INDEXING FOR DOMAIN SPECIFIC HIDDEN WEB

Sudhakar Ranjan, Komal Kumar Bhatia

1Department of Computer Science Engineering, Apeejay Stya University
2Department of Computer Engineering, YMCA University of Science & Technology Faridabad, India

ABSTRACT:
The conventional search engine work to crawl, index and query the “Surface web”. A large amount of data is hidden in the databases behind the search interfaces referred to as “Hidden web”, which needs to be indexed so in order to serve user’s query. Hidden Web’s broad and relevant coverage of dynamic and high quality contents requires an efficient technique for indexing. Efficient index structures need to build to maintain the Hidden web content, which are then accessed to provide relevant answers to many types of user’s query.

Keywords: WWW, Hidden Web, Surface Web, Search Engine, Crawler.

[1] INTRODUCTION

WWW can be broadly classified into: Surface Web and Deep Web. The Surface Web (also known as the Visible Web or Publicly indexable Web) is that portion of the WWW Crawled by general purpose crawler and thus indexed by Search Engine. The Hidden Web (also called the Invisible Web or the Deep Web) refers to WWW content that is not part of the Surface Web and is not indexable by the conventional search engines. Traditional search engines cannot retrieve content from the Deep Web as the content on the Hidden web is hidden behind the search interfaces that the user needs to fill in and as the result of filling the required details on the search interfaces, result pages are generated. These results pages are generated dynamically depending upon the values entered by the user on the search interface. The result pages are hidden behind the search interfaces and are not crawlable by the conventional search engine. For example, if a user wants to search information about some book, then in order to get the required information, he/she must go to the particular site and fill the details in a search form available on the site. As a result he/she gets the desired information. The Hidden Web is particularly important, as organizations with large amounts of information are gradually making their databases available online. It is often the case that a user can obtain the content of these databases only through dynamically generated pages, delivered in response to web queries.

UUIC survey indicates that there are more than 300,000 Hidden Web databases and 450,000 query interfaces available on the Web. The contents on Hidden Web are not only increasing but also spanning well across all topics.
Indexing For Domain Specific Hidden Web

The main characteristics of the Hidden Web are as follows:
- It has Broad, relevant Coverage,
- It contains High quality contents, and
- Its contents exceed all printed contents.

However, crawling Hidden Web is a Challenging task. In order to crawl the Hidden Web contents from the WWW, the crawler should be able to download the search interfaces to automatically fill and submit them to get the Hidden Web pages as shown in Fig. 1.1.

The DSHWC downloads the Hidden web documents and stores them in the repository. After this some index function needs to be applied over this repository in order to index the Hidden web contents. Moreover, simple indexing technique cannot be applied to index the Hidden web contents due to the following reasons:

![Fig. 1.1 Crawler Search Interface Interaction](image)

The Hidden web content existing in the databases hidden behind the search interfaces is structured in nature as compared to the content over the Surface web. So, the index structures need to deal with the structured content existing over the Hidden web.

The Hidden web has broad coverage and huge amount of data existing over the Hidden web, approximately 400-500 times larger that of the Surface web. Thus, an efficient and optimized indexing technique is required for the fast retrieval of documents.

In this paper, a structure has been proposed to index the Hidden web contents. The index is then accessed to answer the user’s query. The paper has been organized as follows: section 2 describes the current research that has been carried out in this area; section 3 describes the proposed work to index the hidden web documents, which provide the users with the relevant information, section 4 shows the performance of proposed work and last section concludes the proposed work.

[2] RELATED WORK

The notion of a Hidden (or Deep or Invisible) Web has been a subject of interest for a number of years. A number of researchers have discussed the problems of crawling the contents of Hidden Web databases. In [5] Raghavan and Garcia-Molina proposed HiWE, a task-specific
hidden-Web crawler. The main focus of this work was to learn Hidden-Web query interfaces. Xiang Peisu et al. [13] 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 various strategies for building (domain, list of values) pairs. Although this work extracts some part of Hidden Web but it is neither fully automated nor scalable. In [12],Bhatia et al. proposed a Domain–Specific Hidden Web Crawler (DSHWC) that automated the process of downloading search interfaces, finding the semantic mappings, merging them and filling the Unified Search Interface (USI) produced thereof, which is then finally submitted to obtain the response pages from Hidden Web. The DSHWC automated the process of searching, viewing, filling in and submitting the search forms followed by the analysis of the response pages.

Thus, there is a need for introducing some efficient mechanism for the purpose of indexing Hidden Web, thus serving the user’s request.

[3] PROPOSED WORK

The general architecture of a search engine for the Hidden web is shown in Fig. 3.1. The figure shows that DSHWC crawls the Hidden web for downloading the Hidden web documents. The detailed process used by the crawler for the purpose of crawling has been discussed. The retrieved Hidden web documents are thereof stored in a repository. The proposed Indexing function is performed by the Indexer module of the Search engine. User provides a query on the query interface of the Search engine, the index is then searched for finding out a corresponding match, if any and the results are returned to the user. The paper focuses on the Indexer module and introduces an efficient index structure for the same.

