have maintained the detailed schema of every interface in cardetails.xml file and the synonyms details are too stored in carsynonym.xml file. So the queries will be mapped perfectly to their original sources and will be filled automatically by the proposed system.

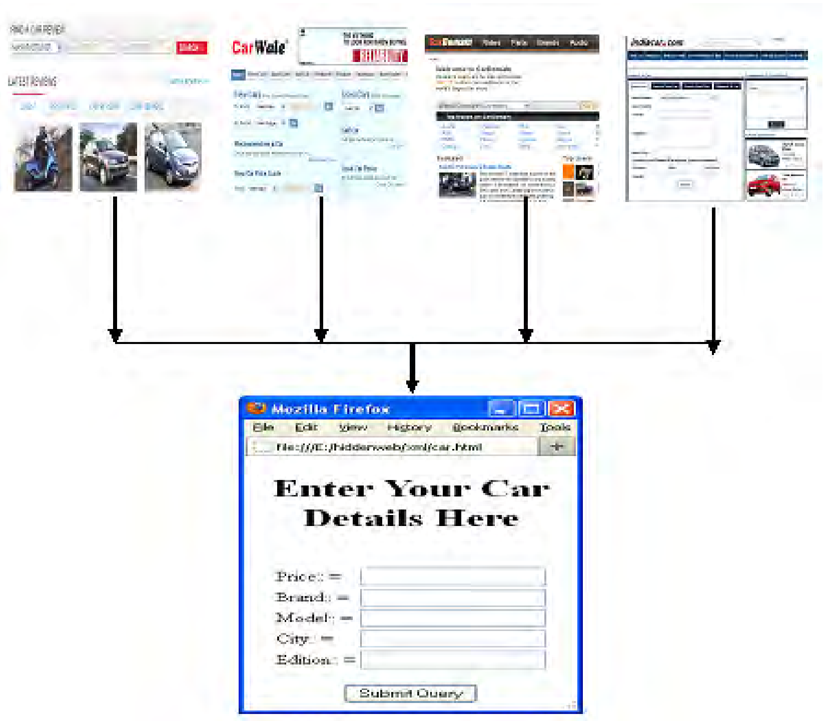
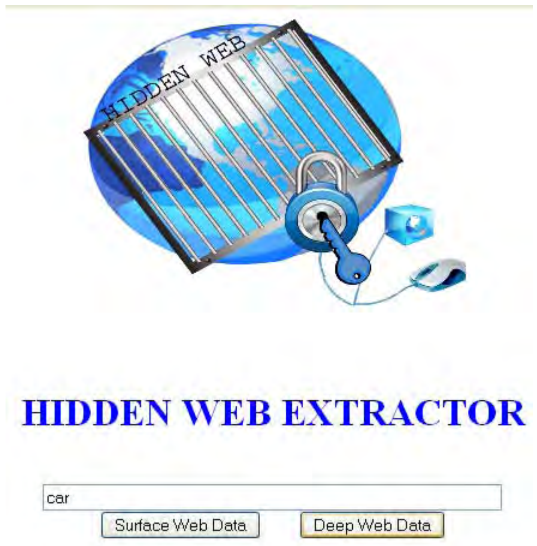

CarWale
Mozilla Firefox
File Edit View History Bookmarks Tools
file:///E:/hiddenweb/xml/car.html
Enter Your Car Details Here
Price: =
Brand: =
Model: =
City: =
Edition: =
Submit Query

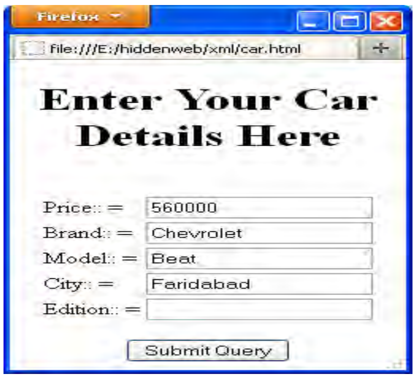
Fig.9. Dynamic Query Interface for the domain of Car.



