An Improved Approach for Deep Web Data Extraction

Shilpa Deshmukh
Department of Computer Application
SIES college of Management Studies
Navi Mumabi, India
shilpad@sies.edu.in

P.P. Karde
Department of Information Technology Government Polytechnic,
Amravati,,India.
p_karde@rediffmail.com

V.R.Thakare
Department of Computer Science, SGB Amravati University,
Amravati, India
vilthakare@yahoo.co.in

Abstract
The World Wide Web is a valuable wellspring of data which contains information in a wide range of organizations. The different organizations of pages go about as a boundary for performing robotized handling. Numerous business associations require information from the World Wide Web for doing insightful undertakings like business knowledge, item insight, serious knowledge, dynamic, assessment mining, notion investigation, and so on Numerous scientists face trouble in tracking down the most fitting diary for their exploration article distribution. Manual extraction is arduous which has directed the requirement for the computerized extraction measure. In this paper, approach called ADWDE is proposed. This drew closer is essentially founded on heuristic methods. The reason for this exploration is to plan an Automated Web Data Extraction System (AWDES) which can recognize the objective of information extraction with less measure of human intercession utilizing semantic marking and furthermore to perform extraction at a satisfactory degree of precision. In AWDES, there consistently exists a compromise between the degree of human intercession and precision. The objective of this examination is to diminish the degree of human intercession and simultaneously give exact extraction results independent of the business space to which the site page has a place.

Keywords: www, deep web, crawler, deep web data extraction, heuristic approach
.

1. Introduction

The growth of World Wide Web (WWW), which is a huge repository of information, is unimaginable. Www opens the doors to the huge amount of data for the world. The search engines help us to retrieve the needed information from such a huge repository. It provides us with a list of links to the websites which might contain the requested information. Nevertheless large data remains invisible to the user because huge amount of information on the web is available today only through search interfaces. E.g. if the user wants to find the information about the flights she has to go to that airways web site and fill the details in a search form. It will display the flights and the available seats as a result, which are dynamically generated pages containing the required information. This part of web which is reachable only through search interface is deep web. Deep web pages are generated through dynamic query on the web databases. As compared to surface web or static web, contains larger amount of data with higher quality as well.

In order to get the information from the deep web, customized search engines provided by the individual websites are used. Such search engines submit the user query to the backend database and retrieve the structured records matching the query.

2. Literature Review

Various web data extractions have been listed in the literature review. In this section, previous work done on deep web data extraction are being reviewed briefly. Detailed survey about information extraction is given in [1-A survey of web Information Extraction System], where three parameters:

Task Difficulties, used techniques, Degree of Automation were considered.

TSIMMIS by Hammer et al. [2]: The target of extraction is specified manually using a specification file written using a declarative language. The command consists of three parts, namely, variables, source and pattern. The variables are used to store the result of extraction. The pattern specifies how to determine the content to be extracted and the source specifies the input text to be extracted. The output of extraction is represented using Object Exchange Model (OEM). It contains the actual data as well as structure of the data.

W4F by Sahuguet et al. [3] (World Wide Web Wrapper Factory): It is a Java Tool Kit used to generate extraction rules. It consists of three steps, namely, retrieval, extraction and mapping layers. The first layer is the retrieval layer which involves cleaning the input HTML document and converting it to the DOM tree. The second layer, namely, the extraction layer involves applying the extraction rules to the DOM tree in order to extract the target data. The output target data is represented as Nested String List structure.

