International Journal of Information Retrieval Research
Volume 12 • Issue 1

A Framework for Automated Scraping of Structured Data Records From the Deep Web Using Semantic Labeling:
Semantic Scraper

Umamageswari Kumaresan, Pondicherry Engineering College, India
Kalpana Ramanujam, Pondicherry Engineering College, India

ABSTRACT

The intent of this research is to come up with an automated web scraping system which is capable of extracting structured data records embedded in semi-structured web pages. Most of the automated extraction techniques in the literature capture repeated patterns among a set of similarly structured web pages, thereby deducing the template used for the generation of those web pages, and then data records extraction is done. All of these techniques exploit computationally intensive operations such as string pattern matching or DOM tree matching and then perform manual labeling of extracted data records. The technique discussed in this paper departs from the state-of-the-art approaches by determining informative sections in the web page through repetition of informative content rather than syntactic structure. From the experiments, it is clear that the system has identified data rich region with 100% precision for websites belonging to different domains. The experiments conducted on the real-world websites prove the effectiveness and versatility of the proposed approach.

KEYWORDS

Deep Web, DOM Tree, HTML, Server-Side Templates, Structured Data, Supervised Extraction, Surface Web, Unsupervised Extraction, Web Scraping, XPATH

INTRODUCTION

Web Scraping involves extracting enormous amount of data embedded in semi-structured HTML pages. The amount of information available with deep web is of several orders of magnitude higher than the surface web. The surface web refers to those web pages indexed to search engines like google, yahoo etc. The deep web refers to web pages that are generated dynamically by querying the back-end database and embedding the resultant data records in server-side templates. Deep web is also referred to as Dark web since it not indexed to search engines. The degeneration of data records is not straightforward since the web pages are intended for human understanding. Getting the data from deep web is easy if the owner of the web site provides the API for accessing it. This is not true in most of the cases since it requires technical expertise and some are not willing to outsource their data. It is due to this reason web scraping is the only solution to get the data from Deep web.

Data from deep web acts as a complementary source of information for many data analytics applications such as opinion mining, sentiment analysis, product intelligence, customer intelligence

DOI: 10.4018/IJIRR.290830

and many more. The huge amount of information needed by the data analytics application is available in the Dark web or Deep web. The first step is to collect the data from the deep web pages. Performing copy paste is practically infeasible since the number of web pages to be processed is huge. Therefore, only possibility is to come up with an automated system which can identify target pages and perform extraction. The problem of web data extraction can be stated as follows:

Let web site S consists of collection of template generated web pages $P = \{p_1, p_2, p_3 \ldots pi \ldots p_m\}$ where each web page pi consists of set of data objects D = {d1, d2, d3…dr}. Each data object d_j in D is a set of attribute value pairs $\{<x_1,y_1>,<x_2,y_2>\ldots\ldots<x_n,y_n>\}$. The problem of web data extraction involves extraction of D from every p_i in P belonging to S.

The design of web data extraction system should be capable of handling various challenges such as heterogeneity of structuring of web pages belonging to different web sites, missing attributes, several levels of nesting within templates in which data records are embedded, identification of extraction target, semantic representation of extracted data, automatic labeling and so on. Although many commercial tools such as Lixto (Baumgartner, Gatterbauer, & Gottlob, 2009), import.io (https://www.import.io/), Connotate (https://www.connotate.com/) are available for web data extraction, their usage requires understanding of site map, manual selection of extraction targets. Many automatic approaches such as ExAlg (Arasu & Garcia-Molina, 2003), RoadRunner (Crescenzi, Mecca, & Merialdo, 2002), FiVaTech (Kayed & Chang, 2010) and Trinity (Sleiman & Corchuelo, 2014) exist in the literature. Semantic Scraper departs from these techniques in the following ways:

1. Automatic identification of data rich section
2. Automatic labeling of extracted data records
3. Ability to extract from a single input page

Section 2 discusses about the state-of-the-art approaches in the literature. Section 3 explains the architecture of Web Data Extraction System (WDES) based on Semantic Labeling, Section 4 shows experimental results and comparison with other state-of-the-art techniques and Section 5 discusses about conclusion and future scope.

RELATED WORKS

The problem studied in this work is concerned with automatic identification of Data Rich Region and extraction of structured records from data rich region. Many state-of-the-art techniques exist in the literature ranging from hand crafted extractors to unsupervised extractors. Initially wrappers are written manually using some extraction programming languages. Writing hand crafted wrappers require high level of expertise which forced the researchers to automatically induce wrappers from labeled training samples. It involves the following steps: Labeling of training samples, learning extraction rules from labeled training samples and applying rules to extract items from similarly structured pages. Systems such as WIEN (Kushmerick, Weld, & Doorenbos, 1997), SoftMealy (Hsu & Dung, 1998), Stalker (Muslea, Minton, & Knoblock, 1998), IEPAD (Chang & Lui, 2001) and Thresher (Hogue & Karger, 2005) are examples of wrapper induction techniques. Limitations of these methods include manual labeling of training examples, accurate learning requires large number of training samples, manual labeling is laborious and time consuming and wrapper maintenance is costly. Later on, many unsupervised techniques came into existence. Some techniques focus on identification of Data Rich Regions (Sleiman & Corchuelo, 2013). (Baskaran & Ramanujam, 2017), (Chang, Kayed, Girgis, & Shaalan, 2006), (Laender, Ribeiro-Neto, da Silva, & Teixeira, 2002) provide comprehensive survey of web data extractors available in the literature.

The proposed approach is compared with the related state-of-the-art approaches such as Trinity (Sleiman & Corchuelo, 2014), RoadRunner (Crescenzi, Mecca, & Merialdo, 2002), ExAlg (Arasu &

Garcia-Molina, 2003) and FiVaTech (Kayed & Chang, 2010). The reason for limiting our comparison to these techniques is that these techniques have been experimented on real world websites. RoadRunner (Crescenzi, Mecca, & Merialdo, 2002) tries to learn Union Free Regular Expression by comparing initial set of rules deduced from input pages with newly seen page and tries to generalize the partial rule learnt when it encounters mismatches to include optional repetitive structure. The time complexity of the algorithm is exponential and the authors has come out with several heuristics to lower the time complexity. ExAlg (Arasu & Garcia-Molina, 2003) considers document as a set of tokens and finds the number of occurrences of each token. It then finds large and frequently occurring equivalence classes (LEFQ) and then learns regular expression. Although determining frequency of occurrence of tokens is simple, finding invalid LEFQs whose tokens do not appear in the same order is complex. ExAlg (Arasu & Garcia-Molina, 2003) has many assumptions such as many number of tokens must have unique roles, each type constructor such as union, group, repetition and optional must be instantiated many times in the document and there should be separators around the attributes. FiVaTech (Kayed & Chang, 2010) is a page-level extraction system based on tree merging and schema deduction. Tree merging involves merging DOM trees simultaneously into a structure called fixed/variable pattern tree. It consists of peer node recognition, peer matrix alignment, pattern mining and optional node merging. Limitations of FiVaTech (Kayed & Chang, 2010) includes time-complexity associated with aligning DOM trees, usage of bias to determine peer nodes and the selection of proper bias influences the accuracy of the system.

