Web Scraper Using Selectors

Ms. Rashmi Bhosale, Prof. Prashant Jawalkar

Student, Computer Department, JSPM's BSIOTR, Maharashtra, India
Associate Professor, Computer Department, JSPM's BSIOTR, Maharashtra, India

ABSTRACT

Today enormous amount of valuable information is available on the net and it is growing at very fast rate. Thus we can call the web as chief knowledge store house. Web scraper is defined as a system that automatically extracts information from websites. Web data analysis applications requires web data extraction. Early techniques visits the interested web sites and constructs the wrapper then extracts the data using that wrapper but this process needs a lot of time. Thus a new technique known as Web Scraper Using Selectors (WSUS) is introduced. WSUS constructs the patterns for selected data sections automatically and extracts data using patterns. Pattern generated by WSUS is relative rather than absolute which makes it stable. It extracts data without comparing the DOM trees.

Keyword : - *Bad tag; Basic block; DOM tree; Web scraper, Wrapper*

1. Introduction

A system that automatically extracts information from websites is called as Web Scraper. Web scraping is also called as web data extraction or web harvesting. Day by day, the volume of information available on the net is growing very rapidly, thus we can call the web as chief knowledge store house. This information is very useful and it can be used in various applications

Web sites provides us huge amount of information about various topics in different formats. The task of manually locating the data of interest and then extracting it from the web sites needs huge efforts. To extract the data, existing techniques builds programs called as "extractor" or "wrapper" or "crawler". These wrappers have some knowledge about the structure of web pages. The web pages belonging to the same web site have common structure or template. Templates are often fixed and data values are changing across the web pages. It is observed that structure of web pages changes often and maintaining the crawler can be impractical and expensive

Thus we have proposed a new technique called as Web Scraper Using Selectors(WSUS). It works in three stages: Web page renderer, section selector pattern generator and data extractor.

The rest of the paper is structured as follows: Section 2 explains existing techniques. Section 3 presents proposed system architecture in details. Section 4 describes experimental setup. Section 5 compares proposed system with existing system. Section 6 concludes the paper.

2. Existing Systems

IEPAD[14] is an automatic data extraction system. It works in three phases, an extraction rule generator, pattern viewer and extractor . Extraction rule generator accepts input web page from user. Pattern viewer discovers repetitive patterns. Extractor module **e**xtracts data from similar pages according to the extraction rule provided by the user.

ViDE[3] extracts data records from deep web pages using visual information . It works in four steps . First, it obtains visual representation of given web page and convert this representation into a tree called as Visual Block tree. In second step, it extracts data records from the VB tree. In the third step, it divides scraped data records into various data items. In the fourth step, it generates set of visual extraction rules.

ROADRUNNER[15] is a page-level data extraction system. It uses template of first input web page as a initial wrapper and checks whether next pages can be created by using the initial wrapper or not. If not, then it will modify initial template accordingly.

EXLAG[11] consists of two phases, Equivalence Class Generation Module(ECGM) and analysis phase. In the ECGM phase, for given input web pages it obtains equivalence classes and finds out reoccurring equivalence classes. In the analysis phase, from large and repeatedly occurring equivalence classes it generates template and then using that template it will extract data.

DELA[10] generates wrapper in two steps. In the first step, it compares DOM trees of two web pages and finds out data-rich sections web pages. In the second step, it discovers repeated patterns by using suffix trees.

DEPTA[7] finds out repeated substrings in the tag tree by comparing adjacent substrings having same parent.

FiVaTech [2] extracts data and detects schema in two phases namely tree merging and schema detection. In the first phase, it will merge DOM trees of all input web pages and constructs fixed/variant pattern tree. In the second phase this tree is used to identify the schema and template.

3. Proposed System

The proposed system WSUS consists of three important components. First component is a web page renderer, which accepts an URL of the page from user and displays it in the browser. After displaying the page it will constructs the DOM tree of the page. Second component is a section selector which will partition the web page into various data regions. User can then easily select the data section. Third component is pattern generator and data extractor which will generate the patterns for selected data sections then data extractor will use those patterns to extract data. Fig 1 shows various components of WSUS.

