Attribute-Value Based Domain-Specific Indexing Technique for Hidden Web

Arnika Jain

Abstract— The Hidden Web is the content on the Web that is not accessible through a search on general search engines. The data retrieved through hidden web is structured and the indexing techniques used to index the unstructured data are of no use in the case of structured data. Index structures for the hidden web differ in two fundamental respects from traditional inverted list index structures used by current web search engines. First, index structures for the hidden web have to deal with structured data because the underlying database is typically richly structured and typed; this is in contrast to the mostly unstructured HTML data available off the surface web. Second, index structures for the hidden web must deal with data volumes that are orders of magnitude larger than that for the surface web. To address these issues, This Paper proposes an index structures for the deep web. This index structure understands the structure of the underlying data. The index structures can also be heavily compressed so that their space requirements are far less than the size of the original index. But the main problem is how to index the data retrieved through hidden web. This Paper also proposes an indexing technique for the same where data is indexed without having duplicates. This Paper also proposes a Data Extraction Architecture to extract the data of user's interest of a specific domain.

Index Terms— Crawler, Deep Net, Hidden Web, Search engine, Surface Web, Query Interface, WWW

1 INTRODUCTION

THE Hidden Web [1] refers to World Wide Web content that is not part of the Surface Web, which is indexed by standard Search Engines. Hidden Web is also called as Deep Web, Deep Net, Invisible Web, or the Undernet. It describes the portion of the World Wide Web that is not visible to the public or has not been indexed by the search engines. Surfacing the Deep Web poses several challenges. First, the goal is to index the content behind many millions of HTML forms that span many languages and hundreds of domains. This necessitates an approach that is completely automatic, highly scalable, and very efficient. Second, a large number of forms have text inputs and require valid input values to be submitted. Since the data retrieved through Hidden web is structured and in bulk, so there is need of an efficient indexing technique to index that data. This paper introduces such an indexing technique and the data extraction architecture that extracts the data corresponding to different users based on their respective requirements specific to a particular domain.

Arnika Jain is currently working as Assistant Professor in Department of Computer Science & Engineering at SRM University, NCR Campus, Modinagar, India. She has 8 years of Teaching Experience. She has received Degree in Master of Technology (Gold Medalist) from Shobhit University, Meerut, India in 2011 and Master of Computer Applications (Gold Medalist) from Gurukula Kangri Vishwavidyalaya, Hardwar (Uttarakhand), India, in 2004. Her Research Areas are MANET, Hidden Web, and Network Security. email: jain.arnika2009@gmail.com

This paper has been organized as follows: Section 2 describes the related work done so far in the area of hidden web data extraction. Section 3 describes the proposed work i.e. Architecture for Data Extraction and a method to extract hidden web data in integrated form corresponding to different user's requirements. Section 4 draws the conclusion.

2 RELATED WORK

This section discusses the work that has been done regarding the hidden web.

Anuradha and A. K. Sharma [2] introduced a Search Query Interface which is considered as an entrance to the websites that are powered by backend databases. Users 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 users with desired results.

Bhatia [3] proposed a Domain–Specific Hidden Web Crawler that automated the process of downloading of search interfaces, finding the semantic mappings, merging them and filling the Unified Search Interface produced thereof has been designed that finally submits the form to obtain the response pages from Hidden Web.

Anuradha and A. K. Sharma [4] proposed a novel method which extracts the individual data units from table data using DOM tree structure of the web page. After collecting data from each table of website, a large repository has been maintained which contains data from various Hidden web

sources of same domain and this repository is used for later searching.

Anuradha and A. K. Sharma [5] detected the domain specific search interfaces considering 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.

Jian Qiu, Feng Shao, Misha Zatsman, Jayavel Shanmugasundaram [6] presented some index structures for querying the hidden web.

HiddenSeek [7] used 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.

Xiang Peisu [8] proposed model of forms and form filling process that concisely captures the actions that the crawler must perform to successfully extract Hidden Web contents. . It described the architecture of the deep web crawler and described strategies for building (domain, list of values) pairs.

So, the retrieval of data from hidden web considers only single attribute and these technique are inefficient for storing the multi-attribute based hidden web data respective to a number of domains. So, there is a need to design a general query interface that can take the query corresponding to any domain and search the data of the user's interest. This paper proposes architecture for the data extraction in Hidden web and also defines the various types of queries that can be processed in a uniform search query interface.