Trinity (Sleiman & Corchuelo, 2014) considers document as a string of tokens. It applies Knuth Morris (Knuth, Jr, & Pratt, 1977) pattern matching algorithm to identify the common pattern among the documents. The initial shared pattern is referred as prefix, the portion of document which is not common is referred to as separator and the final pattern which is common among the documents is referred as suffix. The alignment is carried out recursively to build the trinity tree where each node has 3 child nodes. Template is deduced using Trinary tree. Limitations of Trinity (Sleiman & Corchuelo, 2014)are inability to handle template with alternating formatting for the same content, wrong deduction of template if same sequence of tokens are used as separator for different attributes and the time complexity associated with string alignment.

In (Janosi-Rancz & Lajos, 2015), authors have developed a semi-supervised approach for extraction based on custom developed extraction language R whose syntax is similar to CSS queries. This work served as a base for experimenting semantic analysis in the domain of web data extraction. In (Vela, Cavero, Caceres & Cuesta, 2019), authors have developed a semi-automatic extractor for scraping the data from websites having public transport information. Their notion is to develop a framework for processing and management of data related to public transport. In (Baskaran & Ramanujam, 2018), authors proposed an unsupervised approach based on Semantic analysis is used to extract the post records from Health Discussion Forum sites has been developed. The applicability of the approach to websites belonging to various domains is proved in this paper. Certain works such as (Dönz & Boley, 2014), (Dönz & Bruckne, 2013), (Furche, Gottlob, Grasso, Orsi, Schallhart, & Wang, 2012), (Pavai & Geetha, 2013) concentrate on web integration from multiple online sources.

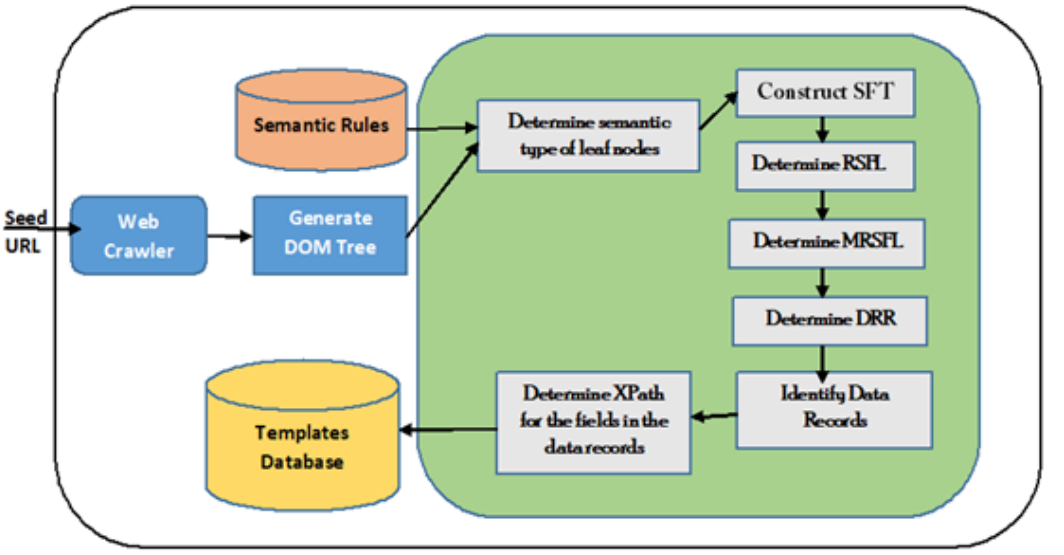
The proposed system departs from all these techniques in the following ways: Instead of considering entire document for the purpose of extraction, it first identifies the data rich region which significantly reduces the complexity involved in carrying out the extraction task. The technique does not consider the HTML code as such. It is based on the heuristics that attributes corresponding to data records will be present as leaf nodes in the DOM tree and data rich region has many such repetitive data records. Once the data rich region is identified we extract the XPath (https://www.w3.org/TR/1999/REC-xpath-19991116/) to each leaf node. It is used as template to extract the remaining data records. The technique has significantly reduced complexity and the experimental results show that it has good precision and recall values irrespective of missing attributes and different attributes formatted using same template.

DESIGN OF WEB DATA EXTRACTION SYSTEM (WDES) USING SEMANTIC LABELING

The architecture of WDES based on Semantic Labeling consists of following components: (i) conversion of web page to DOM tree, (ii) labeling of leaf nodes in the DOM tree using Semantic Rules database, (iii) determining List of Semantic Features (SFL) for non-leaf nodes, (iv) building Semantic Feature Tree (SFT), (v) identifying Repeated Semantic Features List (RSFL) for nodes having more than one child node by computing the similarity between List of Semantic Features using Tanimoto coefficient (Böhm, & Schneider, 2008), (vi) associating a measure of informativeness for nodes having Repeated Semantic Features List (RSFL), (vii) finding Maximum Repeated Semantic Features List (MRSFL) for non-leaf nodes with RSFL, by selecting RSFL of child node with maximum value for informativeness measure, (viii) Identifying data rich region by finding the lowest node in the Semantic Feature Tree whose Maximum Repeated Semantic Features List equals to Maximum Repeated Semantic Features List of root node, (ix) finding nodes representing data records by determining node whose SFL equals to MRSFL of Data Rich Region and determining XPath to each of the leaf nodes in the subtree represented by data record and storing in template database. The architecture is shown is Figure 1.

3.1 Steps in WDES Using Semantic Labeling

Figure 1. Architecture of WDES based on Semantic Labeling



a) Conversion to DOM tree:

The given webpage is converted to DOM tree (https://www.w3.org/DOM/DOMTR) using HTML parser provided by Jaunt API (https://jaunt-api.com/).

Figure 2 shows the HTML page and the corresponding DOM tree representation.

b) Determine type of leaf node using Semantic Rules Database:

The design of WDES using Semantic Labeling is based on the heuristic that, determining semantic type of leaf nodes and accumulating it to root help in identification of Data Rich Region. The content embedded in HTML template is available as leaf nodes in the DOM tree. It is easy for humans to know the type of content whereas for a computer it requires some logic for identification. The heuristics used by humans for identifying the type of content is represented as rules using regular expression for automatic identification. The rules used in identification of type of data fields is referred to as semantic rules. For example, the content embedded in heading elements h1 to h6 represents the title, the content embedded within block elements such as p or div element represents description, the content which matches regular expression for price represents cost and so on. The sematic rules used in identification of type of content for product web sites is shown in Table 1.