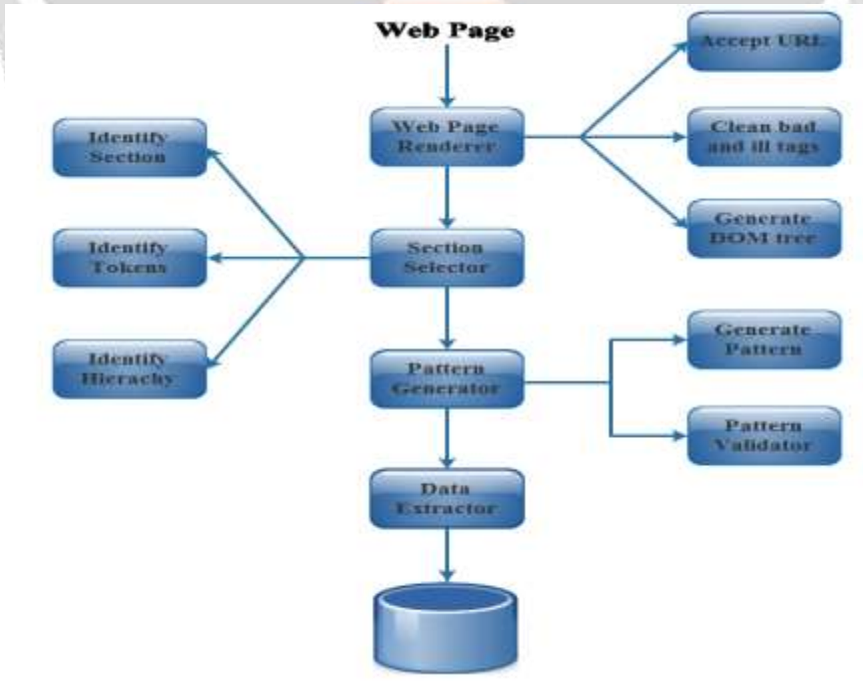


Fig -1 Components of WSUS

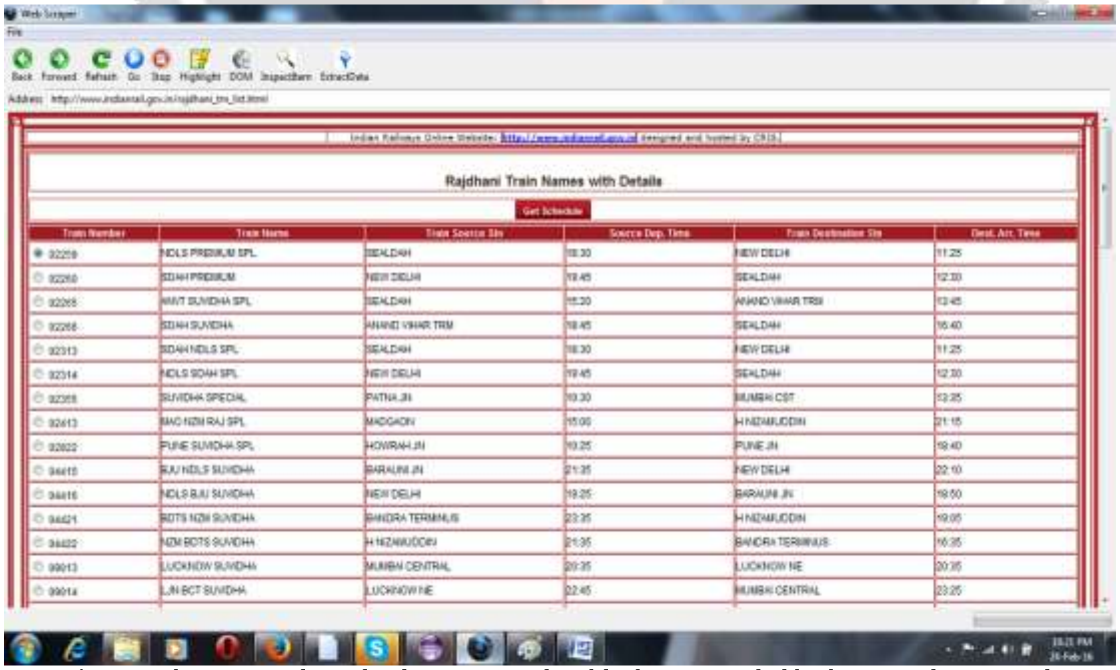
3.1 Web Page Renderer

It performs following three tasks

- URL acceptor
  It will accept the URL from user and displays the requested page in the browser. Rendering is the process of drawing a rectangular box around each visible node on the web page. Each box is called as visual block. One block may contain many inner blocks. The block which does not contain any inner block is called as basic block and may hold data value. Basic blocks are represented by brown color border. Fig 2 shows various data blocks.

- Repair ill or bad tags
  Once the document is retrieved, it will start repairing the bad or ill formatted tags. This process find out missing tags and insert those missing tags and it will remove unnecessary tags. For example !pr tag which is the start tag in document but it is the end tag that does not have a start tag. It also checks nesting of all tags.

- DOM tree generator
  DOM tree is generated after repairing all ill or bad tags. Each html element has end tag, optional attributes, start tag and optional embedded content. Each web page consists of zero or one document type node, zero or more comments and one root element node. Syntactic tokens are then generated using the token tree. Fig 3 shows sample code and corresponding DOM tree.