Supervised Extraction Techniques

It requires less human intercession than the past procedure. This method doesn't include manual formation of covering. All things considered, a bunch of marked pages is given as contribution to the framework which creates the covering consequently. Some of generally utilized apparatuses dependent on this strategy are:

WIEN by Kushmerick et al. (Covering Induction for data ExtractioN) [4]: In this paper, the creators presented covering acceptance, which can develop coverings naturally by summing up from model question reactions. The authors have proposed six types of wrappers, namely, LR (Left-Right), HLRT (Head-Left-Right-Tail), OCLR (Open (Left token Right)* Close), HOCLR (Head Open (Left token Right)* Close), N-LR (Nested-Left-Right) and N-HLRT (Nested-Head-Left-Right-Tail). LR (Left-Right) wrappers are simple class of wrappers which scans the input document to determine left hand delimiter and right-hand delimiter for each and every attribute, until all the columns in the table are extracted. HLRT (Head-Left-Right-Tail) wrappers are more sophisticated than LR wrappers which can be used to extract content from country/code resource. OCLR uses O and C to identify tuples and LR is repeated for individual elements. HOCLR merges the functionalities of HLRT and OCLR. N-LR and N-HLRT searches for the delimiters in a different way and they are extension of LR and HLRT wrappers which can perform extraction of nested data records. Soft Mealy by Hsu et al. [5]: Hsu et al. introduced FST (Finite State Transducer) in order to handle different types of extraction structures. It consists of two different modules namely body transducer and tuple transducer. Body transducer is used to extract the data rich region containing the tuples and the tuple transducer is used to extract the tuples from the data rich region.

STALKER by Muslea et al. (Learning Extraction Rules for Semi-structured, Web based Information Sources) [6]: The authors have introduced a concept called Embedded Catalog (EC) formalism which is a tree-like structure in which the leaves are the attributes to be extracted and the non-leaf nodes represent the data records. It uses multiple passes to determine missing attributes and differently arranged attributes.

- *Semi-Supervised Extraction Techniques*

Unlike the supervised technique in which labelled examples are needed, these tools require post-effort (i.e. manual labelling after extraction). These techniques are appropriate for record-level extraction. The user's effort is needed to specify extraction targets using GUI. Unlike the supervised techniques which require comprehensive set of labelled training samples, these techniques require rough example for wrapper creation.

In [7], Chang et al. proposed a semi-supervised technique in which part of input data set is used for inferring schema. A Finite State Machine is built based on inferred schema and the inferred template is used for extraction of data. This technique is greatly

affected by structural changes of the web pages as it might require changes to the inferred template.

- *Unsupervised Extraction Techniques*

These techniques require neither labelled examples nor user interaction for wrapper induction. Some of the matured unsupervised page-level extraction techniques are:
RoadRunner by Crescenzi et al. [8], EXALG by Arasu et al. [9], FivaTech by Kayed et al. [10] are page-level extraction techniques whereas certain others like DeLa by Wang et al. [11] and DEPTA by Zhai et al. [12] are record-level extraction techniques.

DeLa (Data Extraction and Label Assignment) by Wang et al. [13]: It consists of four major modules, namely, *form crawler, wrapper generator, aligner* and *label assigner*. The form crawler is used to submit queries and to obtain the response pages pertaining to the query. The wrapper generator determines the template in which the data records are embedded. The aligner is responsible for grouping the attribute values so that all the attribute values in a group belong to the same concept. The label assigner assigns meaningful labels to individual data units. Hierarchical Web Crawler is used to perform form submission and to get the Search response pages. DeLa uses wrapper generator derived from IEPAD [14] system. Data-rich Section Extraction (DSE) [15] technique has been employed to strip off non-informative sections of the web page. DSE [16] has been identified by comparing the HTML source code of multiple web pages generated using same server-side templates. In order to identify C-repeated pattern (Continuous Repeated Pattern), suffix-tree has been constructed. Pattern Tree is constructed by starting from an empty root node. Whenever a pattern is determined then it is added as a child to the root node in the Pattern Tree. Longest occurring pattern in the Pattern Tree (PT) represents the general template used for generating the multiple input pages. The shortcoming of this technique is that it won't work with disjunctions in the structure of the web page. DeLa [17] not only handles extraction but also annotation. The next step post extraction is aligning the attribute values. It computes similarity between data units using features such as tag path, presentation styles, etc., and groups the attribute values into disjoint sets. Annotation phase uses four different heuristics such as form keywords, common prefix/suffix, values matching conventional formats and keywords in the query/response page to determine appropriate label to each attribute value group.