Figure 2. E-Bay Webpage and its DOM tree representation

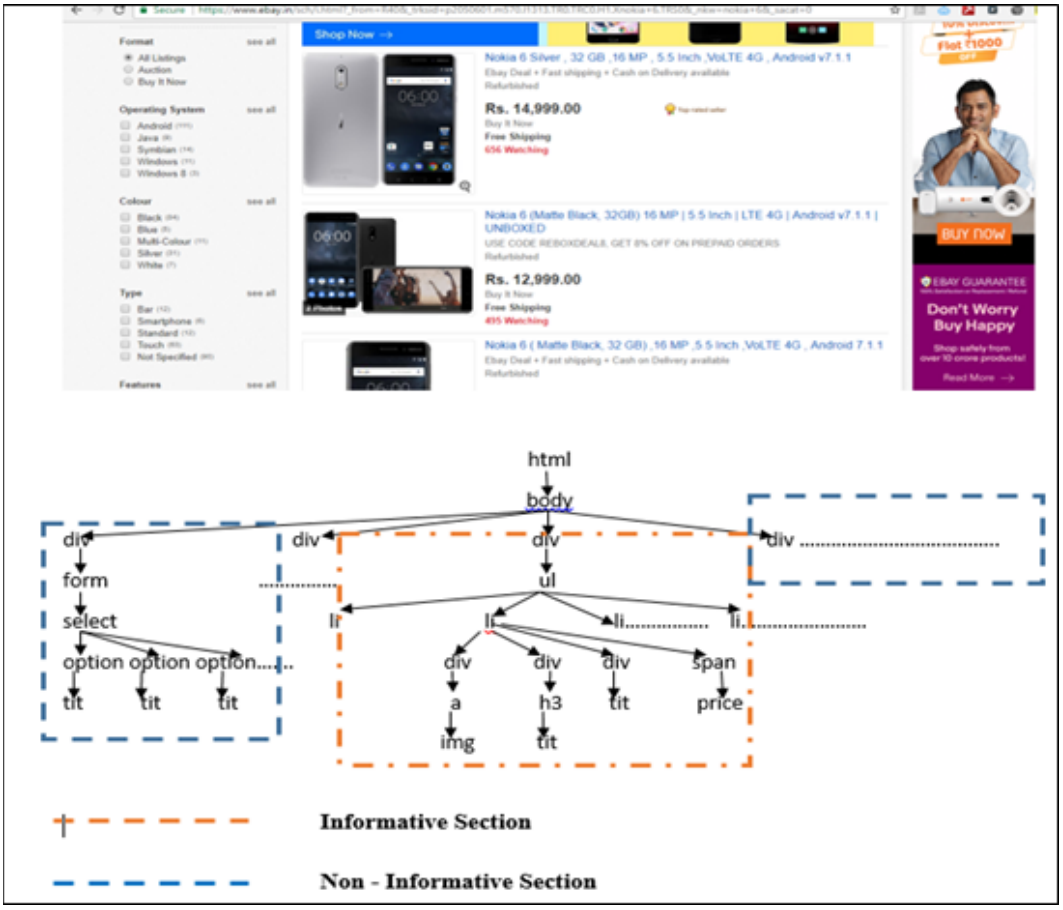


Table 1. Semantic Rules Database containing rules for detecting semantic type of Product Data

Alphabets	**Description**
block_ele	Identifies tags such as <td>, ,<p> etc.
(O)+	one or more occurrence
(O)*	zero or more occurrence
(O)1	one occurrence
(O)?	zero or one occurrence
Patterns	Description
if O is a child of <h#> or O[attr(class)].contains("title") then, assign node_type= $\mathbf{P_{title}}$	$\mathbf{P_{title}}$**:** Identifies title of the product Title: If node is child of heading elements (h1, h2, h3, h4,h5 or h6) and if its class attribute contains "title" then set node_type as title.
ifO.matches(\d[\d\,\.]+) then, node_type = $\mathbf{P_{price}}$	$\mathbf{P_{price}}$**:** Identifies price of the product If content of the node matches with the pattern corresponding to price then, assign node_type as price.
if ((W)+ && count(W) > 10 && W is a child of (block_ele) then, assign node_type= **Pcontent**	$\mathbf{P_{content}}$**:** Identifies description of the Product If content of the node has more than one word and is a child of block elements such as div or p then, set node_type as content.
if O.matches (\\d\\d\\%\\sOff$|\\d+\\%\\soff$	\\d+\\,\\d+\\s+\\(\\d+\\%\\) |\\(\\-\\d\\d\\%\\)) then, node_type=$\mathbf{P_{offer}}$	$\mathbf{P_{offer}}$**:** Identifies offer price of the Product If content of the node matches with the pattern corresponsding to offer price then, assign node_type as offer price.
if O.matches(img) then, node_type = P_{img}	If the node corresponds to <img> element then, set node_type as product image.

After generating DOM tree, the Semantic Feature Tree is built by applying the Semantic rules in Table 1 to the leaf nodes in the DOM tree of Figure 2.

c) Building Semantic Feature Tree (SFT)

Definition 1: Semantic Feature Tree (SFT) is a replication of DOM tree where every node is associated with List of Semantic Features (SFL), Repeated Semantic Features List (RSFL) and Maximum Repeated Semantic Features List (MRSL) and a measure of informativeness.

To obtain the Semantic Feature Tree from DOM Tree, the algorithm build_SemanticFeatureTree is invoked. The algorithm first traverses the DOM top down to identify the leaf nodes. As the nodes in the DOM tree are traversed, the corresponding node is created in the Semantic Feature Tree. Once leaf nodes are reached, they are matched with rules in the Semantic Rules Database and corresponding bit position is set to 1 in the List of Semantic Features associated with the node in the Semantic Feature Tree.

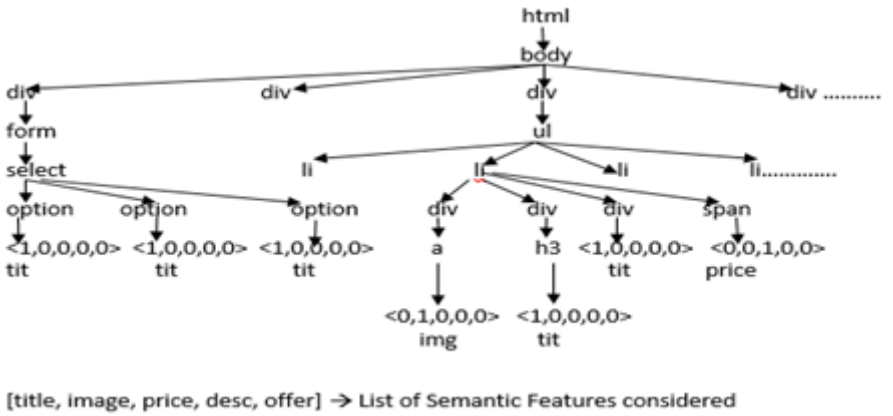
Applying Algorithm Build_SematicFeatureTree to the DOM tree in Figure. 2 results in Semantic Feature Tree shown in Figure 3.