Address: http://www.indianrail.gov.in/rajdhani_trn_list.html

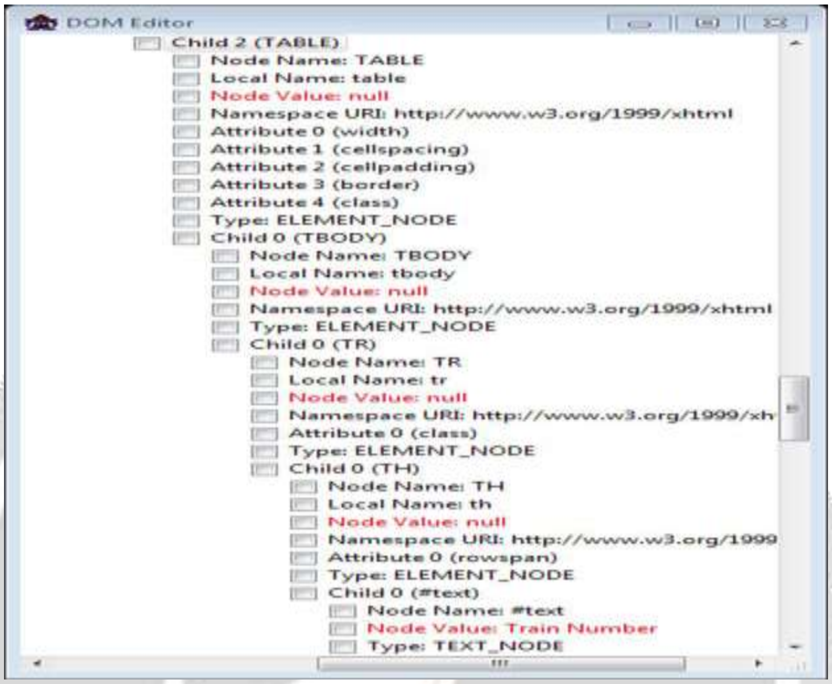
Indian Railways Online Website: http://www.indianrail.gov.in designed and hosted by CRIS.

Rajdhani Train Names with Details

Train Number	Train Name	Train Source Stn	Source Dep. Time	Train Destination Stn	Dest. Arr. Time
02259	NDLS PREMIUM SPL	SEALDAH	18:30	NEW DELHI	11:25
02260	SDAH PREMIUM	NEW DELHI	19:45	SEALDAH	12:30
02265	ANVT SUVIDHA SPL	SEALDAH	15:20	ANAND VIHAR TRM	13:45
02266	SDAH SUVIDHA	ANAND VIHAR TRM	18:45	SEALDAH	16:40
02313	SDAH NDLS SPL	SEALDAH	18:30	NEW DELHI	11:25
02314	NDLS SDAH SPL	NEW DELHI	19:45	SEALDAH	12:30
02358	SUVIDHA SPECIAL	PATNA JN	10:30	MUMBAI CST	13:35
02413	MAO NZM RAJ SPL	MADGAON	15:00	H NIZAMUDDIN	21:15
02822	PUNE SUVIDHA SPL	HOWRAH JN	10:25	PUNE JN	19:40
04415	BJU NDLS SUVIDHA	BARAUNI JN	21:35	NEW DELHI	22:10
04416	NDLS BJU SUVIDHA	NEW DELHI	19:25	BARAUNI JN	19:50
04421	BDTS NZM SUVIDHA	BANDRA TERMINUS	23:35	H NIZAMUDDIN	19:05
04422	NZM BDTS SUVIDHA	H NIZAMUDDIN	21:35	BANDRA TERMINUS	16:35
09013	LUCKNOW SUVIDHA	MUMBAI CENTRAL	20:35	LUCKNOW NE	20:35
09014	LJN BCT SUVIDHA	LUCKNOW NE	22:45	MUMBAI CENTRAL	23:25

Fig -2 Web Page Renderer displays various data blocks surrounded by brown color rectangle

```
<TABLE width="100%" border="0" cellpadding="0" cellspacing="1" class="table_border">
<TR class="heading_table_top">
<TH ROWSPAN = 1>Train Number</TH>
<TH ROWSPAN = 1>Train Name</TH>
<TH ROWSPAN=1>Train Source Stn</TH>
<TH ROWSPAN=1>Source Dep. Time</TH>
<TH ROWSPAN=1>Train Destination Stn</TH>
<TH ROWSPAN=1>Dest. Arr. Time</TH>
</TR>
```



```
DOM Editor
Child 2 (TABLE)
    Node Name: TABLE
    Local Name: table
    Node Value: null
    Namespace URI: http://www.w3.org/1999/xhtml
    Attribute 0 (width)
    Attribute 1 (cellspacing)
    Attribute 2 (cellpadding)
    Attribute 3 (border)
    Attribute 4 (class)
    Type: ELEMENT_NODE
    Child 0 (TBODY)
        Node Name: TBODY
        Local Name: tbody
        Node Value: null
        Namespace URI: http://www.w3.org/1999/xhtml
        Type: ELEMENT_NODE
        Child 0 (TR)
            Node Name: TR
            Local Name: tr
            Node Value: null
            Namespace URI: http://www.w3.org/1999/xh
            Attribute 0 (class)
            Type: ELEMENT_NODE
            Child 0 (TH)
                Node Name: TH
                Local Name: th
                Node Value: null
                Namespace URI: http://www.w3.org/1999
                Attribute 0 (rowspan)
                Type: ELEMENT_NODE
                Child 0 (#text)
                    Node Name: #text
                    Node Value: Train Number
                    Type: TEXT_NODE
```