RoadRunner by Crescenzi et al. [18]: This tool attempts to perform unsupervised extraction by comparing two documents generated using the same template. It finds the similarities and differences between the documents. It generates a union-free regular expression which matches both the input documents. It assumes one of the input pages as the wrapper and compares it with the other page. Whenever a mismatch is encountered, the wrapper is generalized. The mismatches are of two types, namely, tag mismatches and string mismatches. Tag mismatches are resolved by including optional or iterator in the wrapper. Whenever a tag mismatch is encountered, it finds the initial tag and the final tag of the mismatched block and denotes it by a square. Then it matches this block to its upper portion of the source code to find whether it represents a repeated pattern. If it represents a repeated pattern, then it is enclosed inside (…) + to represent repetition. If no such pattern can be found then, it tries to search the wrapper to find whether the block can be skipped. In that case, it represents an optional block which is denoted by enclosing the pattern shows result of applying Road Runner technique [19] for two sample input pages. It shows how repetition of block is determined and how the wrapper is generalized. Whenever string mismatch is encountered, it represents data units and therefore it is replaced by #PCDATA which is used to denote any sequence of characters.
Limitations:

i. Time complexity is exponential in the number of tokens of the input documents. In order to reduce the complexity the authors

have proposed certain strategies such as reducing the number of alternatives to be considered, the number of backtracking to be performed to identify iterations, and removal of some regular sub-expressions.

i. Biases introduced have negative impact on the effectiveness of the technique.

ii. It requires the input documents to be well-formed.

iii. It does not perform automatic annotation.

OMINI[20] also performs unsupervised extraction similar to Road Runner[21]. It consists of three phases, namely, preparing the input web page, identifying target of extraction, and extracting target the data records. The input web page is pre-processed. Pre-processing involves cleaning the web page in order to ensure that it is well-formed and then transforming it to tag tree representation. Identifying target of extraction requires two steps, namely, object rich sub-tree discovery and object separator extraction. The portion of sub-tree containing all data objects of interest is identified by considering features such as number of child nodes of a given node (fan out), the number of bytes of data corresponding to each node and the count of sub-trees. Object separator extraction involves finding the tags used as separators of data records. Five different heuristics are applied for this purpose.

Authors have made assumptions such as most of the tokens should be associated with type constructor, tokens belonging to the template must have unique roles, and the data must be enclosed between tags. Most of the real world websites do not satisfy all these assumptions which affect the effectiveness of the technique.

MDR by Liu et al. (Mining Data Records) [21]: MDR uses two observations: i. a group of data records representing similar objects are placed in contiguous section in the web page and formatted using similar HTML elements, ii. a set of similar records appear as child sub-trees of a common parent node in the tag tree representation of the HTML page. In DEPTA [22], tag tree is constructed based on nested rectangles since each HTML element is rendered using a rectangular box by the browser. The co-ordinates of all the boxes are determined and the co-ordinates are compared in order to know the containment relationship among boxes and then the tag tree is constructed accordingly. Figure 1 shows the construction of tag tree from the visual clues. In order to find the data records region, a measure called string edit distance as in OLERA by Chang et al. [23] is used for the comparison of tag strings of individual nodes and combination of multiple adjacent nodes.

FiVaTech by Kayed et al. [24] (Page-level web data extraction from template pages): It is an unsupervised approach for carrying out page-level extraction task. It consists of two phases: the first phase involves transforming the input documents into collection of DOM trees. The DOM trees are converted into pattern trees which are used to determine the server-side templates involved in generating the input documents; and the second phase involves producing schema from the pattern tree and extracting the embedded data records. But limitations are 1) FiVaTech requires well-formed input documents and error during parsing affects the effectiveness of the approach. 2) It uses pattern matching algorithm which is computationally expensive. 3) It uses a threshold for identifying peer nodes which impacts accuracy if proper threshold is not used.

Almost all methods discussed above methods can extract web data in the mentioned conditions.