Definition 2: List of Semantic Features(SFL) is a vector of bit strings denoted by SFL ($node_m$) = {v_1, v_2, v_3... v_k}, where k is the number of labels associated with the leaf nodes in the Semantic Feature Tree. SFL (nodem)[i]= 1, if the content of leaf node matches the $rule_i$ in Semantic Rules Database. Set to 0, otherwise.

Algorithm Build_SemanticFeatureTree (root_of_DOMTree root, root_of_SFT n)

Input: URL of the Forum page
Output: Semantic Feature Tree
1. begin
2. if(root.type=ELEMENT_NODE) &&root.childNodes.size() >0)
3. Check whether child nodes are text nodes
4. Match the concatenated string against regular expression to determine semantic_type
5. Set the corresponding value in the Semantic Feature List
6. end if
7. if(root.type=ELEMENT_NODE &&root.childNodes.size() !=0)
8. for every child node in DOMTree
9. construct the node in SFT
10. build_SemanticFeatureTree(root.child)
11. end for
12. end if
13. end

Figure 3. Semantic Feature Tree whose leaf-nodes are associated with SFL



Once the Semantic Feature Tree is built, the next step is to find the Semantic Features List of non-leaf nodes which is done by invoking the algorithm Compute_SFL_Non_Leaf_Nodes. The algorithm performs a post-order traversal of Semantic Feature Tree in whichSFL(non-leaf_nodem)= sum(SFL(childNodes(non-leaf_nodem)), where n is the number of child nodes of m

Applying the algorithm Compute_SFL_Non_Leaf_nodes to the SFT in Figure 3 results in SFT as shown in Figure 4.

Definition 3: Repeated Semantic Features List(RSFL) of a $node_m$ in Semantic Feature Tree (SFT) is avg(SFLs of childNodes($node_m$)) such that sim(SFL($node_i$),SFL($node_j$)) > 0.75 where $node_i$, $node_j$ belongs to childNodes($node_m$). 0.75 is chosen as threshold through experiments. By varying threshold between 0.5 and 0.8, it is observed that accurate results is obtained for threshold = 0.75.

Algorithm Compute_SFL_Non_Leaf_Nodes(root_of_SFT root)

Input: root node of SFT
Output: Semantic Feature Tree with nodes having SFL
1. begin
2. for every child node of root do
3. Compute_SFL_Non_Leaf_Nodes(root.childnode)
4. end for
5. for i<- 1 to root.childNodes.size() do
6. root.SFL[i] +=root.child.SFL[i]
7. end for
8. end

Similarity between lists of Semantic Features is determined by using Tanimoto Coefficient (Böhm & Schneider, 2008).

Where, a_i is the j^{th} element in Vector A and b_j is the j^{th} element in Vector B and k is the size of the vectors A and B respectively.

Definition 4: Informativeness Measure associated with RSFL($node_m$) is derived by taking into account the heuristics that informative section contains more number of relevant data fields and hence the number of non-zero values in the Semantic Features. List will be high and number of times the SFL is repeated will also be high. Therefore,

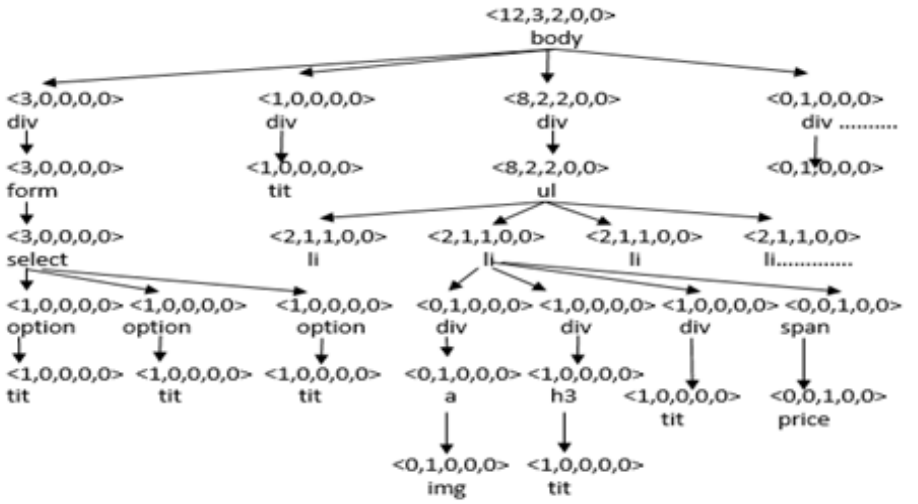
Information value (RSFL) =,

Where,

p - size of the list,

v_i – i^{th} value in the RSFL(v),

Figure 4. Semantic Feature Tree in which nodes are associated with SFL



[title, image, price, desc, offer] → List of Semantic Features considered

n- number of non-zero entries in the feature list and
r – number of times RSFL is repeated.

Algorithm calculate_RSFL performs post order traversal of Semantic Feature Tree. Lines 6, 7, 8 and compares the SFL of every pair of child nodes of the current node and finds the SFLs whose similarity is greater than the threshold. Line 20 determines the informativeness measure associated

Algorithm calculate_RSFL(root_of_SFT root)

Input: root node of SFT
Output: Semantic Feature Tree with nodes having RSFL
1. begin
2. for every child node of root do
3. calculate_RSFL(root.childnode)
4. end for
5. if (root.child.length>1)
6. for i<- 1 to root.childNodes.size() do
7. for j<-i+1 to root.childNodes.size() do
8. if(sim(root.childNodes[i],root.childNodes[j])>0.75)
9. repeatcnt++
10. store the average of SFL of node i& j as RSFL
11. end if
12. end for
13. end for
14. end if
15. store the repeat count
16. for each feature in RSFL
17. if(feature[i]>0)
18. non-zero_cnt++
19. end if
20. end for
21. for each feature in RSFL
22. if(feature[i]>0)
23. info_val = info_val+feature[i]*repeat_cnt*non_zero_cnt
24. end if
25. end for
26. end

Algorithm calculate_MRSFL(root_of_SFT root)

Input: root node of SFT
Output: Semantic Feature Tree with nodes having MRSFL
1. begin
2. for every child node of root do
3. calculate_MRSFL(root.childnode)
4. end for
5. for i<- 1 to root.childNodes.size() do //find child with max RSFL
6. if(root.childNodes[i].info_val>max)
7. max<-root.childNodes[i].info_val
8. max_RSFL<-root.childNodes[i].MRSFL
9. end if
10. end for
11. if(root.info_val>max)
12. MRSFL<-root.RSFL
13. else
14. MRSFL<-max_RSFL
15. end if
16. end

with the RSFL by taking product of each in the list with the number of times repeated and number of non-zero values.