Fig- 3 Sample code and corresponding DOM tree fragment

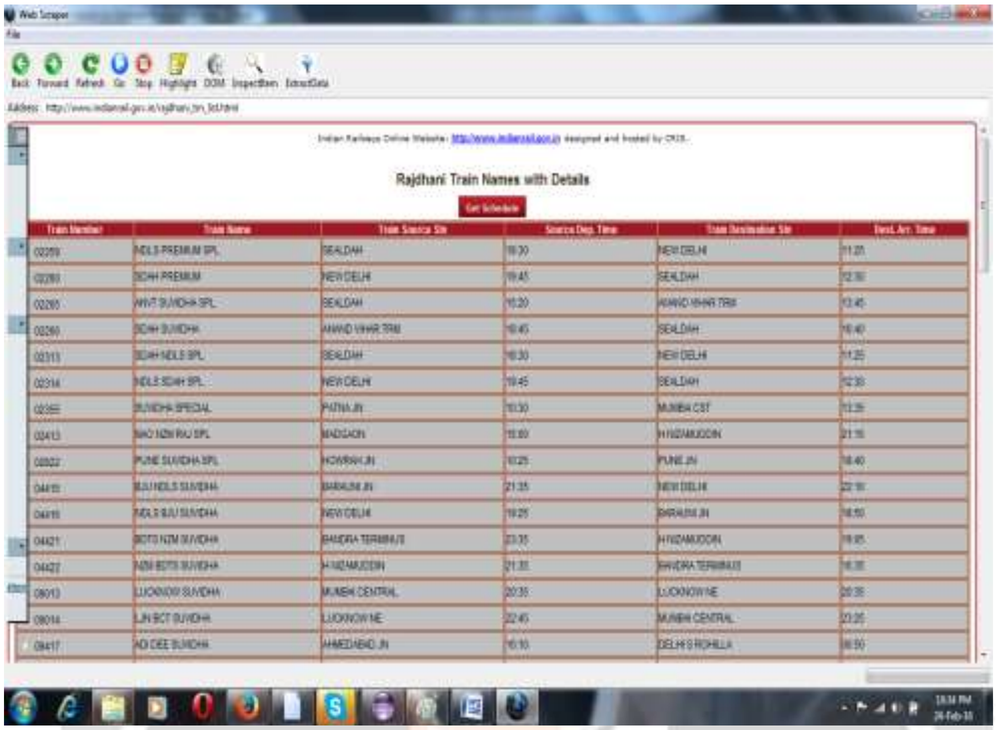
3.2 Section Selector

It will divide the input web page into various data sections. Here we will divide the web page into various data sections like list section, single valued section, and multi valued section etc. It performs following tasks.

- Identify sections
  To carry out this task an interactive user interface will be used. User will identify various data sections in the input web page. Sections includes paragraph, list and table etc. User selects the data section and it will be highlighted. Fig 4 shows highlighted user selected section.

- Identify tokens
  In this task, for selected data section it will gather semantic tokens. These tokens are used by the data extractor to traverse the DOM tree and highlight interested semantic tokens in the source page.

- Identify hierarchy
  In this task, for selected section hierarchical parent structure is retrieved. It gives a set of semantic tokens. Pattern ambiguity (if any) is then resolved using these tokens.


Web Scraper
File
Back Forward Refresh Go Stop Highlight DOM InspectItem ExtractData
Address: http://www.indianrail.gov.in/rajdhani_trn_list.html
Indian Railways Online Website: http://www.indianrail.gov.in designed and hosted by CRIS.
Get Schedule

Rajdhani Train Names with Details

Train Number	Train Name	Train Source Stn	Source Dep. Time	Train Destination Stn	Dest. Arr. Time
02259	NDLS PREMIUM SPL	SEALDAH	16:30	NEW DELHI	11:25
02260	SDAH PREMIUM	NEW DELHI	19:45	SEALDAH	12:30
02265	ANVT SUVIDHA SPL	SEALDAH	15:20	ANAND VIHAR TRM	13:45
02266	SDAH SUVIDHA	ANAND VIHAR TRM	10:45	SEALDAH	16:40
02313	SDAH NDLS SPL	SEALDAH	16:30	NEW DELHI	11:25
02314	NDLS SDAH SPL	NEW DELHI	19:45	SEALDAH	12:30
02355	SUVIDHA SPECIAL	PATNA JN	10:30	MUMBAI CST	13:35
02413	MAO NZM RAJ SPL	MADGAON	15:00	H NIZAMUDDIN	21:15
03022	PUNE SUVIDHA SPL	HOWRAH JN	10:25	PUNE JN	18:40
04415	BJU NDLS SUVIDHA	BARAUNI JN	21:35	NEW DELHI	22:10
04416	NDLS BJU SUVIDHA	NEW DELHI	19:25	BARAUNI JN	18:50
04421	BDTS NZM SUVIDHA	BANDRA TERMINUS	23:35	H NIZAMUDDIN	19:05
04422	NZM BDTS SUVIDHA	H NIZAMUDDIN	21:35	BANDRA TERMINUS	16:35
09013	LUCKNOW SUVIDHA	MUMBAI CENTRAL	20:35	LUCKNOW NE	20:35
09014	LJN BCT SUVIDHA	LUCKNOW NE	22:45	MUMBAI CENTRAL	23:25
09417	ADI DEE SUVIDHA	AHMEDABAD JN	16:10	DELHI S ROHILLA	08:50

Fig -4 Highlighted user selected section

3.3 Pattern Generator and Data Extractor

Here we will focus on pattern generation first. DOM tree nodes are categorized into 12 types as element, text, attr, entity reference, cdatasection, entity, document type, notation, comment, processing instruction, document and document fragment. We are interested in attr node only. For attr node, node name gives attribute name, node type is attr and node value gives attribute value. To construct patterns, we will use

attr node as CSS selector. There are different types of CSS selectors like attribute selectors, universal selector, descendant selectors, child selectors, type selectors, adjacent sibling selectors, class selector and id selector.