3. PROPOSED ARCHITECTURE

From Literature Review it is clear that most of the existing systems have made assumptions such as the availability of multiple input pages; occurrence of missing or optional attributes is rare and so on. Also, these techniques are based on string pattern matching or DOM tree matching and therefore, change in structure of the web page requires occurrence of repeated patterns in the wrappers generated. This problem is addressed in the proposed technique by considering semantic information as the basis of extraction rather than structure or template of the web page. The steps proposed AWDES is as shown in the Figure 1:

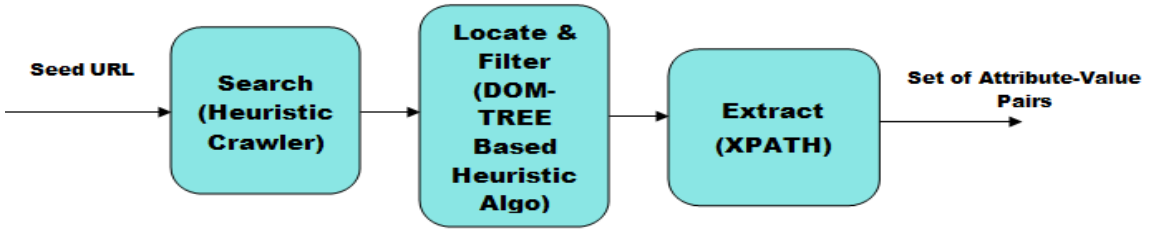

Figure 1: 1 Steps in Web Data Extraction

It involves four steps, namely, Search, Locate, Filter and Extract. The search step involves finding the target web pages given the URL of the website as input. The locate step refers to determining data rich regions in the target web pages and filter step involves the process of extracting the data rich nodes after eliminating non-informative sections such as navigation links, pop-up menus, advertisements, etc. The last step of web data extraction involves extracting the attribute-value pairs from the target web pages, which represents the description of the object.

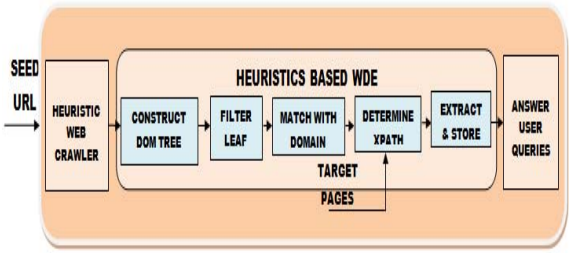

Figure 2: Overall Architecture of the Journal Information Extraction System

3.1 Mathemaical Model

Heuristic Algorithm for Automatic Navigation

Given the URL of the logical distributer's site urlp, explore to the site and concentrate the arrangement of URLs comparing to the diary home pages orchestrated in sequential request (i.e.) A-Z. In the wake of exploring to diary's landing page, it's anything but a bunch of diary particular addressing data like title, ISSN, URL of the diary's landing page, and so on URLs for every one of the diaries are requested by A-Z. The arrangement of target URLs is meant by H = {hi, $1 \leq I \leq n$}. Every diary landing page hello is parsed to gather the required properties like title, ISSN, Impact

3.2 Heuristic Technique for Data Extraction

The proposed approach is based on the observation that the journal home pages linked to publishers' website are well formatted. The uniformity in presentation across several web pages of the website is achieved since they are generated using the same server-side template. If the location of target data is identified for a single web page (XPATH for attribute name-value pairs) from the scientific publisher's website, then the same XPATH can be used for extraction from similarly formatted journal specification pages. Overall architecture is shown in Figure 2:

The point of this procedure is to recognize the area of the objective ascribes, separate the quality qualities, explain and store the removed information records containing property estimations into a social data set. The procedure depends on the heuristics that the objective data to be separated is available as apparent substance which is shown in the substance space of the program. In the DOM tree addressing the site page, all the content hubs will be available as leaf hubs. The method includes visiting every URL and getting the DOM tree of the page containing the diary determination. The calculation getLeafNodes is utilized to acquire the leaf hubs and furthermore, to check whether it's anything but a content hub. On the off chance that they match with the area catchphrases like SJR, SNIP, Impact Factor, and so on, then, at that point its XPath is resolved utilizing the system getFullXPath. The calculation convert_XPath_to_Selector is utilized for changing the XPath over to a selector string. A similar selector string can be utilized to do extraction from correspondingly organized website pages. Following are the algorithm used in proposed approach:

```
Algorithm Extract (Seed_URL)
//Input: Seed URL(URL of Publishers' site)
//Output: Structured records
begin
  target_urls=extractURLs(Seed_URL);
  for each url in target_urls do
  impact_factor, issn = display_journal_metrics(url)
  create database connectivity
  store impact_factor, issn
  end for
end
```

Figure 3: Algorithm Extract

```
Algorithm displayJournalMetrics(target_url)
//Input: target url
//Output: impact_factor,issn
begin
   useragent.visit(target_url);
   issn_pattern = "\\d\\d\\d\\d-\\d\\d\\d\\d|\\d\\d\\d\\d-\\d\\d\\dX"
   impact_pattern = "\\d.\\d{1,3}|\\d\\d.\\d{1,3}"
if (issn_xpath is NULL) then
begin
  doc = construct DOM tree
  root = node corresponding to Body element
  leaf = getLeafNodes(root)
for each node in leaf do
if node.text matches issn_pattern then
  issn_xpath = getFullXPath(node)
  issn = node.text.substring(start, end)
end if
if node.text matches impact_pattern then
  impact_xpath = getFullXPath(node)
  impact_factor = node.text.substring(start,end)
end if
else
  impact_factor_selector = convert_xpath_to_selector(impact_xpath)
  issn_selector = convert_xpath_to_selector(issn_xpath)
  impact_node = doc.select(impact_factor_selector)
  impact_factor = impact_node.text.substring(start,end)
  issn_node = doc.select(issn_selector)
  issn = issn_node.text.substring(start, end)
end if
  return impact_factor,issn
end
```

Figure 4: Algorithm displayJournalMetrics

```
Algorithm getLeafNodes (Node root)
//Input: root
//Output: leaf_nodes
begin
if (root.getType() equals Node.ELEMENT_TYPE) then
for each node in root.getChildNodes()
begin
  if node.getType() equals Node.TEXT_TYPE then
     leaf_nodes.add(node)
  else
     getLeafNodes(node)
  end if
return leaf_nodes
end
```

Figure 5: Algorithm getLeafNodes

4. Result & Analysis

The experiment was conducted on a machine having Intel Core 2 Duo T5870 @ 2.0 GHz with 3 GB of RAM. This machine was running with Windows 7 OS. Tests were performed using WAMP server equipped with php v. 5.2.6 and mysql v. 5.0.51b., Microsoft Visual Basic 2008 .net 3.5 and Net beans IDE. All the tests were performed on Firefox 3.5.4. and were performed multiple times to minimize measurement

Simulation of proposed architecture of deep web crawler and their result are analysed with the help of performance metrics like Precision, Recall and F-measure. Three domains have used i.e. book, auto, job for our analysis. Detailed analysis regarding the book domain is presented in this section. Six commercial implementation of book domain i.e. www.borders.com, www.bookshare.org, www.textbooks.com, www.ecampus.com, www.textbooksrus.com, ww.bookfinder.com for extracting the content of deep web by querying through search interface have been used. QIIIEP server is used for fetching query words related to domain for level value assignment. The GET and POST method is analysed through action tag and stored in query words to the url relation manager. The method implemented at local host is named as fetch query interface and the query argument used as sdd, stands for search deep domain. http:\\localhost\fetchqueryinterface?sdd=borders.com

To measure the performance of proposed deep web crawler "DWC", three metrics are defined [32] are as follows:

i) Precision, which is defined as *PR = CDWP/ (CDWP + WDWP)*

ii) Recall is *RE = CDWP/ (CDWP + NCDWP) and*

iii) F-measure is *FM = 2PR.RE/ (PR + RE)*

Where Precision (PR) and Recall (RE) are weighted equally.

