Discovering the Deep Web through XML Schema Extraction

Yasser Saissi, Ahmed Zellou and Ali Idri
Mohammed V University in Rabat, Rabat, Morocco

Keywords: Deep Web, Schema Extraction, Web Integration.

Abstract: The web accessible by the search engines contains a vast amount of information. However, there is another part of the web called the deep web accessible only through its associated HTML forms, and containing much more information. The integration of the deep web content presents many challenges that are not fully addressed by the actual deep web access approaches. The integration of the deep web data requires knowing the schema describing each deep web source. This paper presents our approach to extract the XML schema describing a selected deep web source. The XML schema extracted will be used to integrate the associated deep web source into a mediation system. The principle of our approach is to apply a static and a dynamic analysis to the HTML forms giving access to the selected deep web source. We describe the algorithms of our approach and compare it to the other existing approaches.

1 INTRODUCTION

The indexed and accessible part of the web by the search engines is called in the literature the Surface web (Lyman, 2003) and it contains a vast amount of information. There is another part of the web that is neither indexed nor reachable by the search engines. This inaccessible part of the web is called in the literature the Deep web (Lyman, 2003) (Bergman, 2001) (He, 2007) (Chang, 2004). The deep web can be defined as the set of all the databases connected to the web that generates web results pages in response to user queries through their associated HTML forms.

For example, the URL www.careerbuilder.com is a deep web source URL giving access to a database of jobs that can be accessed only through the associated HTML form described in Figure 1.

One of the most important features of the deep web is its estimated data size 500 times larger than the surface web (Lyman, 2003) (He, 2007). However, its unique access through the HTML forms is the barrier that prevents the web search engines to access and to index its data.

Actually, we can identify in the literature three different approaches to access the deep web. The crawling and the surfacing approaches are two approaches used to extract all the content of the deep web sources. The extracted content can be used subsequently to respond to the end user query.

The third approach is the form integration approach that queries directly the deep web sources using their associated HTML forms.

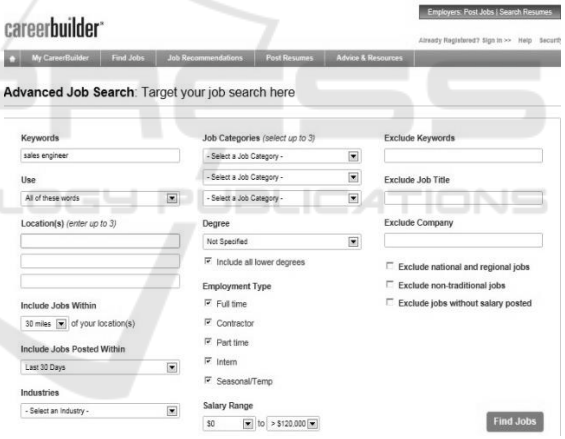

Figure 1: CareerBuilder Deep Web Source HTML form.

The deep web data need to be integrated in order to be accessible like the surface web. Our objective is to implement a web integration system. And to achieve this objective, we need to implement a mediation system that hides the particularities of each integrated source and provides to the end users an interface to access all the data sources in a unified way, hiding their autonomy, their heterogeneity, their location and their scalability (Zellou, 2008).

Therefore, the mediation system in the web environment tries to answer the user's query without asking each source separately to provide a unique

Saissi, Y., Zellou, A. and Idri, A.
Discovering the Deep Web through XML Schema Extraction.
DOI: 10.5220/0006013901410149
In *Proceedings of the 8th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2016) - Volume 1: KDIR*, pages 141-149
ISBN: 978-989-758-203-5

integrated response from all the partial responses obtained (Zellou, 2008) as described in Figure 2. But to build this mediation system, we need to know the schema describing the content of each source before its integration.

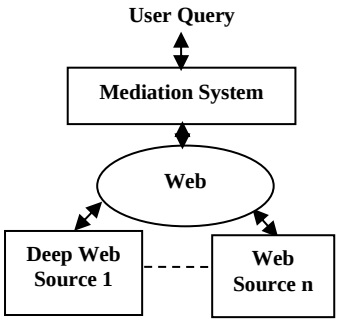

Figure 2: Mediation system overview.

In this paper, we propose our approach to extract the XML schema describing the hidden content of the analysed deep web source. The XML schema extracted is the information needed to integrate the associated deep web source into the mediation system.