3 PROPOSED WORK

Since the hidden web is the biggest source for structured data and is not publicly indexed yet, accessing the same is a challenging task especially when the pages are created dynamically through search interfaces. The problem is how to design an interface for hidden web that can take query respective to any domain and return the result corresponding to that domain only. This Paper proposes the solution to it. First task is to read the files containing the data of hidden web that is in structured form of various different domains and to index this data. Then a search query interface is designed that can take the query regarding any domain and return the data specific to that domain only.

3.1 System Architecture for Data Extraction

Several online databases provide dynamic query-based data access through their query interfaces, instead of static URL links. This Query interface is considered as an entrance to Hidden Web, as the tremendous amount of information is hidden behind these search forms in web pages and traditional crawler cannot replicate the query submission carried out by human beings. A Web search interface for e-commerce typically contains some HTML form control elements such as textbox i.e. a single line text input, radio button, checkbox and selection list i.e. a pull-down menu that allow a user to enter search information.

A Search interface would provide uniform access to the data sources of a given domain of interest. Here, a Query is considered which contains some terms and it should not be blank. The User enters the query in the text box of the Search Query interface which is then accessed by Query Processor. The two main steps done by Query Processor are extracting the Query String and then tokenizing it. The Tokens are then matched with the attribute values stored in the repository to retrieve the posting lists. These lists are then intersected by the query processor to return the result to the user with the help of a Result Page. This architecture is shown in Fig 1.

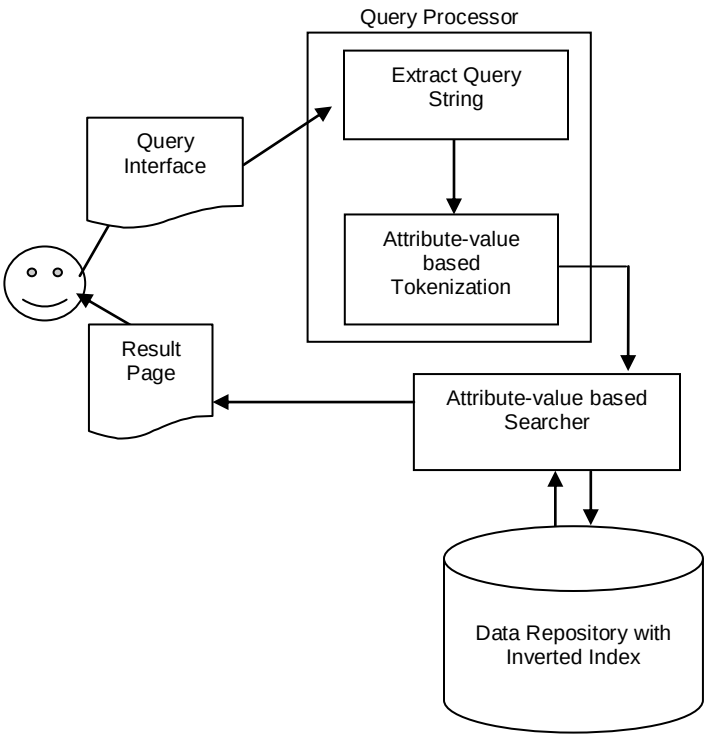


Fig 1. Architecture for Data Extraction

There are mainly five components of the Proposed Architecture of Data Extraction as shown below:

Query Interface: A Query Interface is designed as an entrance to the Project which is used by the user to enter the query. It is

the user interface where the user has to input the query to get the required data. This interface contains a Text Box where the user has to place the query. User need not specify the particular domain explicitly; it should be put as a part of the query like "Flight from Delhi to Bombay". In this particular query, the domain is itself a part of the query.

Result Page: After the query has been processed by the query processor and the Attribute-value based Searcher, the required data is returned to the user in the form of result page.

Query Processor: The user queries are processed by the Query Processor to fetch the desired data and return it back to the user with desired results. It has the following two components:

1. Extract Query String: The Extract Query String module of the Query Processor extracts the Query String from the Query Interface and passes it to the Attribute-value based Tokenization module.
2. Attribute-value based Tokenization: It tokenizes the Query string into a number of tokens. It then identifies the Domain of the query by analyzing its first token and passes the various tokens to the Attribute-value based Searcher for searching.