HIDDEN WEB EXTRACTOR

car

Fig. 10. Query given in HWE

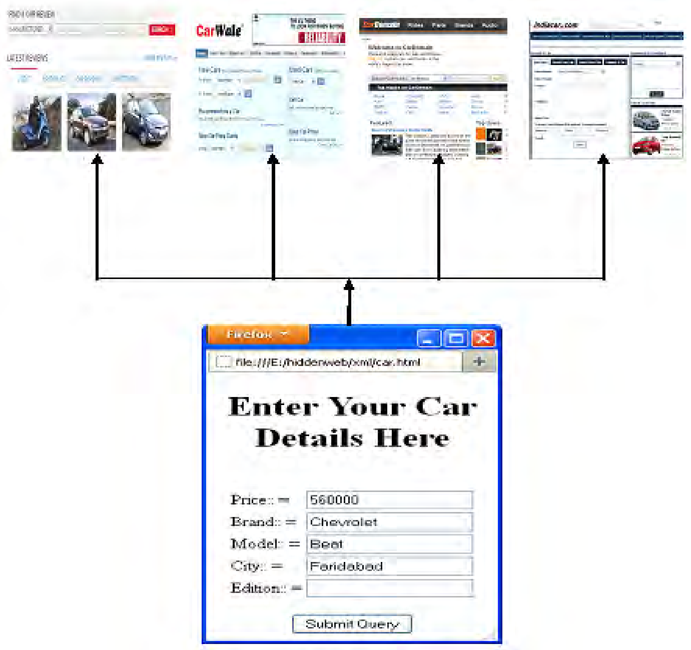


file:///E:/hiddenweb/xml/car.html

Enter Your Car Details Here

Price:: =	560000
Brand:: =	Chevrolet
Model:: =	Beat
City:: =	Faridabad
Edition:: =	

Fig. 11. Dynamic Query Interface

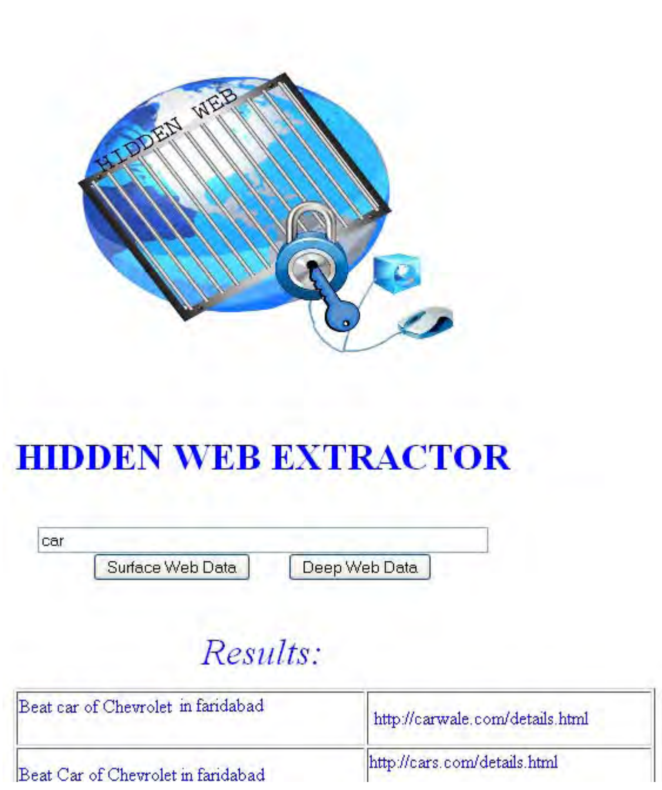


file:///E:/hiddenweb/xml/car.html

Enter Your Car Details Here

Price:: =	560000
Brand:: =	Chevrolet
Model:: =	Beat
City:: =	Faridabad
Edition:: =	

Fig. 12. Query Translation into the original sources



Beat car of Chevrolet in faridabad	http://carwale.com/details.html
Beat Car of Chevrolet in faridabad	http://cars.com/details.html

Fig. 13. Results

G. *Result Extraction*

As the user has skipped filling some attributes in the previous step so the results will be pre-processed before displaying to the users. The result obtained can contain some "No Information Page" as user has skipped filling some attributes of DQI, so the pre-processing of the results is required. It will filter the results and will remove the poor quality results.

VI. CONCLUSION

HWE retrieves the hidden data by filling all the query forms automatically. User only fills a common dynamically generated web page and the data behind the query interfaces is extracted and is presented to the users. As the semantic of the query interfaces is stored in hidden web repository, so the mapping required in query translation will be efficient and best results will be displayed to the users. Every time user send the query the information from the website servers will be retrieved, so freshness of the results will be there.

REFERENCES

[1] Bergman, M. (2001). The deep Web: Surfacing hidden value. *The Journal of Electronic Publishing*, Vol. 7.
[2] Chang, K, et al (2004). Structured databases on the web: Observations and implications. SIGMOD Record Volume 33 Issue 3, pp 61 – 70
[3] Chang, K; He, B; Zhang, Z. (2005). Toward Large Scale Integration: Building a MetaQuerier over Databases on the Web. *CIDR*, pp 44-55.
[4] Callan, P; Connell, M. (2001). Query-based sampling of text databases. *Information System. ACM* Volume 19 Issue 2, pp 97 – 130
[5] Raghavan, S; Molina, H. (2001). Crawling the hidden web. *VLDB Journal*
[6] Rahm, E; Bemstein, P. (2001). A Survey of Approaches to Automatic Schema Matching. VLDB Journal, 10, pp 334- 350.
[7] Sherman, C; Price, G. (2001). "Uncovering Information Sources Search Engines Can't See". Cyber Age books Second printing.

AUTHORS PROFILE

First Author: Dr. Anuradha ;PHD ;Assistant Professor in YMCA,India.
Second Author: Babita Ahuja;M.Tech; Assistant Professor in MRCE, India.