To extract this XML schema, our approach identifies first the domain of the selected deep web source. The identification of the domain name allows us to use our private knowledge about this domain during the static and the dynamic analysis of the associated HTML forms. The static analysis of the HTML forms analyses all the aspects of their HTML source code to discover all the information about the data behind. And the dynamic analysis of these HTML forms queries all their fields to extract new information about the data that lie behind from the web pages results generated. During its process, our approach identifies also the constraints of the form fields. All the information extracted is used to build the XML schema describing the analysed deep web source.

In section two below, we describe the state-of-the-art of the deep web access approaches. In Section three, we present in more details our proposed approach with its algorithms and its implementation. And in section four, we conclude with our contribution and future works.

2 STATE-OF-THE-ART

In this section, we present the three existing approaches to access the deep web: the Crawling approach, the Surfacing approach and the Form Integration approach.

2.1 The Crawling Approach

At the beginning of the study of the deep web, the same strategy and principles used in the surface web was used to discover the deep web. Therefore, the first approach used was the Crawling approach (Khelghati, 2013) described in the part (a) of the Figure 3.

The Crawling approach extracts all the data of the selected deep web source and stores it in an external database. The principal limitation of this approach is that the data extracted lose all its semantics when it is stored out of its initial context. Another limitation or difficulty of this approach is the huge size of the deep web source data that needs a huge storage space.

This approach takes the URL of the selected deep web source as input and identifies its associated HTML forms. After, it queries these forms and extracts all the data from the web results pages generated by the queries. Then, it stores the extracted data in an external database that can be used after to respond to the end user queries.

The Liddle & Embley method (Liddle, 2003) is an example of the crawling approach implementation. This method starts to query the associated HTML forms of the selected deep web source with the possible values of each HTML form field and ends when no new data information are extracted from the web results pages generated.

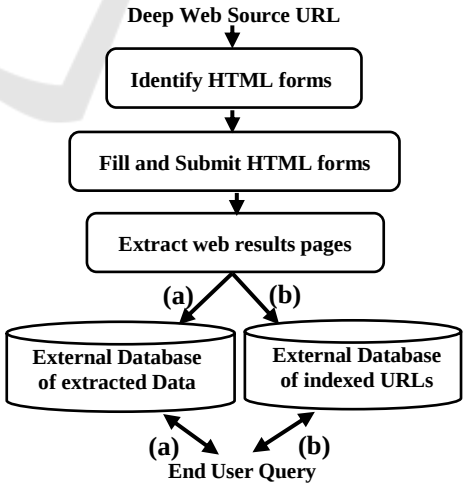

Figure 3: The Crawling and the Surfacing approach.

2.2 The Surfacing Approach

The second approach used to access the deep web is the Surfacing approach (Doan, 2012) described in the part (b) of the Figure 3. The Surfacing approach takes the URL of the selected deep web source as input and queries its HTML forms to generate web results pages. The URLs of these generated web results pages are indexed by the approach to be used after by a search engine to respond to the user queries.

The Google's Deep Web Crawl method (Madhavan, 2008) is an example of the surfacing approach implementation. This method identifies the form fields that generates relevant and different results and launch a first query using the values associated to the domain name. The results generated are analysed to identify the frequent words used after to generate new queries.

As described in the Figure 3, the surfacing approach is very similar to the crawling approach. To sum up, the crawling and the surfacing approaches were implemented to make the deep web accessible by the search engines. But the problem is that these two approaches try to adapt accessing the deep web content to the constraint and the nature of the used search engines in the web. And these two approaches do not match in the majority of the cases with the wealth and the structure of the information provided by the deep web.

Therefore, the most adapted way to access the deep web source is to query it directly by the end user. And the only way to do this is to use the associated HTML form. This is the principle of the following third approach to access the deep web: the Form integration approach.

2.3 The Form Integration Approach

The Form integration approach takes as input the URLs of a set of deep web sources. It identifies and extracts their HTML forms and analyses them to build a global form. This global form is able to address the end user queries to all the associated deep web sources through their access HTML forms.

This approach is more adapted to the deep web features because it access the deep web source through its HTML form. And it does not need to store all the information of the deep web source because it responds to the end user query directly from the information location.