For Book domain, about 1400 query words received & fired and response pages were analysed. It may be noted from Fig. 8 that as the number of received queries word increases for Site1:www.borders.com, the value of Precision, Recall and F-measure increases. When about 200 received query words were fired, the value of the Precision, Recall and F-measure were 73%, 75%, and 74.03% respectively. As the number of query word reached to 1400, the values of the Precision, Recall and F-measure reached at 97%, 89% and 92% respectively.

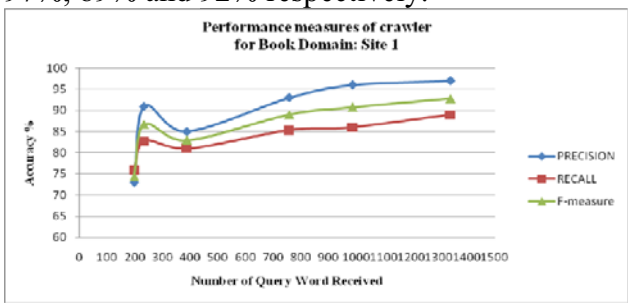

Figure 6: Performance Measure

The number of received queries word increases for Site2: www.bookshare.org, the value of Precision, Recall and F-measure increases. When about 200 received query words were fired, the value of the Precision, Recall and F- measure were 67%, 70%, and 68% respectively. As the number of query word reached to 1400, the values of the Precision, Recall and F-measure reached at 93%, 85% and 88% respectively.

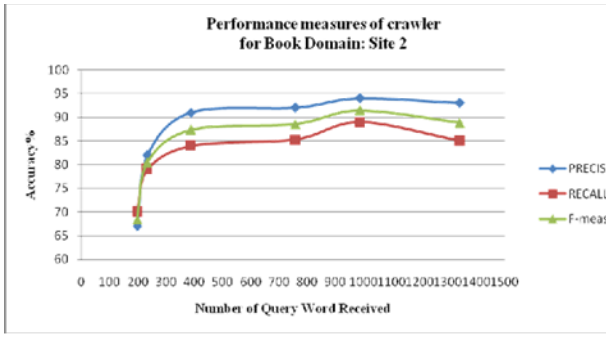

Figure 7: Performance Measure

Figure 6 and 7 shows the performance measures of proposed deep web crawler "DWC" at different quantity of query words received by QIIIEP server that are used to fetch the content from the deep web sites. From the analysis of this graph we can easily visualize that the number of query words received for specific domain, the precision, recall and f-measure of information extraction have continuously increased. So it can be concluded that when enough knowledge base is collected at QIIIEP server, the result is obviously improved.

5. Conclusion

The presence of high quality information in Deep Web Pages which are dynamically generated by embedding structured data records in server-side templates has made many researchers design techniques for automatic extraction of such data. The limitation of the existing systems includes their dependency on the structure of HTML pages for inducing wrappers. Our proposed research addressed this challenge.

This system enables integration of journal information from multiple publishers' websites and it enables the user to get the complete information about journals in a single interaction with the system. The system can be used to extract data records from websites where the attribute label is present explicitly. It is not always true since some websites like social discussion forums are not having attribute labels mentioned explicitly.

6. References

[1] Luciano Barbosa and Juliana Freire, "Searching for Hidden-Web Databases", In Proc. of Web Database, 2015.

[2] J. Cope, N. Craswell, and D. Hawking, "Automated Discovery of Search Interfaceson the web", In Proceedings of the Fourteenth Australasian Database Conference (ADC2019), Adelaide, Australia, 2019.

[3] Z. Zhang, B. He, and K. Chang, "Understanding Web Query Interfaces: Best-Effort Parsing with Hidden Syntax", In Proceedings of ACM International Conference on Management of Data, pp.107-118, 2019.

[4] L. Barbosa and J. Freirel, "Siphoning Hidden-Web Data through Keyword-Based Interface" , In Proceedings of SBBD, 2004.