![Fig. 3.1 Architecture of a Search Engine for Hidden Web](image)

The paper assumes that the content over the Hidden web is categorized among several domains like book, airline, railway, movie, auction etc. Since, the information about domains is hidden behind the interfaces; the contents in the databases are accessible through the search interface. The search interface contains various fields related to the domain that the DSHWC
Indexing For Domain Specific Hidden Web

needs to fill in, in order to crawl the Hidden Web databases. Also, there are a large number of websites related to a particular domain and each web site provides a search interface to the user, asking to provide details for the attributes available on the search interfaces.

As available in the literature [12], the crawling task of DSHWC has been divided into five phases. Phase1 is concerned with the automatic downloading of the search interfaces. Phase2 employs Domain-specific Interface Mapper [15] that automatically identifies the semantic relationships between attributes of different search interfaces. Using these semantic mappings, the interfaces are then merged to form a Unified Search Interface (USI), which is then filled automatically. The filled USI is then submitted to the Web and the response pages thus generated are collected and analyzed i.e. single unified search interface is presented to the crawler and upon submission of a query via the interface, equivalent queries are submitted to many hidden web databases via front-end query interfaces and then the results are extracted from different web-sources.

The advantage of this, for example, in the “airline” domain, is to prevent querying from each airline site among large airline websites which is time consuming; the second advantage is that this integration of several interfaces into a single interface presents a simple and easy query interface that need to be filled. Then data is then downloaded from several hidden airline databases.

![Fig. 3.2 Tree like structure showing organization of data among domains](image)

The general structure of the index has been shown in Fig. 3.2. At the higher level the index consists of all the domains at the same level. Each domain contains the most common attributes of that particular domain. It may be noted that in Fig. 3.2, books, airline, movies etc. are different domains and ISBN, rate, author, title etc. are the attributes within the book domain. The next level successors contain the values of different attributes within a particular domain. The values of the attributes that looks similar are clustered, like C2, C2, C3 are different clusters containing values (B1, B6), (B2, B5) and (B3, B4) respectively.

![Fig. 3.3 Unified Search interface for “Book” domain](image)
The attribute names at the third level has been chosen from the attributes of the Unified Search interface as they has been taken as the basis for the indexing of hidden web documents for a particular domain and that is why this type of indexing is named as attribute based indexing. For example, the Unified Search interface (shown in Fig. 3.3) of Book domain in DSHWC contains Author, ISBN, Title, Year and Rate as attributes or fields and the index structure for the same is also containing the same fields. To make the understanding of the proposed concept, the Book domain has been taken as an example. For convenience, different domains taken in the index have been encoded as shown in Fig. 3.4(i) and their corresponding attributes has also been encoded as shown in Fig. 3.4(ii). It may be noted from Fig. 3.4(i) and Fig. 3.4(ii) that Book domain has been assigned a domain identifier (domID) as D1 and attributes of D1 are assigned the attribute identifier (attrID) as A1, A2… A5. Now for each attribute there exists large number of values, say for A1, we have B1, B2….B6 values corresponding to A1 and these values exists in different documents D1, D2……Dn. The values of attributes are then clustered based upon some similarity measure. In our case, B1, B6 lies in cluster C1; B2, B5 lies in cluster C2 and B3, B4 lies in cluster C3. It may be noted here that by doing this we have arranged the documents containing different values in different clusters i.e. now C1, C2, C3 contains (d1,d6,d7,d8), (d2,d3,d4,d6) and (d1,d4,d5) respectively. Clustered index structure has been shown in Fig. 3.5.
Indexing For Domain Specific Hidden Web

The algorithm for index construction has been shown in Fig. 3.6.

Algorithm index_construction() 
{
Step1: Classify the documents according to the domains.

2. Assign domID to each domain.
3. Identify the attributes of different domains from Unified search interface of that domain and assign attrID to each attribute in the index structure.
4. Create the clusters of documents depending upon the value of each attribute.
}

Fig. 3.6 Algorithm for index construction

The algorithm provides the stepwise process for index construction. The overall index structure has been shown in Fig. 3.8. The user’s query needs to be answered by the query module of the search engine. Corresponding to the user’s query, the query module at first identifies the appropriate domain. Then the list of attributes belonging to the domain is looked for. For all the values of each attribute, the index structure is searched to answer the user’s query. The algorithm for searching the index is given in Fig. 3.7.

Algorithm Index_search() 
{
Read a user’s query // Identify the domain and its domain ID
dom=domain
domid=domain ID // Identify the attributes and their values
attr[]=attributes for each attribute in attr [] // retrieve the attribute I
attrID[]=attribute ID // retrieve the value
val[]=value
for each attributeID in attrID[ ] // LOOK UP INDEX
for value in val [ ] // retrieve the clusterID
cID[]=cluster ID
for each cID in cID[ ] // retrieve the data sources
list_of_ds=data sources
return list_of_ds
}

Fig.3.7. Algorithm for Index search

After constructing the index, the next step is to search the required item in the index. This section discusses the same by taking the same example of book domain and takes a small index shown in Fig. 3.5.
Let us suppose the user issues the following queries one by one:


The above query contains some stopwords and redundant information. Thus there is a need to optimize the query. “Retrieve”, “with” and “as” are treated as stopwords and thus they are removed. The query now contains the terms “books”, “ISBN” and “B2”. The first step is to identify the domain out of these terms. The keyword “books” identifies that the query is related to “book” domain. Now a match is found for B2 is found in the index and a set of documents corresponding to the cluster of “B2” will be returned as the result.

