A Bootstrapping Approach to classification of Deep web Query Interfaces

G. Pavai¹, Dr. T. V. Geetha²
1Anna University, Chennai, India, Email: pavai_gops@yahoo.co.in
2Anna University, Chennai, India, Email: tv_g@hotmail.com

Abstract— Classification of Deep web sources is a very important process for the data extraction process while accessing the deep web content since it deals with domain specific data only. The existing methods cannot effectively classify these web databases. Hence, to solve this problem, we propose a new framework that uses the bootstrapping approach for automatic and accurate classification of the query interfaces of web databases. Our system was able to classify the deep web sources with a better average precision when compared to the existing systems. Also, the classified results used for surfacing the deep web content gave good results equivalent to the results obtained when manually classified query interfaces were used.

Index Terms— Deep web, surfacing, query interface, forms, data extraction, bootstrapping

I. INTRODUCTION AND MOTIVATION

To discover content on the Web, search engines use web crawlers that follow hyperlinks through known protocol virtual port numbers. This technique is ideal for discovering resources on the surface web but is often ineffective at finding deep Web resources. For example, these crawlers do not attempt to find dynamic pages that are the result of database queries due to the infinite number of queries that are possible. It has been noted that this can be (to a certain extent) overcome by providing links to query results, but this could unintentionally inflate the popularity for a member of the deep Web. Research has proved that over 95% of the web data is hidden in the web databases which are in no way accessible to the surface web user. Normally, a person gets access to these data via filling details in the query interfaces (forms) and submitting them to the web server. The server then sends queries to the web databases and retrieves results as per the details filled in by the user in the form. Such result pages are normally having dynamic URLs and these dynamic URLs cannot be indexed by the search engines. So we lack access to such abundant data because it is hidden in the deep web. Currently, researchers are working on bringing out the deep web data for the search engine users.

There are two methodologies for accessing the deep web data. One is, the user gives a query, and accordingly, the data from the query is used to fill a relevant query interface and it is posted to the web server. The web server accesses the data in the web database by sending a query to the web database. The result page of the database access is showed as the required result page. The other methodology is surfacing the hidden web data. i.e., we retrieve all possible data from a given web database by giving queries to the web database and extracting the data from the result pages. The data thus extracted is checked for duplicates and the data is written to an html file thereby giving the result page a static URL. This page now behaves just like any other web page i.e. can be indexed. So, now that the page is brought into the index, it is a searchable

DOI: 01.IJRTET.11.1.1307
© Association of Computer Electronics and Electrical Engineers, 2014
document. We follow this approach since the former consumes online search time. Whereas, in the surfacing of
the deep web content, since the data extraction is done offline, the search time does not increase.
In our earlier paper [9] we addressed the architecture for surfacing the contents of deep web databases. There
we had constructed a unified query interface for the tourism-airfare domain and used that to surface the deep
web contents. We manually selected the query interfaces pertaining to air fares and proceeded with the
process. However, it is a tedious process to classify the domain of a deep web query interface when it is done
manually. So, we wanted to automate this process and this forms the motivation for our work. In this paper,
we propose a bootstrapping approach for the classification of deep web query interfaces.

II. BACKGROUND

We describe the work relating to deep web first and then about the existing work in classifying the deep web
query interfaces followed by the literature for classification using the bootstrapping algorithm.

Bin He et al. [6] discusses about
i) Where to find the required domain specific query interfaces and to what depth should the crawl be
 performed to get the query interfaces?
iii) Deep web is structured or not.
iv) The subjects contained in the Deep web databases.
v) How far popular search engines handle Deep web data?
vi) The coverage of Deep web directories.

Ying Wang et al. [8] discusses how to discover domain specific Deep web sources by using a focussed
crawling methodology that locates web sites for domain specific data sources and then judges whether the
corresponding Deep web query interface exists within three stages of crawl and then checking whether the
query interface is relevant to a particular topic. While this paper deals with how domain specific Deep web
query interfaces are found out, Hui Li et al. [7] discuss an effective incremental web crawler that maintains
an up to date local repository of web pages by calculating priorities for all the web pages.

Bin He et al. [4] discuss how web query interfaces are integrated using a three step process, where step 1 is
interface extraction process where the attributes in the query interface are automatically extracted, step 2 is
schema matching where semantic correspondences among attributes are found out and step 3 is interface
unification where we construct a unified query interface based on the matches we find in step 2. The same
process of web query unification is being dealt with in a different method in Pierre et al. [5] where domain
knowledge is used to unify the query interfaces. In our work, we combine both the above methodologies to
form the unified query interface.