[5] X. B. Deng, Y. M. Ye, H. B. Li, and J. Z. Huang, "An Improved Random Forest Approach For Detection Of Hidden Web Search Interfaces", In Proceedings of the 7th International Conference on Machine Learning and Cybernetics, China. IEEE, 2008.

[6] Y. Ye et al., "Feature Weighting Random Forest for Detection of Hidden Web Search Interfaces", In Computational Linguistics and Chinese Language Processing , Vol.13, No. 4, pp.387-404, 2019.

[7] P. Bai, and J. Li, "The Improved Naive Bayesian WEB Text Classification Algorithm", In International Symposium on Computer Network and Multimedia Technology, IEEE Explorer, 2019.

[8] Anuradha and A.K. Sharma, "A Novel Approach For Automatic Detection and Unification of Web Search Query Interfaces Using Domain Ontology", In International Journal of Information Technology and Knowledge Management , Vol. 2, No. 2, 2010.

[9] J. Madhavan, P. A. Bernstein, and E. Rahm, "Generic Schema Matching with Cupid", In the 27th VLBB Conference, Rome, 2009.

[10] H. H. Do, and E. Rahm, "COMA-a System for Flexible Combination of Schema Matching Approaches", In Proceedings of the 28th Intl. Conference on Very Large

Databases (VLDB), Hong Kong, 2012.
[11] A. H. Doan, P. Domingos, and A. Levy, "Learning source descriptions for data integration", In Proceedings of the WebDB Workshop, pp. 81-92, 2017.
[12] A. Doan, P. Domingos, and A. Halevy, "Reconciling schemas of disparate data sources: A machine learning approach", In Proceedings of the International Conference on Management of Data (SIGMOD), Santa Barbara, CA, New York: ACM Press, 2011.
[13] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity Flooding: A Versatile Graph Matching Algorithm", In Proceedings of the l8th International Conference on Data Engineering (ICDE), San Jose, CA,2019.
[14] O. Kaljuvee, O. Buyukkokten, H. G. Molina, and A. Paepcke, "Efficient Web form entry on PDAs", In Proceedings of the 10th International Conference on World Wide Web, pp. 663- 672, 2018.
[15] H. He, W. Meng, , C. Yu, and Z. Wu, "Constructing interface schemas for search interfaces of Web databases", In Proceedings of the 6th International Conference on Web Information Systems Engineering (WISE'05),pp. 29-42, 2015.
[16] W. Wu, C. Yu, A. Doan, and W. Meng, "An interactive clustering-based approachto integrating source query interfaces on the Deep Web", In Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 95-106, 2014.
[17] B. He, and K. C. C. Chang, "Automatic complex schema matching across web query interfaces: A correlation mining approach", In Proceedings of the ACM Transaction on Database Systems, Vol.31, pp.1-45, 2016.
[18] P. Lage et al., "Collecting Hidden Web Pages for Data Extraction" In Proceedings of the 4th International workshop on Web Information and Data Management, pp. 69-75, 2012.
[19] B. He, and K. C. C. Chang, "Statistical schema matching across web queryinterfaces", In the SIGMOD Conference, 2013.
[20] X. Zhong, et al., "A Holistic Approach on Deep Web Schema Matching", In Proceedings of the International Conference on Convergence Information Technology, 2017.
[21] S. Raghavan and H. Garcia-Molina, "Crawling the Hidden Web", In Proceedings of the 27th International Conference on Very Large Data Bases, Roma, Italy, pp. 129–138, 2016.
[22] B. Qiang, J. Xi, B. Qiang, and L. Zhang, "An Effective Schema Extraction Algorithm on the Deep Web. Washington, DC: IEEE, 2018.
[23] M. Niepert, C. Buckner, and C. Allen, "A Dynamic Ontology for a Dynamic Reference Work", In Proceedings of the (JCDL'17), Vancouver, Canada, 2017.

[24] A. Deitel, C. Faron, and R. Dieng, "Learning ontologies from rdf annotations", Inthe IJCAI Workshop in Ontology Learning, 2001.