As per the algorithm (in Fig.3.7),

- dom=“Book”, (obtained by semantically mapping the identified domain with the listed domain names in the field “domain” in Fig. 3.4(i)).
- domID=“D1”, (using Fig. 3.4(ii))
- attr[]=“ISBN”, (obtained by semantically mapping the identified attributes with the listed attribute names in the field “attribute” in Fig. 3.4(ii)).
- attrID[]=“A1”, (using Fig. 3.4(ii))
- val[]=“B2”, (using Fig. 3.5)
- cID[]=“C2”, (using Fig. 3.5)

For cID=“C2”:
- list_of_ds=d2, d3, d4, d6 (using Fig. 3.5)

The list of data sources obtained above indicates the Hidden Web documents that have already been crawled and stored in Hidden Web repository. The index structure as discussed above is maintained. On issue of a user’s query say, to search for books with ISBN is “B2”, the index maintained would be searched to give a list of Hidden Web documents i.e. d2,d3,d4,d6. The function of index lookup is performed by the indexer module.


After the elimination of stopwords and redundant information, the remaining terms are “books”, “ISBN”, “B2”, “title” and “C programming”’. The identified domain is “book”. The
keywords “C” and “programming” are first ANDed to simply the query term. Now, a match is found for “B2” and “C programming” in the indexes maintained and the set of documents (containing both the query terms) corresponding to the cluster will be returned as the result.

As per the algorithm (in Fig.3.7),

\begin{itemize}
\item \text{dom=Book}, \quad \text{(obtained by semantically mapping the identified domain with the listed domain names in the field “domain” of Fig. 3.4(i))}
\item \text{domID=“D1”, \quad (using Fig. 3.4(i))}
\item \text{attr[]=\{“ISBN”,”title”\}, \quad (obtained by semantically mapping identified attributes with the listed attribute names in the field “attribute” of Fig. 3.4(ii))}
\item \text{attrID[]=\{“A1”,”A2”\}, \quad (using of Fig. 3.4(ii))}
\item \text{val[]=\{“B2”,”C  programming”\},}
\item \text{cID[]=\{“C2”,”C1”\}, \quad (using Fig. 3.5)}
\item \text{For cID=“C2”:}
\item \text{list_of_ds=d2, d3, d4, d6. \quad (using Fig. 3.5)}
\item \text{For cID=“C1”:}
\item \text{list_of_ds=d1, d2, d3. \quad (using Fig. 3.5)}
\end{itemize}

The list of data sources obtained above indicates the Hidden Web documents that have already been crawled and stored in Hidden Web repository. The index structure as discussed above is maintained. On issue of a user’s query say, to search for books with ISBN is "B2" and title is "C programming", the queries will be fired as two separate queries on the attributes ISBN and title i.e. search for books with ISBN is “B2” and search for books with title “C programming”. Index maintained would be searched.

At first, the index is searched for the information on books where ISBN is “B2” to give a list of Hidden Web documents i.e. d2, d3, d4, d6. Then, the index would be searched for obtaining information on books where title is “C programming”. It gives a list of documents i.e. d1, d2, d3. Since, the user requests for the information on books that satisfy both the conditions, the resultant list of documents would be AND to give the final result of documents i.e. d2, d3. The function of index lookup is performed by the indexer module.

One of the major issues involved is the identification of the “domain” and “attributes” belonging to the domain. Like, for the user’s query “Retrieve books with ISBN as ‘B2’, the domain is “book” and attributes= \{“ISBN”\}. Similarly, for the query “Retrieve books with ISBN as ’B2’ and title as ‘C programming’, the domain is”book” and attributes is \{“ISBN”,”title”\}.

On issue of query, at first stop listing and stemming is performed. Then a search is performed for the nouns and noun phrases. Considering the first query, the identified nouns are “book”, “ISBN” and “B2”. In Fig. 3.7, for identification of the domain, a search is performed in the domain field. Searching for “ISBN” and “B2” would be failed in the table. However, a match is obtained for “book”. Corresponding to the identified domain, the domID is retrieved. Using the pointer, a set of values corresponding to the attribute is obtained. For each value, a cluster is specified, containing a list of data sources. The fetching of the list thus satisfies the user’s request.
[4] CONCLUSION

This paper has proposed, Indexing function is performed by the Indexer module of the Search engine. User provides a query on the query interface of the Search engine, the index is then searched for finding out a corresponding match, if any and the results are returned to the user. The paper focuses on the Index construction algorithm and Index search algorithm and introduces an efficient index structure.
Indexing For Domain Specific Hidden Web

REFERENCES

[4] Lovekesh Kumar Desa: A Distributed Approach to Crawl Domain Specific Hidden Web

Sudhakar Ranjan and Komal Kumar Bhatia

68