The Form integration approach is implemented by many methods that we can classify under two types of methods. The first type builds a unique global form for a selected set of deep web sources to respond to all user queries. For this first type, the set of deep web sources is given as input to the approach. The second type builds a global form for a set of deep web sources to respond only to one user query. This second type discovers dynamically the set of deep web sources associated with each user query.

An example of the first type described of the form integration is the Wise method (He, 2005). This method extracts the schema describing each form of the selected deep web sources. The extracted schemes are used to generate one global query interface able to transmit all the end-user queries to the associated deep web sources automatically.

Another interesting method in the same family is the Ontology-based web Pattern Analysis with Logic (OPAL) method (Furche, 2011) (Furche, 2013). This method does not take in entry a set of deep web sources URL but only the domain name of interest. After, the OPAL method uses its domain ontology to build a global schema able to map each end user query to all the deep web source associated with the same domain.

The Metaquerier method (Chang, 2005) (Zhang, 2004) is an example of the second type described of the form integration. The Metaquerier is an automatic method that receives in entry the end user query. The method identifies dynamically all the deep web sources that can respond to this query. And it builds a global query interface able to respond only to this query from the identified deep web sources. The Metaquerier method has to generate a new global query interface for each new end user query.

2.4 Comparison of the Deep Web Access Approaches

As described in Table 1, we compare these three approaches and we find that the crawling and the surfacing approaches have two limitations. Firstly, these two approaches do not know if all the existing data are extracted. Secondly, the storage of their results is done out of the initial data context, and this can affect the original data semantics. On the other side, the form integration approach is more adapted to the data extraction in the deep web environment. But this approach focuses more on the forms than on the data behind.

Our approach described in the following section uses the analysis of the HTML forms giving access to the selected deep web source in order to extract a schema describing the data behind the form with the identification of its nature and its constraint and not only a schema describing the form itself.

Table 1: Comparison of the deep web sources access approaches.

Approaches	Results	Limits
Surfacing	Indexed URLs	Indefinite source coverage Weakened semantics
Crawling	Extracted data	
Form Integration	Global Query Interface	Form Oriented

3 OUR APPROACH

3.1 General Description

In this section we present our automatic approach to extract the XML schema describing a selected deep web source (Saissi, 2014; Saissi, 2015).

The objective of our approach is to extract from each deep web source an XML schema describing its content so that it can be possible to integrate all the deep web sources in a mediation system in order to access the deep web information easily.

Our approach benefits from all the features of the deep web to maximise the precision and the exhaustiveness of the XML schema extracted. In addition to all the deep web features previously described, our approach uses in its process the following characteristics of the deep web domain names. The studies (He, 2007) (Chang, 2004) show that the number of the different data domains in the deep web is less than twenty. The most frequent domains in the deep web are: Government, Health, Arts, Humanities, Politics, Lifestyles, News, Media, Society, People, Sports, Education, Science, Business, Travel, Shopping, Jobs and Library. Our approach uses this feature to define and use a knowledge database about each deep web domain where we can store, use and enrich our accumulated knowledge about each domain.

Our approach uses also the following features of the HTML forms. The form uses a vocabulary easy to understand (Bing, 2007) (He, 2005) and gives additional information describing its fields (Wang, 2003) such as validation code.

As described in Figure 4, our approach has three main steps: the domain identification step, the static analysis or the form processing step, and the dynamic analysis or the form querying step.

The domain identification step identifies the domain name of the analysed deep web source in order to be able to use our knowledge database about the associated domain name during the approach process. Our approach uses also a thesaurus, if required, to identify a new discovered keyword and a new domain name.

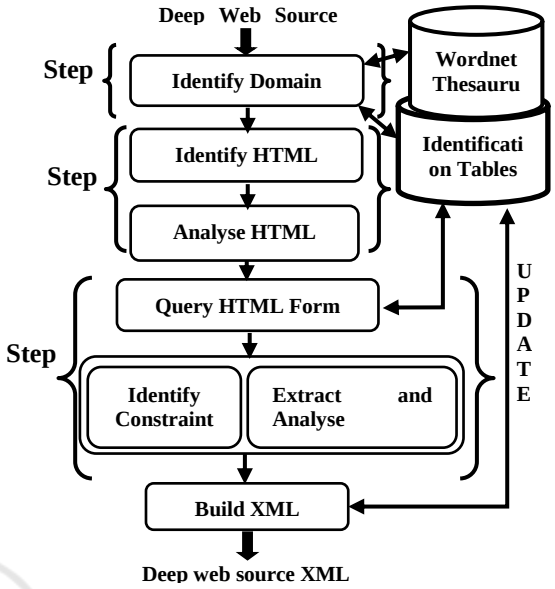

Figure 4: Our approach process.