- ***Attribute Selector***
  Attribute selector specifies rules that will be used to match elements having certain attributes. It matches in following ways:

  i. [att]
  This type represents an element that has "att" attribute then whatever may be the value of the attribute it will be matched.

  ii. [att=val]
  This type matches an element which has "att" attribute and value is "val".

  iii. [att~=val]
  Here "val" is a white space separated list of words. It will matches an element with the "att" attribute and its value is one of the word represented by "val".

  iv. [att|=val]
  It matches an element having attribute "att" and value either starting with "val" and followed by "-" or being "val" itself.

- ***Class Selectors***
  Class attribute is denoted by the period (.) or (~=) notation. Thus div.value and div[class~=value] are same. Here, the attribute value must be preceded by ".". For example : ".intro": will select the elements which has class="intro".

- ***ID Selectors***
  There are attributes of type ID in the document. Attributes of type ID never have same value. Elements in document have unique ID. CSS ID selector's syntax is "# ID_value". For example : "#middlename", will retrieve the element having id="middlename". Selected section is an element of DOM tree.

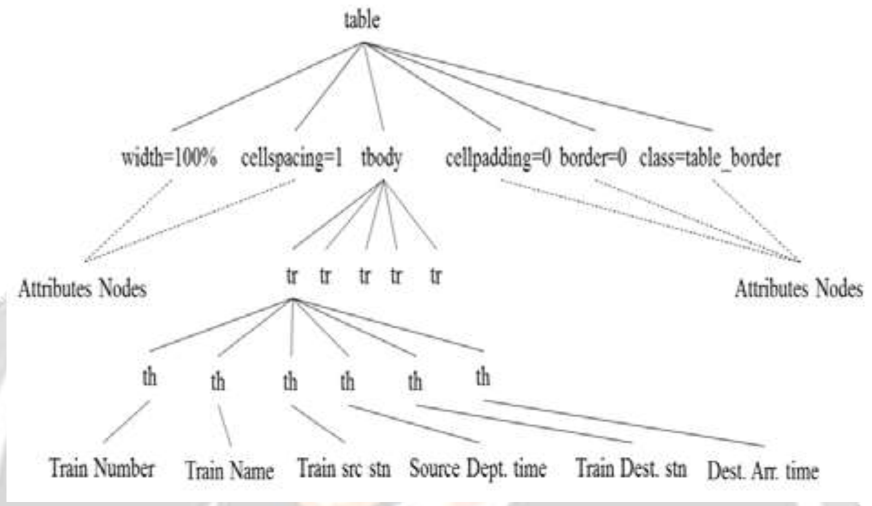
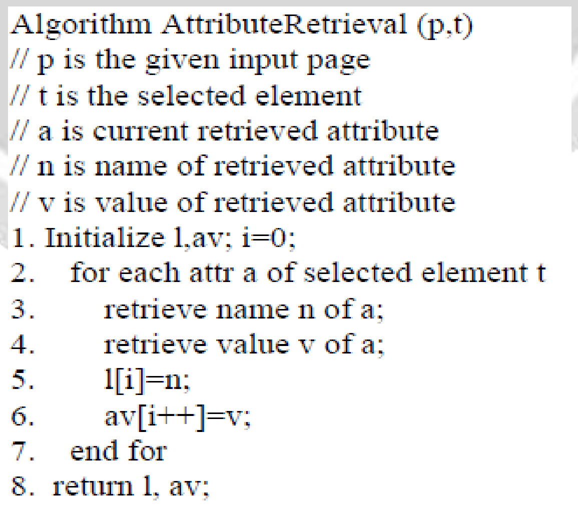


Fig -5 Part of DOM tree of rajdhani train details web page

Fig 4 shows the web page of Indian rail which displays information regarding rajdhani train names and other details. The part of DOM tree for rajdhani train web page(Fig 4) is shown in fig 5. Now we will see the algorithms to retrieve the attributes and to construct the pattern.



```
Algorithm AttributeRetrieval (p,t)
// p is the given input page
// t is the selected element
// a is current retrieved attribute
// n is name of retrieved attribute
// v is value of retrieved attribute
1. Initialize l,av; i=0;
2.    for each attr a of selected element t
3.        retrieve name n of a;
4.        retrieve value v of a;
5.        l[i]=n;
6.        av[i++]=v;
7.    end for
8.  return l, av;
```

Algo- 1 Attribute retrieval

```
Algorithm ConstructPattern(p, t, l, av)
// p is the input page
// t is the selected element
// l is the list of attribute names
//av is the list of attributes value
1.Initialize q, i=0, j=0;
2.n=sizeof(l);
3.q[j++]=insert(t.name);
4.      for each i from 0 to n
5.              q[j++]=insert([);
6.              q[j++]=insert(l[i]);
7.              q[j++]=insert(=);
8.              q[j++]=insert(av[i]);
9.              q[j++]=insert(]);
10.     end for
11.return q
```

Algo -2 ConstructPattern

From the attributes retrieved by algo 1, a pattern for selected table is generated using algo2. For example : table[width=100%][cellspacing=1][cellpadding=0][border=0][class=table_border].