By applying the algorithm to the SFT in Figure. 4, we get the RSFLs for the following nodes:

<select> elements RSFL: <select> element has many <option> elements as its child nodes (let us assume 10). SFL associated with <option> element is <1,0,0,0,0> and the similarity between any two SFLs is, since all are same. Therefore, RSFL of <select> element is <1,0,0,0,0>, the number of times it is repeated is 10. Information measure of RSFL <1,0,0,0,0> is [1X10X1+0X10X1+0X 10X1+0X10X1+0X10X1] = 10.

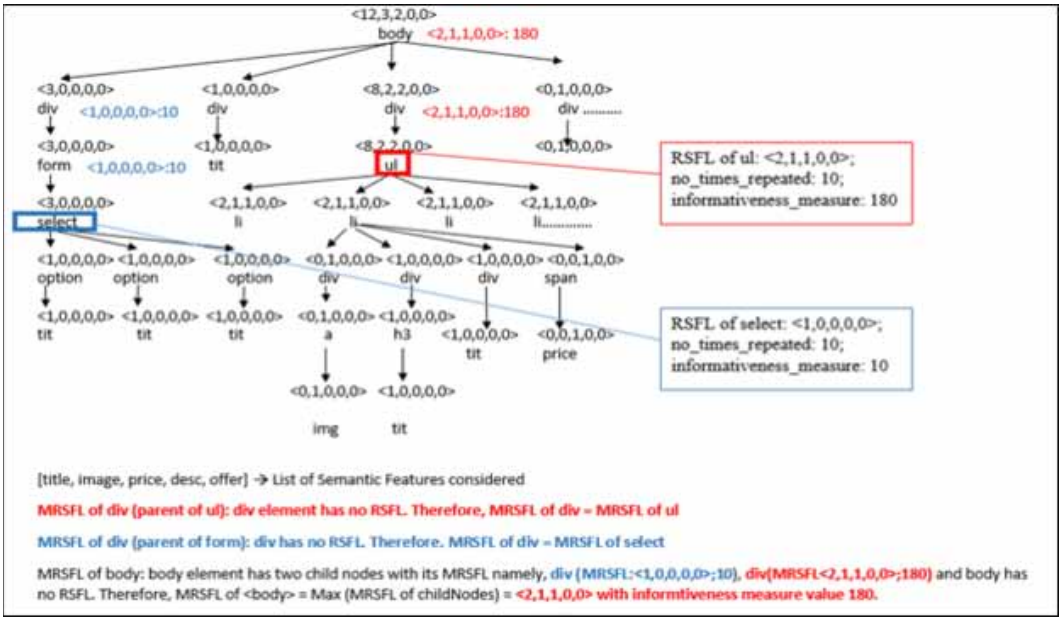
 elements RSFL: <ul> element has many <li> elements as its child nodes (based on number of data records). Let us assume it has 10 <li> elements as its child nodes. SFL associated with li element is <2,1,1,0,0> and the number of times it is repeated equals to 10. Number of non-zero values in the RSFL is 3. Even if any data record has missing attribute, the technique considers its SFL since sim >threshold(0.75). Informativeness measure of RSFL <2,1,1,0,0> is [2X10X3 + 1X10X3 + 1X10X3 + 0X10X3 + 0X10X3] = 180.

Definition 5: Maximum Repeated Semantic Features List (MRSFL) of a $node_m$ in the Semantic Feature Tree (SFT) is determined by

Max {info_value(RSFL(nodem)), info_value(MRSFL(childNodes(nodem)))}

Algorithm calculate_MRSFL performs a post order traversal of Semantic Feature Tree and determines the MRSFL of the current node by comparing the RSFL of current node and MRSFL of leaf nodes and choosing the one with the maximum informativeness measure. Lines 5, 6, 7, 8 and 9 finds the MRSFL with maximum informativeness measure. Lines 10,11,12,13 compares MRSFL with maximum informativess measure with the informativeness measure associated with current nodes RSFL and assigns the MRSFL of current node accordingly.

Figure 5. Determine MRSFL of root for the SFT



Algorithm find_DRR(root_of_SFT root)

Input: Root node of SFT
Output: Node with MRSFL same as that of root's MRSFL
begin
for every child node of root do
for j <- 0 to num_features do
if(child.MRSFL[j] = root.MRSFL[j])
cnt++;
end if
end for
end for
if(cnt == num_features)
flag=1
find_DRR(root.childnode)
else
if (flag == 1)
return root//node representing DRR
end if
end

Algorithm find_DataRecords (drr_node)

Input: Node corresponding to DRR of SFT
Output: Node with SFL same as that of drr_node's MRSFL
begin
for every child node of drr_node do
for j <- 0 to num_features do
if(child.SFL[j] = drr_node.MRSFL[j])
cnt++;
end if
end for
if(cnt = num_features)
find_XPATH(child)
end if
end for
end

Upon applying calculate_MRSFL to the SFT in Figure 4, <select> element's MRSFL equals its RSFL, <ul> element's MRSFL equals to its RSFL. From Figure 5, it is clear that <body> element get its MRSFL from <ul> element since the informativeness measure associated with its MRSFL is high.

Finding Data Rich Region

Definition 6: Data Rich Region consists of Repeated Data Records whose Maximum Repeated Semantic Features List (MRSFL) is same as Maximum Repeated Semantic Features List (MRSFL) of root.

Since the MRSFL is propagated from leaf nodes to root based on informativeness measure, the root gets its MRSFL from the node containing data records which has the maximum information value compared to other nodes in the DOM tree. Therefore, the algorithm find_DRR determines the lowest node whose MRSFL is same as root node's MRSFL.

Lines 3, 4 and 5 compares the MRSFL of the current node with the MRSFL of root. The variable cnt keeps track of number of bits that matches in both the vectors. Line 8 checks whether the count matches with the length of the vector. If it is true then, find_DRR is recursively called on its child node. When the recursion is exited, the last node whose MRSFL is same as root's MRSFL represents the DRR.

From Figure 5, it is clear that body element gets its MRSFL from ul element. The lowest element whose MRSFL is same as the MRSFL of root is ul element. Therefore, ul element represents the Data Rich Region.