The second step is the form processing step where the HTML forms associated with the deep web source are analysed at two different levels: lexically and structurally. The lexical analysis identifies the HTML form elements, with their labels, their data types and their constraints. The structural analysis identifies automatically the relations between the form fields in the HTML form source code. The third step is the form querying step where we use our proposed algorithm to query each HTML form with one objective: to complete the identification of all the elements of the form and to discover new ones from the web results. This step takes up two challenges: to generate the maximum of web results pages for the minimum of queries, and to propose the right values to the form fields with undefined values. For this, our approach uses the identification tables to propose values to these fields if the domain name is identified. Otherwise, we use a thesaurus as described in the next subsection. After each query, we analyse all the web results pages generated to extract their information structured in tables. These tables are analysed to extract new elements. These new elements are used with the elements extracted during the form processing step to build the XML schema describing the analysed deep web source.

In the last step of our approach, we update the identification tables with the new elements discovered after each deep web source analysis.

In the following subsection, we will describe, in more details, the three steps of our approach.

3.2 Detailed Process

3.2.1 Domain Identification Step

Before describing our first algorithm, we define first our knowledge database called the Identification tables. The identification tables are used to store each domain name with its associated keywords and labels. This knowledge database is used because the number of deep web domain name as seen in the previous section is limited. Thus, we can store and reuse the knowledge extracted from a deep web source to optimize the analysis of a new deep web source associated with the same domain name.

The identification tables are composed of three tables:

Domain Table = {(idDomain, $D_{idDomain}$), $1 \leq idDomain \leq n$};
Keyword Table = {(idDomain, K_j), $1 \leq j \leq m$};
Label Table = {(idDomain, (L_i, T_i)), $1 \leq i \leq p$};

Where $D_{idDomain}$ is the domain name, K_j are the keywords associated with $D_{idDomain}$, L_i are the labels associated with $D_{idDomain}$ and T_i the data type of L_i that can be a set of constants, an atomic type like integer, string … or a tuple of atomic types with or without constraints.

Before their first use, the identification tables are initialized with the known domain names with the trivial associated labels and keywords.

Algorithm 1: Domain identification
Input: Deep Web Source URL, N_w
Output: Domain Name

Begin
1: Pre-processing Filter;
2: Extract the most used words W_i, from the HTML components *Title*, *alt*, Headers *h1…h6* and the text;
3: Sort the set of tuples {(W_i, Occ_i), $1 \leq i \leq n$} by the number of their occurrence Occ_i;
4: Select the N_w tuples (W_i, Occ_i) with the maximal Occ_i;
5: **for** each selected word W_i **do**
6: Calculate its distance of Levenshtein d_{Lv} (W_i , K_j) With all the keywords K_j of the Keyword Table;
7: **if** (all d_{Lv} (W_i, K_j) $\neq 0$) **then** Search the domain of W_i in the Thesaurus Wordnet;
endfor;
8: **if (**all W_i does not have the same domain) **then**
9: $D_{url} \leftarrow$ the domain name of the W_i with the maximal Occ_i;
endif;
End

The algorithm one describes the first step of our approach. We first pre-process the web pages of the deep web source URL to filter out its template, the advertising content and the stop words to focus only on the valuable information of the web pages (Malki, 2002). After, we extract the N_w most used words from all the remaining HTML components of the web pages (lines 1-4). To identify the domain name of each word, we calculate its similarity with the elements of the keyword table using the distance of Levenshtein. If we succeed, the domain name found in the identification tables is associated with the similar word extracted (lines 5-6). If we do not find any similarity in the keyword table, we use a thesaurus to identify the domain name of the extracted word (line 7). At the end, if we identify different domain names for the same deep web source, we choose only one: the domain name identified and associated with the word extracted with the maximal occurrence (lines 8-9). The thesaurus used is the free thesaurus Wordnet able to propose a domain name for a given keyword.