3.4 Data Extractor

Using the patterns generated in previous step, data extractor will extract data. Algo 3 shows algorithm "ExtractData" to extract the data. Here "q" is a data structure which holds the patterns. "l" is the list. First this algorithm matches the selected element in the DOM tree. Once element and all of its attributes are matched, data from the element will be retrieved.

Ambiguity arises when pattern identifies two or more data sections. In this case algo 2 fails to find out correct data section. For example a web page can have two similar tables, one is under td tag and another is under div tag as shown in fig 6. Here pattern is matched twice and ambiguity arises. To resolve this ambiguity pattern validator is called.

```
Algorithm ExtractData (q , T, l)
//  l is list containing attributes
// q is the pattern derived by algo 2
// {a,v} is the attribute value pair.
// T is the DOM tree for page.
1. Flag=false;n=0;
2. retrieve element e from q
3. match e in T
4. if match found then
5.    n= sizeOf(l);
6.    For each i from 0 to n
7.        retrieve {a,v} pair
8.        match {a,v} in T in same sub tree where
          e is matched
9.      endfor
10. endif
11. retrieve text from matched sub tree
12. return text
```

Algo -3 Extract Data

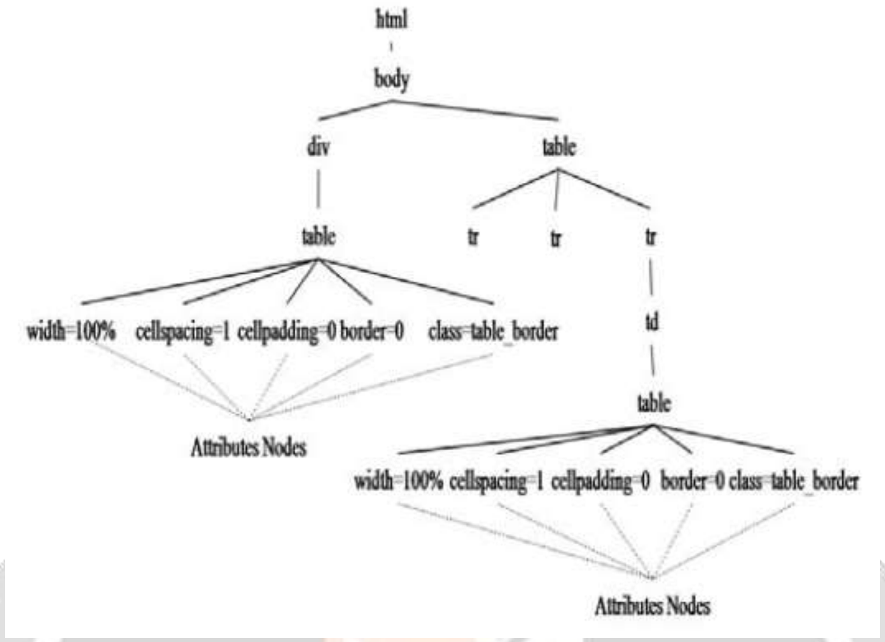

html
body
div
table
table
tr
tr
tr
td
table
width=100% cellspacing=1 cellpadding=0 border=0 class=table_border
Attributes Nodes
width=100% cellspacing=1 cellpadding=0 border=0 class=table_border
Attributes Nodes

Fig -6 Ambiguity in identifying table section

3.5 Pattern Validator

When the ambiguity arises while retrieving the data, there is a need of validating the pattern. To avoid such kind of ambiguity we will add the parent of selected section in the pattern. For example

td>table[width=100%][cellspacing=1][cellpadding=0][border=0][class=table_border].
div>table[width=100%][cellspacing=1][cellpadding=0][border=0][class=table_border].
Now, this two patterns will uniquely identify table under div tag and td tag. Thus data can be retrieved using our algorithm.

3. EXPERIMENTAL SETUP

Experiments will be performed on a m/c with an Intel Core 2 Duo processor working at 2.2 GHz clock speed and 800 MHz FSB, with 4 GB of RAM.
Java Development Kit 1.7 and Windows 7 operating system are used to perform the experiments.
It is assumed that input web page doesn't implement any anti-scraping technique.

4. EXPERIMENTAL RESULTS

Table 1 shows time based comparison between existing system and proposed system. Column 3 shows time required by existing system to extract data and column 4 shows time required by proposed system to extract data from respective webpages.