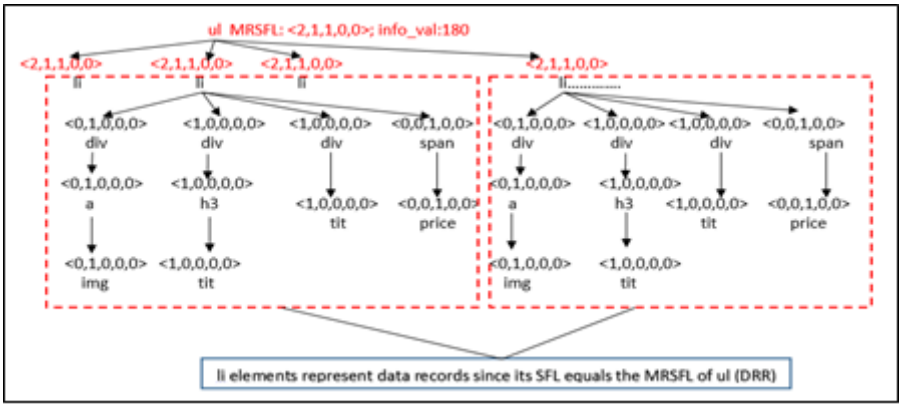
a)

Finding Data Records

Definition 7: Data Rich Region gets its MRSFL from the node representing data records. Since the data records contain repeated pattern, the SFL gets repeated which resulted in MRSFL of DRR.

The algorithm find_DataRecords finds the node whose SFL equals MRSFL of DRR. Lines 3, 4, 5 and 6 determines number of bits whose SFL matches with the MRSFL. If the number of bits equals

Figure 6. Identify data records



length of the vector then, the current node represents data record. The XPath to each field within the data record is determined and stored in template database. The XPath in template database is used to extract data records from similarly structured pages.

By applying the algorithm for the SFT in Figure 5, each li element who's SFL equals the MRSFL of DRR represents the data record as shown Figure 6. We get the XPaths shown in Table 2 by finding the path to each of the leaf nodes from the li element. The XPaths are stored in template database and used to retrieve the data records from similarly structured pages belonging to the same website.

EXPERIMENTAL ANALYSIS

Table 2. XPath to data fields within the data record

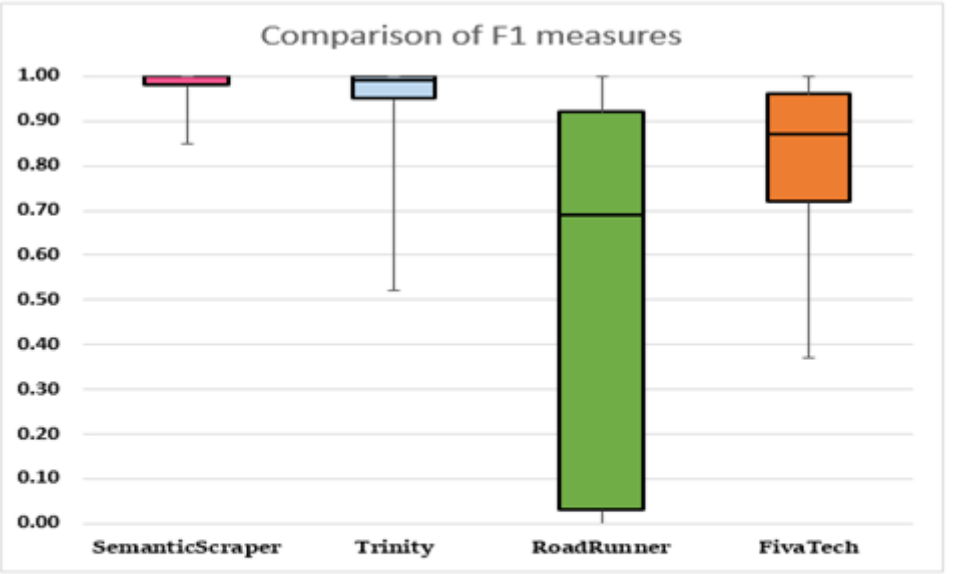
Data Field	XPath
Img	body/div/ul/li/div/a/img
Title	body/div/ul/li/div/h3/tit and body/div/ul/li/div/tit
price	body/div/ul/li/span/price

The system is implemented using JDK 1.7 using Netbeans IDE. Jaunt API (https://jaunt-api.com/) is used for parsing HTML documents and for the construction of DOM tree. Experiment has been carried out in 31 real-world datasets belonging to 7 different domains. The system is able to identify Data Rich Region with 100% accuracy provided at least two fields belonging to the data records satisfy the rules in the Semantic Rules Database. Otherwise, the semantic rules need to be updated to recognize unseen pattern. Semantic Scraper is compared with other state-of-the-art techniques using the measures: recall and precision. Recall value is the fraction of number of data records extracted correctly over the actual number of data records present in the document. Precision value is the fraction of number of records extracted correctly over the number of data records extracted. From the experimental results in Table 3, it is clear that semantic scrapers outperform the techniques based on string pattern matching (Arasu & Garcia-Molina, 2003) (Crescenzi, Mecca, & Merialdo, 2002) and DOM tree matching (Kayed & Chang, 2010) in the following aspects:

Table 3. Experimental results

	Category	No. of Pages	Semantic Scraper			Trinity			RoadRunner			FivaTech		
			P	R	F1	P	R	F1	P	R	F1	P	R	F1
Books	Aloe Books	30	1.00	1.00	1.00	1.00	1.00	1.00	0.65	0.58	0.61	0.92	0.99	0.95
	Many Books	30	1.00	1.00	1.00	0.99	0.99	0.99	0.97	0.95	0.96	0.77	0.97	0.86
	Awesome Books	30	1.00	1.00	1.00	1.00	0.87	0.93	0.78	0.53	0.63	0.85	1.00	0.92
Movies	IMDB	30	0.99	1.00	0.99	0.93	0.86	0.89	0.39	0.35	0.37	-	-	-
	Disney Movies	30	0.98	0.99	0.98	1.00	1.00	1.00	0.67	0.67	0.67	0.71	0.67	0.69
	Albania Movies	30	0.99	0.99	0.99	0.95	0.98	0.96	0.75	0.77	0.75	0.82	0.81	0.81
Cars	Auto Trader	30	1.00	1.00	1.00	0.99	1.00	1.00	-	-	-	-	-	-
	Car Max	30	1.00	1.00	1.00	1.00	1.00	1.00	0.98	0.98	0.98	0.45	0.89	0.60
	Car Zone	30	0.99	0.98	0.98	0.98	1.00	0.99	0.72	0.81	0.76	0.92	1.00	0.96
Jobs	4 Jobs4	30	0.97	0.99	0.98	0.92	0.98	0.95	0.00	0.00	0.00	-	-	-
	Career Builder	30	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	0.00	0.80	0.83	0.82
	Job of Mine	30	0.80	0.90	0.85	0.86	1.00	0.93	0.72	0.72	0.72	-	-	-
Real Estate	Trulia	30	1.00	1.00	1.00	0.63	1.00	0.77	0.00	0.00	0.00	-	-	-
	Remax	30	1.00	1.00	1.00	0.70	0.98	0.82	0.00	0.00	0.00	-	-	-
	Haart	30	1.00	1.00	1.00	1.00	1.00	1.00	0.79	0.79	0.79	0.94	1.00	0.97
Sports	SoccerBase	30	1.00	1.00	1.00	0.97	1.00	0.98	0.95	1.00	0.97	-	-	-
	UEFA	30	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	-	-	-
	NFL	30	0.90	0.92	0.91	1.00	1.00	1.00	0.98	0.98	0.98	0.53	0.81	0.64
Doctors	WebMD	30	1.00	1.00	1.00	1.00	1.00	1.00	0.06	0.06	0.06	0.77	1.00	0.87
	AMA	30	1.00	1.00	1.00	0.98	1.00	0.99	-	-	-	-	-	-
	Dentists	30	0.97	0.98	0.97	0.92	1.00	0.96	0.96	0.96	0.96	0.56	0.99	0.72
RoadRunner	Amazon	21	1.00	1.00	1.00	0.93	0.73	0.82	0.00	0.00	0.00	0.60	0.67	0.63
	UEFA	20	1.00	0.99	0.99	1.00	0.90	0.95	1.00	1.00	1.00	0.91	0.94	0.92
	E-Bay	19	0.97	1.00	0.98	0.97	1.00	0.98	0.82	0.82	0.82	0.83	1.00	0.91
	Netflix	50	1.00	1.00	1.00	0.99	0.99	0.99	0.76	0.79	0.78	0.82	0.80	0.81
	Major League	9	0.99	0.98	0.98	0.98	0.55	0.70	0.00	0.00	0.00	0.99	1.00	0.99
RISE	Bigbook	235	0.99	0.99	0.99	0.95	0.94	0.94	0.01	0.00	0.00	-	-	-
	IAF	252	1.00	1.00	1.00	0.84	0.38	0.52	0.90	0.09	0.17	0.53	0.69	0.60
	Okra	10	1.00	0.97	0.98	1.00	0.82	0.90	0.12	0.01	0.02	0.49	0.34	0.40
	LA Weekly	28	0.99	0.99	0.99	0.97	0.92	0.94	0.00	0.00	0.00	0.83	0.57	0.68
	Zagat	91	1.00	1.00	1.00	1.00	0.86	0.92	0.00	0.00	0.00	1.00	0.98	0.99

Figure 7. Box Plot of F1-Measures



	SemanticScraper	Trinity	RoadRunner	FivaTech
Minimum	0.85	0.52	0.00	0.37
Quartile 1	0.98	0.95	0.03	0.72
Median	1.00	0.99	0.69	0.87
Quartile 3	1.00	1.00	0.92	0.96
Maximum	1.00	1.00	1.00	1.00

a. Performs automatic labeling of extracted instances based on semantic rules
b. Ability to perform extraction even if a single input page is available
c. The pattern matching based techniques won't be able to perform extraction if the web page has missing attributes or if attributes are formatted using alternating templates. Semantic Scraper has no dependency on templates used to format data records and therefore, it is able to perform well even if some data records has missing attributes or differently formatted attributes.
d. The complexity of pattern based technique is high, exponential in case of RoadRunner (Crescenzi, Mecca, & Merialdo, 2002).

Complexity of Semantic Scraper has two major components: construction of Semantic Feature Tree, determining Repeated Semantic Features List. Construction of SFT requires in-order traversal of DOM tree whose complexity is O(n) where n is the number of nodes in the DOM tree. To determine Semantic Features List, Semantic Feature Tree is traversed in post-order and similarity between Semantic Feature List of child nodes are computed. Let n be the number of nodes in the Semantic Feature Tree, let m be the number of child nodes for each n in SFT and $m<n/4$, k be the size of semantic feature vector. For each pair of child nodes of node n, similarity computation requires k^2 iterations. Number of comparisons equals $n*m(m-1)*k^2$. Since the number of features k considered is constant usually less than 20, k^2 is negligible. Also, $m(m-1)<n$ in a DOM tree, since number of child nodes, a node has is very less compared to total number of nodes in the tree. Therefore, the complexity reduces to $O(n^2)$.

The techniques are compared based on F1 score which is the weighted harmonic mean of precision and recall values. The box plot in Figure 7 shows the comparison of techniques by taking into account

the entire range of values by considering min, max, Q1, median and Q3. The maximum value for F1 score is 1.00 for all the techniques since they were able to perform 100% extraction for certain cases. Semantic Scraper departs from the rest of the techniques with respect to its interquartile range which is the difference between Q3 and Q1 is the least proving the consistency of the approach compared to all other state-of-the-art approaches.

CONCLUSION AND FUTURE DIRECTIONS

Most of the unsupervised algorithms like ExAlg (Arasu & Garcia-Molina, 2003), RoadRunner (Crescenzi, Mecca, & Merialdo, 2002), FivaTech (Kayed & Chang, 2010) etc. try to learn template first, which is then used to carry out the extraction process. The drawbacks associated with these approaches are their dependency on string matching or tree matching makes them computationally expensive, inability to perform extraction if only a single source page is available, missing attributes, use of same template for formatting different attributes or use of alternate formatting for same attribute remarkably degrades the accuracy of extraction. From the extensive study of these approaches it is clear that all these drawbacks are associated with template deduction. Semantic Scraper approach is based on the heuristic that attribute values corresponding to data records are available as leaf nodes in the DOM tree and by scoring the leaf nodes representing relevant content and accumulating the score to the non-leaf nodes helps in identification of data rich region. A single post order traversal of the Semantic Feature Tree identifies the Data Rich Region which contains the target data records and thus computational complexity gets reduced significantly. This approach is capable of performing extraction even if single source page is available. Since the approach is template independent and dependent only on the repetition of data records, it performs extraction irrespective of formatting of attributes and missing attributes. It is proved from the experiments the versatility of the proposed approach i.e. its applicability to web sites belonging to various domains.

The limitation of SemanticScraper is its inability to classify leaf nodes to appropriate semantic type if it is not covered by the semantic rules. The accuracy of the system depends heavily on the semantic rules which is used to classify the semantic type of leaf nodes. Framing Semantic rules require thorough domain knowledge. In future, the work can be extended to include domain ontologies for framing semantic rules and represent the extracted data using semantic representations such as RDF (https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/Overview.html), JSON (https://www.json.org/) etc. to facilitate analysis.