3.2.2 Form Processing Step

Algorithm 2: Form Processing
Input: Deep Web Source URL
Output: Set of elements

Begin
1: **for** each HTML Tag *Form* **do**
2: Extract {(A_i, L_i?, T_i), d_k?, F_j?), $1 \leq i \leq n$, $1 \leq j \leq m$ and $1 \leq k \leq p$} where A_i is the HTML element name, L_i its label if it exist, T_i its data type, F_j its associated *Fieldset* and d_k its associated *div*;
endfor;
3: **for** each element A_i **do**
4: Extract the associated validation code constraint Jc_i ;
5: Enrich the data type Ti with the constraint Jci;
endfor;
End

This second algorithm describes the form processing step of our approach. It takes in entry the deep web source URL, and identifies its access HTML forms. After, the source code of each HTML form is analysed at different levels. Firstly, at the lexical level, we extract its elements with the associated labels and data types. Secondly at the structural level, we identify the elements associated with the same HTML tag Fieldset and or the same HTML tag div. At the end of this step, we obtain a set of tuples that identifies the element with its label, its data type and its associated Fieldset and div (lines 1-2). The validation code associated with each element is also analysed if it is accessible in order to extract new constraints to enrich the associated data type (lines 3-5).

3.2.3 Form Querying Step

The following algorithms three, four and five describe the form querying step.

Algorithm 3: Form Querying
Input: Deep Web Source URL, Stop condition SK
Output: Set of elements

Begin
1: k ← 0**;**
2: **for** each HTML Form **do**
3: Query all the fields with the default or empty value;
4: Generate Web results pages;
5: **Table Processing (**Web results pages);
6: Query all the defined fields with the possible values;
7: Generate Web results pages;
8: **Table Processing (**Web results pages);
9: **for** each undefined field **do**
10 **if** (the domain name exist in T_D) **then**
11: **if** (the label of the field exist in T_L) **then**
12: Query the undefined field with the associated values from T_L;
13: Generate web results pages;
14: **Table Processing (**Web results pages)**; endif; endif;**
15: **else while** (k <=SK) **do**
16: Query with new values from the Thesaurus Wordnet;
17: Generate web results pages;
18: **Table Processing (**Web results pages);
19: k ++;
endwhile; endfor; endfor;
End

Algorithm 4: Table Processing
Input: Web result page URL
Output: Set of elements

Begin
1: Extract the linked web results pages;
2: **for** each web results page **do**
3: Extract the HTML tables;
4: **if** (the HTML tables have a typed column) **then**
5: **if** (the HTML tables have a label for each column)
6**: then** Extract $\{(L_j, T_j), 1 \leq j \leq m\}$ where L_j is the label of the column j and T_j its associated set of values;
endif; endif; endfor;
End

The third algorithm is the form querying algorithm. It takes in entry the deep web source URL and starts by querying the associated HTML forms to generate the web results pages. The first query sent is the default query if it exists or the query with all the form fields empty (lines 1-4). After, we send all the possible queries on all the form fields with their known and defined values (line 6). For the fields with undefined values, we propose values from the Label table (lines 9-13). Otherwise, we use the thesaurus Wordnet to propose values for these fields with undefined values (lines 16-17).

During this step of querying fields with undefined values, we use a stop condition SK to optimize our automatic approach and not overload the deep web source server.

During all the process of this third algorithm, all the values used to query the form fields that generates No Results or Errors are identified as a constraint of the associated label.

In parallel, all the web results pages generated by our queries are analysed by our fourth algorithm. This algorithm identifies all the linked web pages by following the URLs of the associated next page links (line 1). For each web results page, we first identify all the HTML tables in the HTML source code and we focus only on the HTML tables that has strictly more than one line and where all the values of each column have the same data type. From these tables, we extract the labels of the columns and their set of values (lines 2-6). The table processing algorithm return the set of the extracted elements with their associated values.

Many techniques exist in the literature to identify the label of each column if it is not explicitly identified. We use in our approach the two following techniques. In the first technique, the elements of the same column are analysed to identify a maximal suffix or prefix. This maximal suffix or prefix can help to identify the label of the associated column (Lu, 2007). In the second technique, the elements of the same column are analysed to identify if a character exists in the same position and with the same occurrence in all the column elements. If this character exists in a predefined list {@, $, £, / /, etc...}, it can help to identify the label and or the data type of the associated column (Malki, 2003; Lu, 2007; Lu, 2013).