Attribute-value based Searcher: Attribute-value based Searcher matches the tokens respective to the domain with the various attribute and their corresponding values stored in the Data Repository to retrieve the postings lists which are then intersected to return the result page to the user.

Data Repository with Inverted Index: The Data Repository stores the various postings lists of the attribute values along with the attributes and their respective domain.

The Search interface allows a user to search some set of items without altering them. The user enters a query by typing to get the data of interest. Result Page is a page containing data of interest. This Paper implements attribute-value based domain-specific indexing technique for hidden web.

This Paper presents the index structure of hidden web for two domains: *Airline* and *Book*. It takes the files corresponding to different domains where each domain is described with the help of some attributes. Each file is treated as a Text Document and is assigned a unique docID as soon as its file path is added to the xml file. The data of these Files is arranged in such a way that the first line of the file represents the domain to which the file belongs and the next line contains the various attribute names of the respective domain and the rest of the file constitutes the various values of these attributes. One file corresponding to Airline domain is shown in Fig 2 and another corresponding to Book domain is shown in Fig 3. Any number of files can be indexed but the file which is to be index, its path is just added to the xml file.

Airline

Name	Departure City	ArrivalCity	Time1	Time2
KingFisher	Delhi	Bombay	09:30AM	11:30 AM
Indigo	Jammu	Kargil	03:50PM	05:00 PM
JetAirways	Bangalore	Kerala	01:15PM	02:30PM

Fig 2. A File corresponding to Airline Domain

Book

Title	Author	Volume	Price
OS	Galvin	Third	$12
Database	Navathe	Fourth	$15
Architecture	KaiHwang	Second	$17
C Programming	Kanetkar	Second	$18

Fig 3. A File corresponding to Book Domain

There can be a number of Synonyms corresponding to each domain and corresponding to attribute names. The synonyms for Airline Domain are Flight and Airways. The Attributes corresponding to a file of Airline domain are *Name* representing the Flight Name, *DepartureCity* representing the City from where the Flight departs (Synonyms: Leaving From, From City), *ArrivalCity* representing the City to where the Flight arrives (Synonyms: Coming At, To City), *Time1* representing the Time of Departure of Flight from Departure city and *Time2* representing the Time of Arrival of Flight to Arrival city. Likewise, the synonyms for Book Domain are Publications and Papers. The Attributes corresponding to a file of Book domain are *Title*, *Author*, *Volume* and *Price*. The file can contain any synonym for domain and their corresponding attribute names.

This paper extracts the data from these files and organizes it domain wise where the content of first line specifies the domain. Then the various attributes of the corresponding domain are arranged in next line. Then the data corresponding to different attributes commences. The various extracted values of the attributes from files are organized corresponding to their attribute names and are mapped onto the document frequency and the Inverted Index is implemented for the values of these attributes which is mapped upon the respective docIDs which contains that attribute value.

As a view of hierarchy, the Root of the tree represents the Keyword Domain. The various domain names are represented as internal nodes and as the child of this root node, which is the second level. The next level of the tree represents the various attribute names domain wise and the values of these attribute names are arranged at the next level. Then an inverted index is created for these values where the various

values constitutes the Dictionary and each term of Dictionary has pointed to the Document Frequency which defines the number of documents containing this term and a pointer to the respective Document IDs. The hierarchy of this project is shown in Fig 4.