REFERENCES

Arasu, A., & Garcia-Molina, H. (2003). Extracting structured data from web pages. In *Proc. ACM SIGMOD* (p. 337-348). ACM.

Baskaran, U., & Ramanujam, K. (2017). Web harvesting: web data extraction techniques for deep web pages. In A. Kumar (Ed.), *Web usage mining techniques and applications across industries* (pp. 351–378). IGI Global.

Baskaran, U., & Ramanujam, K. (2018). Automated scraping of structured data records from health discussion forums using semantic analysis. In *Informatics in Medicine Unlocked*. Elsevier. .10.1016/j.imu.2018.01.003

Baumgartner, R., Gatterbauer, W., & Gottlob, G. (2009). Web data extraction system. In Encyclopedia of database systems (pp. 3465 – 3471). doi:10.1007/978-0-387-39940-9_1154

Böhm, H. J., & Schneider, G. (2008). *Virtual Screening for Bioactive Molecules*. Retrieved from https://pubs.acs.org/doi/abs/10.1021/ja0152052

Chang, C. H., Kayed, M., Girgis, M. R., & Shaalan, K. A. (2006). Survey of web information extraction systems. *IEEE Transactions on Knowledge and Data Engineering*, *18*(10), 1411–1428. doi:10.1109/TKDE.2006.152

Chang, C. H., & Lui, S. C. (2001). IEPAD: information extraction based on pattern discovery. *Proceedings of the Tenth International Conference on World Wide Web (WWW).* doi:10.1145/371920.372182

Crescenzi, V., Mecca, G., & Merialdo, P. (2002). Roadrunner: automatic data extraction from data-intensive websites. SIGMOD. doi:10.1145/564691.564778

Dönz, B., & Boley, H. (2014). Extracting data from the deep web with global-as-view mediators using rule enriched semantic annotations. In *Proceedings of the RuleML 2014 Challenge and the RuleML 2014 Doctoral Consortium hosted by the 8th International Web Rule Symposium* (*Vol. 1211*, pp. 1-15). Academic Press.

Dönz, B., & Bruckne, D. (2013). Extracting and integrating structured information from web databases using rule based semantic annotations. *Industrial Electronics Society IECON 2013–39th Annual Conference of the IEEE*, 4470-4475. doi:10.1109/IECON.2013.6699855

Furche, A., Gottlob, T., Grasso, G., Orsi, G., Schallhart, G., & Wang, C. (2012). *AMBER: Automatic supervision for multi-attribute extraction*. CoRR abs/1210.5984 2012.

Hogue, A., & Karger, D. (2005). Thresher: automating the unwrapping of semantic content from the world wide web. *Proceedings of the 14th International Conference on World Wide Web (WWW)*. doi:10.1145/1060745.1060762

Hsu, C. N., & Dung, M. (1998). Generating finite-state transducers for semi-structured data extraction from the web. *Journal of Information Systems*, *23*(8), 521–538. doi:10.1016/S0306-4379(98)00027-1

Janosi-Rancz, K. T., & Lajos, A. (2015). Semantic data extraction. Elsevier Procedia Technology, 19, 827–834. doi:10.1016/j.protcy.2015.02.119

Kayed, M., & Chang, C. H. (2010). FiVaTech: Page-level web data extraction from template pages. *IEEE Transactions on Knowledge and Data Engineering*, *22*(2), 249–263. doi:10.1109/TKDE.2009.82

Knuth, D. E., Jr. J. H. M., & Pratt, V. R. (1977). Fast pattern matching in strings. SIAM J. Comput., 6(2), 323–350.

Kushmerick, N., Weld, D., & Doorenbos, R. (1997). Wrapper Induction for Information Extraction. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence* (pp. 729-735). Nagoya, Japan: Academic Press.

Laender, A. H. F., Ribeiro-Neto, B. A., da Silva, A. S., & Teixeira, J. S. (2002). A brief survey of web data extraction tools. *SIGMOD Record*, *31*(2), 84–92. doi:10.1145/565117.565137

Muslea, I., Minton, S., & Knoblock, C. (1998). A hierarchical approach to wrapper induction. *Proceedings of the Third International Conference on Autonomous Agents (AA-99)*. doi:10.1145/301136.301191

Pavai, G., & Geetha, T. V. (2013). A unified architecture for surfacing the content of deep web databases. *Proc. of Int. Conf. on Advances in Communication, Network, and Computing*, 35 – 38.

Sleiman, H. A., & Corchuelo, R. (2013). A survey of region extractors from web documents. *IEEE Transactions on Knowledge and Data Engineering*, *25*(9), 1960–1981. doi:10.1109/TKDE.2012.135

Sleiman, H. A., & Corchuelo, R. (2014). Trinity: On using trinary trees for unsupervised web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, *26*(6), 1544–1556. doi:10.1109/TKDE.2013.161

Vela, B., Cavero, J. M., Cáceres, P., & Cuesta, C. E. (2019). A Semi-Automatic Data–Scraping Method for the Public Transport Domain. *IEEE Access: Practical Innovations, Open Solutions*, *7*, 105627–105637. doi:10.1109/ACCESS.2019.2932197

Umamageswari Kumaresan has received Ph.D. in the area of Computer Science and Engineering from Pondicherry University in April 2019. She was born in June 1984 at Puducherry. She received her B.Tech. in Computer Science and Engineering from Pondicherry Engineering College in 2005, her M.Tech. in Computer Science and Engineering from Bharath University in 2012. She worked as Senior Software Engineer in Wipro Technologies from August 2005 till January 2007. She worked as Assistant Professor in the Department of Information Technology at New Prince Shri Bhavani College of Engineering and Technology from June 2012 till April 2018. She has published more than ten research papers in reputed International Journals and Conferences. Her research interests include web mining, web data extraction, information security and sentiment analysis. She is a member of IAENG.

Kalpana Ramanujam is currently working as Professor in the Department of Computer Science and Engineering at Pondicherry Engineering College, Puducherry, India. She received her B.Tech. degree in Computer Science and Engineering from Pondicherry University, Puducherry, India in the year 1996 and M. Tech. degree in Computer Science and Engineering from Pondicherry University, Puducherry in1998. She completed her Ph.D in Computer Science & Engineering in the year 2013 in the field of Parallel Computing Systems. Her areas of interest include Parallel Computing Systems, High Performance Computing, Web services and Distributed Computing. She has published more than 100 research papers in International Journals / Conferences. She is also a member of ISTE. She is also a recipient of CMI awardee for technical leadership.