The algorithm five is used to identify the constraints of the form fields to enrich their data type and values. It takes in entry the selected label to analyse. After, it generates numerical and string values to query this label (lines 2-4 and 8-10). It checks if No Results or Errors are generated by this query, and if it is the case, the data type of the associated label is enriched with the new constraints (lines 5-6 and 11-12). At the end, the algorithm generates the numerical interval and or the regular expression of the constraints associated with the labels analysed.

Algorithm 5: Constraint Identification
Input: Label L, Stop Condition SK_{NUM}, SK_{ALPHA}
Output: Constraint of the label L

Begin
1: knum ←0; kalpha ←0; Constraint (L) = ∅;
2: **while** (knum < SK_{NUM})
3: Generate numerical value (knum, val);
4: Query the label L with val;

5: **if** (No Result or ERROR generated) **then**
6 : Enrich the Constraint (L) with the Interval (val) ; **endif ;**
7: knum++; Constraint Identification (L, knum, kalpha);
endwhile;
8: **while** (kalpha < SK_{ALPHA})
9: Generate string value (kalpha, str);
10: Query the label L with str;
11: **if** (No Result or ERROR generated) **then**
12: Enrich the Constraint (L) with the regular expression of str; **endif;**
13: kalpha++; Constraint Identification (L, knum, kalpha);
endwhile;
End

All the elements extracted by our approach are used to build our XML schema using the following rules: Firstly, the elements associated with the same HTML tag Fieldset define a new sequence of the XML Schema. Secondly, the elements associated with the same HTML tag div defines also a new sequence of the XML Schema. The constraints identified enrich the data type of the associated elements.

Algorithm 6: Identification Tables Update
Input: Identification tables T_D ,T_K ,T_L
Output: Updated Identification tables T_D ,T_K ,T_L

Begin
1: **if** the Domain name exist in T_D **then**
2: T_L is enriched with the new labels (L, T) identified;
3: T_K is enriched with the first new most used words extracted; **endif;**
4: **else**
5: $D_{idDomain}$ ← the New domain name founded;
6: T_L is enriched with the associated labels (L, T) extracted;
7: T_K is enriched with the first most used words extracted;
endelse;
End

The last algorithm six of our approach is used to update the identification tables at the end of the analysis of each new deep web source. This algorithm enriches the existing domain names with the new extracted keywords (lines 1-3). And if a new domain is identified, it updates the Domain table, the Keyword table and the Label table (lines 4-7).

3.3 Our Tool DWSpyder

We implemented an automatic tool called DWSpyder to experiment our approach. The DWSpyder tool is actually realized in the PHP language and it is using the thesaurus Wordnet with our private identification tables. The Figure 5 is a screenshot of the DWSpyder tool.

DWSpyder allows the user to enter the deep web source URL with some needed parameters, and it provides as output the associated XML schema.

The Figure 6 describes the XML schema extracted by the DWSpyder after the analysis of the associated HTML form of the deep web source CareerBuilder.com described in Figure 1.

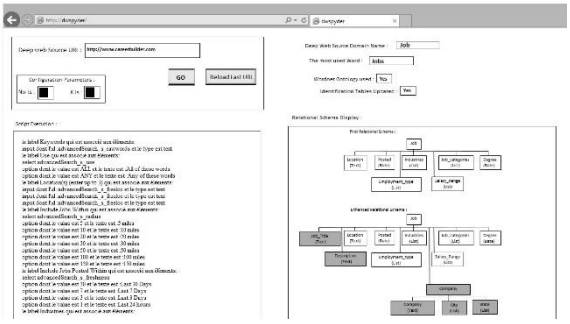

Figure 5: DWSpyder screenshot.

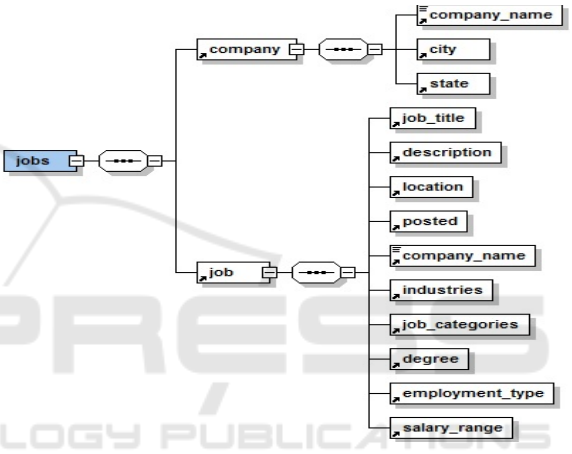

Figure 6: The XML schema extracted from CareerBuilder.com.