Websites	**No of Pages Tested**	**Existing System Time Requirement(in ms)**	**Proposed System Time Requirement(in ms)**
Amazon(Cars)	**10**	**13672**	**128**

Amazon(Artists)	**10**	**30681**	**178**
Buy	**2**	**7500**	**50**
Majorleaguebaseball (Players by name)	**10**	**36926**	**380**
Majorleaguebaseball (Players by statistics)	**10**	**13672**	**265**
Jewelery	**2**	**77172**	**43755**

Table -1 Time based comparison

As shown in below fig 7, the line graph represents the time analysis for the sites given in the table. Here, X-axis represents the name of website on which experiment is done and Y-axis represents time in milliseconds. Here for Y-axis value of one unit is 1000 ms. Blue line represents time required by existing system for extracting data from web pages and orange bar represents time required by proposed system for extracting data from same web pages.

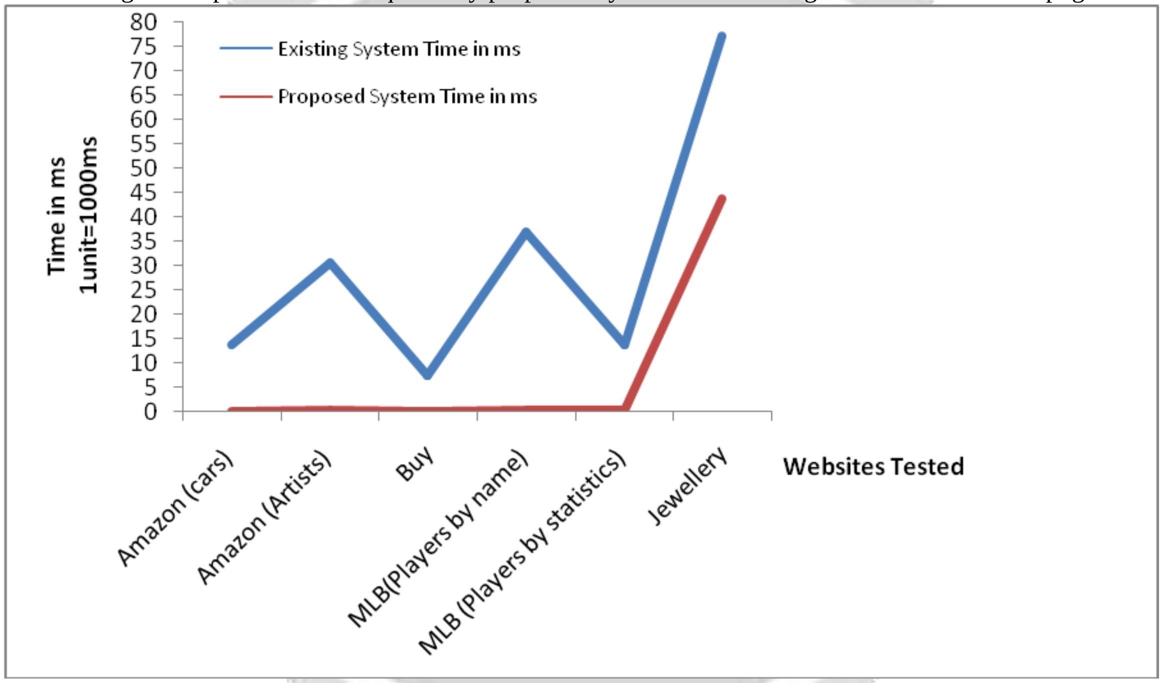


Fig - 7. Time Analysis for tested sites

Table 2 shows performance of our proposed technique. SI indicates section identification, DE indicates data extracted. An cr and pcr means correct and partially correct respectively, whereas wr means wrong. Column 2 of table contains the information about the page source i.e. URL of page. Column 2 indicates description of pages. Column 3 indicates number of pages from each source. Column 4 and 5 indicates number of pages from which data section is correctly identified and number of pages from which data section is partially correct or wrongly identified. Column 6 and 7 indicates number of pages from which data section is correctly retrieved and number of pages from which data section is partially correct or wrongly retrieved. For measuring performance of our system, we used 290 pages. Out of 290 pages, data section of 290 pages is correctly identified. For measuring performance we considered partial correct data as wrong data. After evaluation we got values of precision and recall as 100%. From experiments, we got conclusion that, our system is 100% efficient for web pages containing data in any format.

We are comparing our system with RoadRunner [15]. Experiment shows that our system is much efficient than RoadRunner in detecting sections and extracting data. We used same input pages, as those are used for evaluating RoadRunner. RoadRunner failed to extract correct data from about 21 pages. Data retrieved by

RoadRunner is partial from collection “national team info” from uefo.com. So performance of our system, DEUDS is much better than Road Runner.