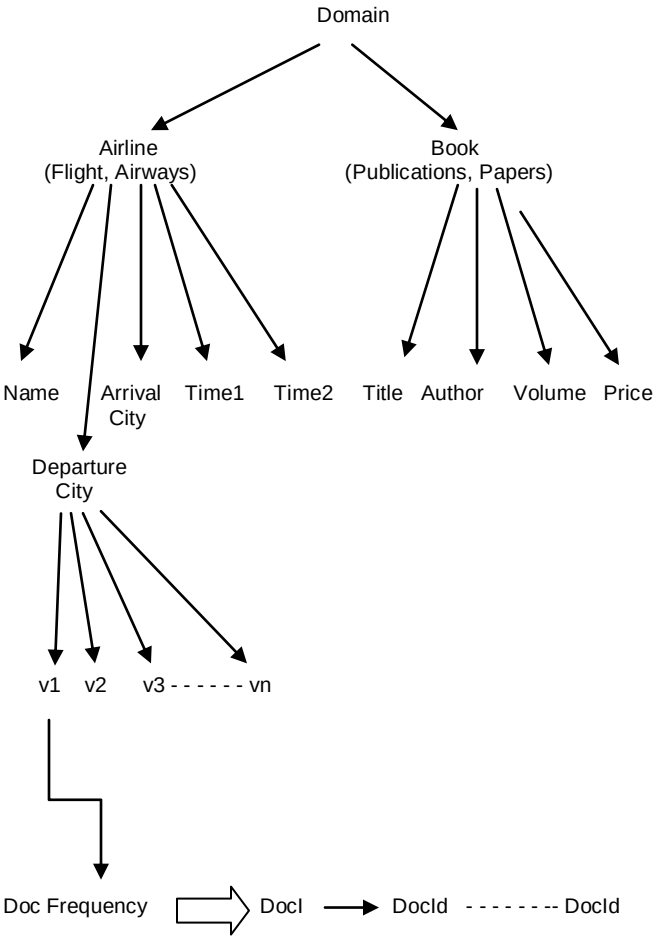


Fig 4. Index Structure for Airline and Book domain

The goal of this paper is to extract the data from various hidden web databases and this data in integrated form will be stored in large repository. 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 this interface. These queries fetch the data from hidden sources and send it back to user with desired results.

Type of Queries:

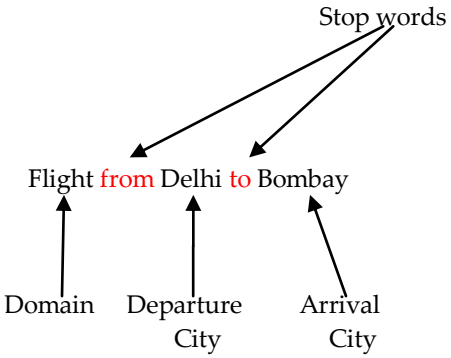
There can be various types of queries that the user enters in searchQueryForm and some of them are described below:

Type 1: Query containing the domain name and the values of the attributes.

If User enters a Query containing the domain name and the values of the attributes then the query processor processes it in the following steps:

1. The Query processor will tokenize the query into tokens which includes the rejection of all the stop words.
2. Then the first token of the query is used to identify the domain and the remaining terms are searched in the values of the attribute names of that particular domain.
3. If the tokens match then the respective posting list are retrieved and intersected to return the docIDs in result.
4. The resultant docIDs represents the files that contain the data corresponding to the given query.

Example 1. Query: *"Flight from Delhi to Bombay"*

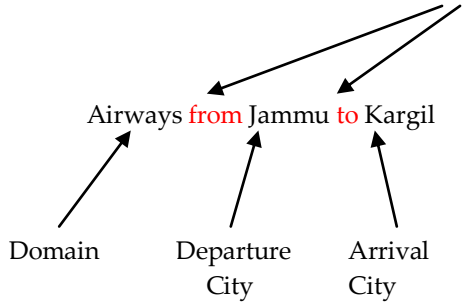


Steps:

1. The Query processor will tokenize the query into tokens neglecting all the stop words and the first token of the query will identify the Airline domain.
2. The second term will be searched in the values of the Departure City attribute.
3. If it does not match then a blank Posting List is returned corresponding to given Departure City.
4. If it matches then the respective posting list is returned.
5. Likewise the third term will be searched in the values of the Arrival City attribute.
6. If it does not match then a blank Posting List is returned corresponding to given Arrival City.
7. If it matches then the respective posting list is returned.
8. Finally, both the Posting Lists are intersected to return the docIDs of the files that contain a Flight for the given Departure City and Arrival City in Query as a result.

Example 2. Query: *"Airways from Jammu to Kargil"*

Stop words



Now, since Airways and Flight are synonyms of each other. Writing the query as Airways from Jammu to Kargil also runs in the same way.

Type 2: Query containing a single term

If User enters a Single Query Term in the query field then the query processor processes it in the following steps:

1. It first analyses the Query term to check whether it is some Domain name or not.
2. If yes, then it returns the various attribute names of that domain, attribute values and their corresponding postings lists.
3. If not, then query processor will check it against the various attribute names of all the domains to find whether it matches with any of them or not.
4. If it matches any of the attribute names then the various values corresponding to that attribute name and their respective posting lists are returned to user as result.
5. If it does not match then the query processor will check it against all the values of all the attribute names regardless of domain.
6. If it matches with any of the value of attribute name then the query processor will return the corresponding posting list of that particular value else it will return NULL.