3.4 Discussion

As described in this paper, our approach takes the deep web source and identifies its domain name using the identification tables or the thesaurus Wordnet to identify the domain name.

The identification tables are our way to reuse for the same domain the information extracted from each deep web source analysed.

The HTML forms of the deep web source are analysed to extract all their information during the form processing step to discover the structure of the data that lies behind.

After, we query the form fields to extract more information about the structure of the data behind. During this querying step, the challenge is to optimize the queries of the fields with undefined values. For this, our first strategy is to use the label table if a similar label is identified. If it is not the case, we use the values proposed by the thesaurus Wordnet.

Following that, our approach analyses the web results pages generated and focuses on the tables extracted from these results because the deep web sources generates frequently the structured information table (Wang, 2003).

All the information extracted by our approach is used to build the XML schema describing the deep web source analysed. We choose the XML schema to describe the deep web source content because it is the most adapted way to describe the web content.

All our proposed algorithms have a low complexity O(n) except for the algorithms one and three that has the complexity $O(n^2)$ that is a still an acceptable complexity.

To resume, our approach uses all the accessible elements of the deep web source with a cumulated private knowledge about the deep web to maximize our chance to extract the most descriptive XML schema. And if we compare our approach to the crawling and the surfacing approaches, our approach needs only to extract the schema description and not all deep web source data and this optimize the number of the queries generated during our query processing step in comparison with these two approaches. And contrary to the form integration approach that sends the final user query to all the deep web sources through its global form to respond to it, the mediation system based on our extracted XML schema description with its associated constraints extracted can identify precisely the deep web sources that can respond to the final user query.

The XML schema extracted by our approach to describe a deep web source is a promising solution to integrate the deep web in a mediation system.

4 CONCLUSION AND FUTURE WORKS

Our objective is to be able to implement a web mediation system, and in this paper, we have presented our automatic approach to extract the needed XML schema describing a selected deep web source.

Our approach applies a static and dynamic analysis to the HTML forms giving access to the deep web sources. During its process, our approach uses its private knowledge about the deep web domains to optimize and enrich its XML schema extraction. And during the dynamic step, the web results pages generated by our queries are analysed to identify also the constraints of the form fields. All the elements extracted by our approach during the static and the dynamic analysis of the HTML forms are used to build the XML schema describing the associated deep web source.

The XML schema extracted is the key information needed to integrate the associated deep web source in a web mediation system able to respond to the end user query directly in the web environment.

Currently, we are working on many issues to improve our approach in parallel with the ongoing experiments on the DWSpyder tool. Firstly, we are working on the optimization of the number of the queries generated and also on the optimization of the content of the identification tables. For this we plan to score the values of the identification tables that generate interesting web results. These scored values will be used to optimize the number of queries generated to analyse the deep web source associated to each domain name.

We are also working on the analysis of the unstructured elements like the text of the web results pages to realise an exhaustive analysis of all the elements generated by our queries and not only the HTML tables.

REFERENCES

Bing, L. 2007. Web Data Mining. Springer.

Bergman, M.K. 2001. The Deep Web: Surfacing Hidden Value. The Journal of Electronic Publishing, Vol 7.

Chang. K. C.-C., He, B., Li, C., Patel, M., Zhang, Z. 2004. Structured Databases on the Web: Observations and Implications. ACM SIGMOD Record, Vol. 33, n. 3, 61-70.

Chang, K.C-C., He, B., Zhang, Z. 2005. Toward large scale integration: Building a MetaQuerier over databases on the web. In proceedings of the Second Conference on Innovative Data Systems Research (CIDR), 44-55.

Doan, A., Halevy, A., Ives, Z. 2012. Principles of Data Integration. Elsevier.

Furche, T., Gottlob, G., Grasso, G., Guo, X., Orsi, G., Schallhart, C. 2011. Real understanding of real estate forms. In Proceedings of the international conference on Web Intelligence, Mining and Semantics, Article No. 13.

Furche, T., Gottlob, G., Grasso, G., Guo, X., Orsi, G. Schallhart, C. 2013. The Ontological Key: Automatically Understanding and Integrating Forms to Access the Deep Web. VLDB Journal Volume 22, Issue 5, 615-640. DOI = http//doi.acm.org/ 10.1007/s00778-013-0323-0.