Pierre et al. [5] also discuss how to extract the valuable data from Deep web databases by probing the forms
with all types of values and analysing all the result pages to obtain all the fields of the Deep web database.
Then many such result pages are analysed and maximum amount of data is extracted from the web databases
by probing the query interfaces with all possible values to all the fields. In our work, we have used the auto-
filling field's data that is available in the forms to fill the query interfaces in order to extract data from the
deep web databases.

Jufeng et al. [1] proposes a technique for data extraction from hidden web pages using rules of structure,
logic and application whereas, Anuradha and A.K. Sharma. [2] proposes a technique for data extraction from
hidden web databases by identifying the templates and tag structures of a document by sending queries
through HTML query interfaces of the Databases. But, no single database will provide all desired data. This
gives rise to F. Wang et al. [3] that takes a further step by extracting data from multiple databases. Some
issues handled in their work are the order in which the multiple Databases are queried before the others and
the non-availability of certain databases at times due to network and hardware problems that is handled in
this work by proper query planning to design alternative plans when optimal plans fail. Our work handles the
multiple database problem by just removing the duplicates. We do not handle the query plans part.

Zhou et al. [14] describe a Naïve Bayes classifier for classifying the deep web query interfaces based on the
features in them. Though this work is giving good results, it suffers from two defects.

i) Being a probabilistic method, this one demands the prior probabilities for classification which in
turn demands that all examples (query interfaces) should be available initially itself. However,
this can be done only when the entire crawl process is over.

ii) When a new set of examples have to be classified, this method starts from scratch rather than
classifying incrementally.

From the above literature survey, we get the following research gap in the area of domain classification of deep web query interfaces. We need a classification method

(i) That is less time consuming
(ii) That classifies sources as and when we crawl.
(iii) That uses very few training examples.
(iv) That continuously learns from new examples.
(v) That uses content rather than heuristics.

We go in for the bootstrapping methodology for classification of deep web query interfaces because,

i) It is less time consuming.
ii) It classifies sources as and when we crawl and does not wait for the entire crawl process to be over.
iii) It starts initially with very few seed patterns (examples).
iv) It incrementally learns new patterns from new examples and learning here is a continuous process.
v) It uses content of the deep web query interfaces rather than the heuristics about the structure and content of the web pages.

Thus, we have used the bootstrapping method which is a semi-supervised learning method to classify the deep web query interfaces.

III. METHODOLOGY

The bootstrapping algorithm used for classification is described first followed by using these classified deep web query interfaces to construct the unified query interfaces for the respective domains and thereby surfacing the deep web content.

A. Bootstrapping approach to classifying the deep web query interfaces

The methodology of classifying the deep web query interfaces using the bootstrapping algorithm is described clearly in the fig 1.

We have taken the Metaquerier project's deep web data set for 4 domains namely air fare, book, cars and the job domains. We represent each query interface as a set of strings denoting the various fields in the query interfaces. The Initial seed patterns are given by manually selecting the features that best help to distinguish a domain from the others. We begin with one representative pattern for each domain that we want to classify. For example, for the air fare domain, we used the initial seed pattern as \{from, to\} denoting the source and destination of the journey. We use these patterns to classify the query interfaces in the input set.

The Bootstrap classifier module given in Fig 1, does the classification of the deep web sources using the initial seed patterns. Every input vector representing a deep web source is compared to the seed patterns one by one. In case the vector matches with any of the seed patterns, then the deep web source represented by the vector is classified belonging to the domain represented by the seed pattern that the vector had a match. Some of the query interfaces will be classified in the first iteration itself by their similarity to the existing seed patterns.

New patterns are learned with the classified query interfaces and the patterns are added to the already existing seed patterns. Then the classified query interfaces are removed from the input set. The new pattern is nothing but the vector represented by the query interface.

The matching that we saw so far is the exact matching scheme where the words in the pattern appear exactly the same way in the vector as well. However, this kind of matching may be a success with only a very few of the query interfaces. Those query interfaces that are not classified in the above process are subjected to a
partial matching scheme where the exact word match is not required but only a match with a semantically equivalent word is required. This is done with the help of domain knowledge. Domain knowledge helps to resolve problems of the sort where one website may call the source of journey as “from” and another as “start”. The vectors that are classified by the partial matching scheme are also removed from the input list and new patterns are extracted and are added to the seed patterns list.

![Diagram](image)

Fig 1. Bootstrapping Algorithm for Deep web query interface classification.