Example 1. Query: *"Flight"*
The above query will return the list of all the flights along with the document frequency and the corresponding postings list.

Example 2. Query: *"book"*
The above query having a single term "book" returns the list of all the books along with their document frequency and the corresponding postings list.

Example 3. Query: *"name"*
The above query does not match with any of the domain; the query processor will check it against the various attribute names irrespective of the domain. Then all the flight names are returned along with their document frequency and the corresponding postings list.

Example 4. Query: *"title"*
The above query also does not match with any of the domain so the query processor will check it against the various attribute names irrespective of the domain. Then the titles of all the books are returned along with their document frequency and the corresponding postings list.

Example 5. Query: *"author"*
Since the above query does not match with any of the domain so the query processor will check it against the various attribute names irrespective of the domain. Then all the author names are returned along with their document frequency and the corresponding postings list.

Example 6. Query: *"Indigo"*
Since the above query term does not match with any of the domain, even not with any of the attribute name, so it will be searched among all the attribute values. As it matches with one of the flight name so it will return the document frequency of "Indigo" along with its postings list.

Example 7. Query: *"OS"*
The above query term is matched with the various domains, as it does not match with any of them then it is matched with all the attribute names. As no attribute with this name exists so it will be searched among all the attribute values. As it matches with one of the book title so it will return the document frequency of "OS" along with its postings list.
Example 8. Query: *"Navathe"*
Since the above query contains only one term which is neither any domain nor any attribute name but a value of Author attribute. So, it will return the document frequency of "Navathe" along with its postings list.

Type 3: Blank Query

If User does not enter a Single Query Term in the query field then it will not extract anything in the result but a message will be prompted to say "Please enter the Query".

4 CONCLUSION

The hidden web is not really hidden, but because searchable databases are not indexable or queryable by todays search engines, so they appear hidden to the average Internet user. In this paper information retrieval architecture for hidden web

has been proposed that can help users to find the desired information in integrated form. A Data Repository with inverted Index is formed for the data extracted from various files corresponding to different domains. In addition to this, duplicate index was also removed from repository and this repository is prepared for user search. In this Paper, when user fills the Search Query Interface form for searching, these queries are processed by the Query Processor to fetch the desired data and return it back to the user with desired results. The Query Processor extracts the Query String from the Query Interface and then it tokenizes this string into a number of tokens. It then identifies the Domain of the query by analyzing its first token and matches the remaining tokens with the various attribute values stored in the Data Repository to retrieve the postings lists which are then intersected to return the result page to the user. Hence, the user's effort is minimized by just entering a query into search query interface to get the desired data.

Building a User-Friendly functionality is an evolution process. There are always features and functionalities that can be improved, to provide users with more ease of use and better output. Besides improving the effectiveness of the current solutions, the future work can be the study of how to transfer the techniques implemented here to other contexts, such as mining the extensive bioinformatics literature to help match schemas of data sources in that domain, and mining text documents that accompany real-world database schemas for further metadata information. This project can also be expanded through adding multiple domain and their corresponding sub domains. Along with modification in implementation, new algorithms for better understanding and quick results of multiphase dynamic queries can be introduced.

REFERENCES

[1] http://en.wikipedia.org/wiki/Invisible_Web.

[2] Anuradha, A.K.Sharma, *"A Novel Technique for Data Extraction from Hidden Web Databases"*. International Journal of Computer Applications (0975 – 8887), Volume 15– No.4, February 2011.

[3] Komal Kumar Bhatia, A.K.Sharma, "*A Framework for an Extensible Domain-specific Hidden Web Crawler (DSHWC)"*, communicated to IEEE TKDE Journal Dec 2008.

[4] Anuradha, A.K.Sharma, *"Hidden Web Data Extraction Using Dynamic Rule Generation"*. International Journal on Computer Science and Engineering (IJCSE).

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

[6] 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.

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

[8] Brian E. Brewington and George Cybenko. "*How dynamic is the web*." In Proceedings of the Ninth International World-Wide Web Conference, Amsterdam, Netherlands, May 2000.