He, H., Meng, W., Yu, C., Wu, Z. 2005. WISE-Integrator : A System for extracting and integrating complex web search interfaces of deep web. In proceedings of the 31st VLDB conference, p.1314.

He, H., Meng, W., Yu, C., Wu, Z. 2005. Constructing interface schemas for search interfaces of web

databases. Web Information Systems Engineering chapter, Lecture Notes in Computer Science Volume 3806, 29-42. DOI = http//doi.acm.org/10.1007/ 11581062_3.

He, B., Patel, M., Zhang, Z., Chang, K. C.-C. 2007. Accessing the Deep Web: A survey. Communications of the ACM, Vol. 50, 94-101. DOI = http//doi.acm.org/10.1145/1230819.1241670.

Khelghati, M., Hiemstra, D., V.Keulen, M. 2013. Deep Web Entity Monitoring. In Proceedings of the 22nd International World Wide Web, page 377. DOI = http//doi.acm.org/ 10.1145/2487788.2487946.

Liddle, S.W., Embley, D.W., Scott, D.T., Yau, S.H. 2003. Extracting Data Behind Web Forms. In A. Olivé, M. Yoshikawa, E. S.K. Yu (Ed.), Lecture Notes in Computer Science, Advanced Conceptual Modeling Techniques chapter, Vol. 2784, 402-413. DOI = http//doi.acm.org/10.1007/978-3-540-45275-1_35.

Lu, Y., He, H., Zhao, H., Meng, W. 2007. Annotating structured data on the deep web. In IEEE 23rd International Conference on Data Engineering, ICDE, 376-385. DOI = http//doi.acm.org/ 10.1109/ ICDE.2007.367883.

Lu, Y., He, H., Zhao, H., Meng, Yu, C. 2013. Annotating Search Results from Web Databases. In IEEE Transactions on Knowledge and Data Engineering, Vol. 25, Issue 3, 514-527. DOI = http//doi.acm.org/ 10.1109/TKDE.2011.175.

Lyman, P., Varian, H.R. 2003. How Much Information. 2003? University of California.

Madhavan, J., D. Ko, L. Kot, V. Ganapathy, A. Rasmussen, A. Halevy. 2008. Google's Deep Web Crawl, Proceedings of the VLDB Endowment, page 1241, ISSN: 2150-8097.

Malki, M., Flory, A., Rahmouni, M.K. 2002. Extraction Of Object-Oriented Schemas from existing relational databases: a Form-driven Approach. INFORMATICA, Vol. 13, No. 1, 47-72.

Saissi, Y., Zellou, A., Idri, A. 2014. Form driven web source integration. In proceedings of the 9th IEEE International Conference in Intelligent Systems: Theories and Applications.

Saissi, Y., Zellou, A., Idri, A. 2014. Extraction of relationl schema from deep web sources: a form driven approach. In Proceedings of the 2nd IEEE World Conference in Complex System. DOI = http//doi.acm.org/ 10.1109/ICoCS.2014.7060888.

Saissi, Y., Zellou, A., Idri, A. 2015. Deep web integration: Architecture for relational schema extraction. In Proceedings of the 26th International conference on Software & Systems Engineering and their Applications.

Saissi, Y., Zellou, A., Idri, A. 2015. Deep Web integration : the tip of the iceberg. In International Review on Computers and Software, Vol 10, n.10. DOI = http://dx.doi.org/10.15866/irecos.v10i10.7755.

Wang, J., Lochovsky, F. H. 2003. Data Extraction and Label Assignment for Web Databases. WWW 2003, ACM 1-58113-680-3/03/0005. DOI = http//doi.acm.org/ 10.1145/775152.775179.

Zellou, A. 2008. Contribution to the LAV rewriting in the context of WASSIT, toward a resources integration. Doctoral Thesis, Dept. Computer Engineering, University Mohammed V, EMI, Rabat, Morocco.

Zhang, Z., He, B., Chang, K. C-C. 2004. Understanding web query interfaces :Best-effort parsing with hidden syntax. In proceedings of the ACM SIGMOD international Conference on Management of data, p. 107. DOI = http//doi.acm.org/ 10.1145/ 1007568.1007583.