There may still be many more inputs that stay unclassified by the end of this two-step process. This reduced input set becomes the input set for the next iteration. The above process is repeated till either no more query interfaces remain in the input set that is all the inputs have already been classified or the existing patterns are not enough to classify the remaining query interfaces in the input set. This situation can be deduced if there is no change in the input set for two consecutive iterations i.e., if nothing new can be learnt in one iteration then the same is true for all the following iterations too.

Thus we obtain a well classified list of the query interfaces which we use in the construction of the unified query interfaces for the respective domains in the next step so as to extract and surface the deep web content. The main advantage of using the bootstrap classifier is that it can incrementally learn new patterns and once the patterns are learnt, they can be used to classify new deep web sources without having to restart the learning of the patterns phase again unlike the tf-idf based methods where every time a new deep web crawl is done, new index is formed for the tf-idf calculation. The added advantage of the bootstrap method is that incremental learning of patterns help in quicker classification time for the second and subsequent crawls if the seed patterns learnt during the previous crawl is used as the initial seed pattern list. The reason for this high reduction of time is mainly due to the fact that there are many patterns already learnt and there are high chances of a newly found deep web source is classified in the very first iteration itself because majority of the patterns are already present in the seed patterns list for the particular domain that the deep web source belongs to.

B. Surfacing the deep web content

Normally, we say that the contents of deep web databases are not directly visible to the surface search engine users. But, the query interfaces, that are indexed to the search engine are displayed in the results for a user query based on the relevance of the form to the query. If the user happens to click on this link and fill in the form and submit it to the web server, the results of the query to the database may be displayed. This is one way of getting the deep web data using a surface search engine. But it depends on the probability of the user clicking on the query interface. This is an online process, so the time to get the results after giving the user gives a query is high. So, we go in for doing the filling in of forms and extraction of data from the result pages as an offline process and index the extracted results thereby reducing the time to get the desired results. Fig 2 gives a detailed description of the deep web architecture.

C. Unification of Query Interfaces

Finding out domain specific query interfaces is the first step in deep web search. Deep web search is a very specific search process and cannot be generalized. So, finding out only the domain specific query interfaces
is very important. This is a focused research topic in the Deep web domain. In our previous work [9], we had manually found out the query interfaces belonging to the tourism-airfare domain. In this paper, we have used the semi-supervised approach of the bootstrap classifier for classifying the domains of the deep web sources. For unifying these forms together to form a unified query interface, we have used domain knowledge [5] and the seed patterns that we have learned using the bootstrap classifier. We select the longest pattern vector for each domain as the fields for the unified query interface for that domain and construct the unified query interface.

![Deep web architecture](image)

Fig 2. Deep web architecture

The mapping of the unified query interface with that of the composite query interfaces is also done using the help of domain knowledge. This is different from the unification of the query interfaces suggested by our earlier paper where domain knowledge based query interface unification was done by incrementally adding all the fields present in all the component query interfaces [4] and not repeating similar fields.

![Composite query interface-1](image)

Fig 3. Composite query interface-1

Our unified query interface gives a common name for the same fields, with different names in the different query interface. Two sample query interfaces along with the unified query interface that we obtained for the Air fare domain is shown in Fig 3,4 and 5.

D. Automatic Form filling and Data Extraction

Once the unified query interface is ready, we automatically fill it in with the details from the input restricted fields of the composite query interfaces. We also use the heuristic of filling only the mandatory fields rather
than filling all the fields [2] along with our input restricted fields heuristics. One example could be the source cities and the destination cities in the forms i.e. only the names of Indian cities are allowed to be filled in the source and destination columns of the form in goindigo.com.

Fig 4. Composite query interface-2

Fig 5. The unified query interface

We have also considered the correlation among input fields of the query interfaces for the automatic form filling process that was not done in our previous work or to the best of our knowledge anywhere else. Here, we consider the correlation that may exist between some of the fields in the query interfaces. A perfect example for this could be the state and the city fields of the job search domain where filling in only the city (eg. Mumbai) automatically fills in the state (Maharashtra) and the country (India). This step vastly reduced the number of error pages in the result pages that were retrieved due to the mismatch of data filled in the correlated fields by the automatic form filler. However, this also reduced the network traffic by reducing the usage of the network for moving the error pages. This step was not followed in our previous work where we suffered with many error pages as results.