Page information			SI		DE	
Page src	**Page info**	**NOP**	**cr**	**wr/pcr**	**cr**	**wr/pcr**
amazon.com	pop artist by style	19	19	0	19	0
amazon.com	cars by brand	21	21	0	21	0
buy.com	product subcategories	20	20	0	20	0
buy.com	product information	10	10	0	0	0
rpmfind.net	packages by distribution	20	20	0	20	0
rpmfind.net	packages by maintainer	20	20	0	20	0
uefa.com	players in the national team	20	20	0	20	0
uefa.com	national team info	20	20	0	20	0
wine.com	accessories	11	11	0	11	0
wine.com	wines by producers	10	10	0	10	0
majorleguebaseball.com	players by initial	10	10	0	10	0
majorleguebaseball.com	player statistics	10	10	0	10	0
nba.com	team stats	10	10	0	10	0
nba.com	team roaster	10	10	0	10	0
RISE	LA Restaurants	28	28	0	28	0
RISE	Pharma Web	10	10	0	10	0
RISE	Corel	21	21	0	21	0
Indianrail.com	Indian rail	4	4	0	4	0
Flipkart.com	Eng. books	1	1	0	1	0
Dejavutrands.com	Computers and laptops	5	5	0	5	0
shopping.yahoo.com	laptop	10	10	0	10	0
	total	290	290	00	290	0
	Precision	280/290	100%	Recall	290/290	100%

Table 2 Performance Measure of web scraper tool System

5. CONCLUSIONS

Web data extraction plays vital role in web data analysis applications. Existing systems like RoadRunner[15], DELA[10], DEPTA[7], IEPAD[14] have some limitations. Thus in this paper, we have proposed a new technique called as Web Scraper Using Selectors. WSUS automatically discovers patterns for selected data section and using those patterns it extracts data. If the structure of web site changes you don't need to change scraper, scraper is able to adapt these changes quickly. It works in three phases namely web page renderer, section selector and pattern generator and data extractor. In earlier techniques, using training examples extraction rules were identified. In this paper, we have proposed an unsupervised approach to discover the patterns and generated pattern is stable.

5. ACKNOWLEDGEMENT

I would like to thank Prof. G. M. Bhandari(HOD, Department of Computer, JSPM's BSIOTR, Pune) and Prof. Prashant Jawalkar(Assistant Professor, Department of Computer, JSPM's BSIOTR, Pune) for sharing their pearls of wisdom during development of this paper.

6. REFERENCES

[1]. S. Shinde and S. Joshi, "Schema Inference and Data Extraction from Templatized Web Pages," International Conference on Pervasive Computing (ICPC) 2015 IEEE.

[2]. Mohammed Kayed and Chia-Hui Chang, "FiVaTech: Page-Level Web Data Extraction from Template Pages," IEEE transactions on knowledge and data engineering, vol. 22, no. 2 , pp. 249-263, February 2010.

[3]. W. Liu, X. Meng, and W. Meng, "Vide: A vision-based approach for deep web data extraction", IEEE Transactionson Knowledge and Data Engineering, 22:447–460, 2010.

[4]. "Extracting data records from the web using tag path clustering," In WWW Conference, pp. 981–990, 2008.

[5]. H. Zhao, W. Meng, and C. Yu. "Mining templates from search result records of search engines", In SIGKDD Conference, New York, NY, USA, pp. 884–893, 2007.

[6]. C.-H. Chang, M. Kayed, M.R. Girgis, and K.A. Shaalan, "Survey of Web Information Extraction Systems", IEEE trans. Knowledge and Data Eng., vol. 18, no. 10, pp. 1411-1428, Oct. 2006.

[7]. Y. Zhai and B. Liu, "Web Data Extraction Based on Partial Tree Alignment", Proc. Int'l Conf. World Wide Web (WWW-14), pp. 76-85, 2005.

[8]. K. Simon and G. Lausen , "Viper: augmenting automatic information extraction with visual perceptions", In CIKM Conference, pages 381–388, New York, NY, USA, 2005.

[9]. H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. "Fully automatic wrapper generation for search engines," In WWW Conference, New York, NY, USA, pp. 66–75, 2005.

[10]. J. Wang and F.H. Lochovsky, "Data Extraction and Label Assignment for Web Databases", Proc. Int'l Conf. World Wide Web (WWW-12), pp. 187-196, 2003.

[11]. A. Arasu and H. Garcia-Molina. "Extracting structured data from web pages," In SIGMOD Conference, New York, NY, USA, pp. 337–348, 2003.

[12]. B. Liu, R. Grossman, and Y. Zhai. "Mining data records in web pages", In SIGKDD conference, New York, NY, USA, pp.601–606, 2003.

[13]. A.H.F. Laender, B.A. Ribeiro-Neto, A.S. Silva, and J.S. Teixeira, "A Brief Survey of Web Data Extraction Tools", SIGMOD Record, vol. 31, no. 2, pp. 84-93, 2002.

[14]. C.-H. Chang and S.-C. Lui, "IEPAD: Information Extraction Based on Pattern Discovery," Proc. Int'l Conf. World Wide Web (WWW- 10), pp. 223-231, 2001.

[15]. V. Crescenzi, G. Mecca, and P. Merialdo, " RoadRunner: towards automatic data extraction from large Web sites", Proceedings of the 26th International Conference on Very Large Database Systems (VLDB), Rome, Italy, 2001, pp. 109-118.

[16]. S. Chakrabarti, "Mining the Web: Discovering Knowledge from Hypertext Data" Morgan Kaufmann Publishers, 2002

[17]. R. Bhosale, P. Jawalkar, "Automatic Web Scrapping Using Visual Selecors", International Journal of Computer Technology and Applications(IJCTA), Vol 6 (6),1006-1009