The field values are in turn filled into the composite query interfaces and the query is submitted to the respective web databases. The reverse mapping of the fields from the unified query interface to the composite query interfaces is also done with the help of domain knowledge. The result pages obtained by the above process are saved. Then we use the data extraction methodology where data in between tags <TD> and </TD> will be giving the required data [2]. The data so obtained is checked for any duplicates and are stored in a different file for analysis to be done. We analyze the results to get the results for certain queries that have the answers in the databases that we have considered. Some such queries are, 1) maximum airfare cost for an adult from place A to place B 2) minimum airfare cost for an adult from place A to place B 3) maximum airfare cost for a child from place A to place B 4) minimum airfare cost for a child from place A to place B
5) Name of the flight providing the cheapest service 6) number of tickets available from place A to place B on a given date. The results of the analysis are stored in the form of HTML files.

E. Indexing and Searching

The HTML files created as a result of deep web data extraction are indexed to the search engine just like any other crawled page is being indexed. Once the pages are indexed, we can go in for the search process. The important point to be noted here is that the present approach cannot answer any type of queries. Queries based on the pages that we have indexed alone are answerable. Once the query is given, search process is like the normal COREE search process and the retrieved results are displayed. We can use any content based ranking methodologies here. But we found that the link analysis based algorithms are not efficient in ranking the surfaced deep web pages since, these are a kind of stand-alone pages that do not have any links thereby giving the minimum rank to our pages that contain the data extracted from the deep web databases.

IV. EVALUATION

We have used the precision measure to calculate the effectiveness of our classification methodology.

\[
\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \quad \text{eqn (1)}
\]

The above equation describes how precision is calculated generally.

![Fig 6. Precision with and without correlation](image)

Precision of the result pages with and without using the correlation between the input fields given in the automatic filling of the query interfaces section is the first that uses the above equation. In this case true positives are nothing but the correct result pages and false positives are nothing but the error pages in the results. This is clearly shown in the Fig 6.

The results show that when correlation among the input fields was considered for the automatic form filling process, the number of error result pages were vastly reduced than what we obtained without using the correlation among the inputs for the automatic form filling purpose.

The fig 7 gives an account of the precision values of the query interfaces classified under the four domains given earlier.

The Fig 7 proves that our classifier works better than the tf-idf based classifier [13] with respect to all the considered domains. The proposed bootstrapping based classifier is therefore considered to be a better classifier than the modified tf-idf based classifier for the deep web source pages.

We have considered 35000 tourism specific documents along with the deep web content extracted to 73 HTML files. We gave the search engine 80 queries of which 60 were deep web specific queries belonging to the various domains specified earlier and the other queries were not related to the content in the databases that we have considered. Any search engine can be used for this purpose. But a Link analysis based algorithm...
will not be fruitful for ranking the surfaced Deep web content since these pages neither contain inlinks nor contain outlinks. So, content based ranking algorithms can be used.

Fig 8 gives the average precision for the deep web specific queries and other queries. Our experimental evaluation shows that the deep web specific queries, had an average precision of 0.93, 0.81, 0.83 and 0.85 for the airfare, book, job and the car domains respectively which are comparatively better values. The other queries which did not have any relevant information in the deep web databases that we considered for our work had to search for the answers in the index of the other crawled pages.

![Fig 8. Average Precision](image)

The average precision for these queries is only 0.56. Though the difference in the average precision values between these two types of queries stand as a strong evidence of the presence of more relevant data in the deep web databases, we should remember that not all the Deep web data can be surfaced and accessed via the surface search. If this has to be done all possible set of values should be filled in the unified query interface, the result pages saved, data extracted from them, integrate all such result pages after removing the duplicate entries, create HTML pages for the data and index them to the search engine. But this is a tedious process to do. This search that we considered so far is domain specific (tourism). There are several domains and several deep web databases for each domain which adds to the space and analysis time complexity of indexing all the deep web content to the surface search engine. Another important fact to be considered is that certain data in the Deep web databases keeps on changing instantaneously. For example, the number of available tickets in a flight on a specific trip could have been 20 when the data was extracted from the deep web database but it would have changed to just 2 within a few minutes. But the user gets the result as 20, till the next time the data extraction is done. This is a major drawback of this method. There is no use of indexing such results since by the time the data page is indexed to the search engine, it becomes outdated. Such data can be dealt with dynamically accessing the Deep web databases at the cost of time. We are currently working on an incremental crawl strategy that keeps track of the changes in the databases, so that the above said problem is being dealt with efficiently.
V. CONCLUSION AND FUTURE WORKS

In this work, we have proposed a bootstrapping approach for classifying the deep web query interfaces and have come up with good results and we were successfully able to classify a good percentage of the deep web query interfaces correctly. As future works, we would like to work on more efficient querying mechanisms to query the deep web so that more deep web data is available to the surface search engine users. We are currently working on a semi supervised approach to incrementally crawl the deep web pages that change quite frequently.

REFERENCES