

## Deep Data Mining: An Approach to Convert Semistructure Data to Structured Data Using Improved Tree Matching

K. Jayamalini and M. Ponnaivaikko

Department of Computer Science Engineering, Bharath University, Chennai, India

**Abstract:** Large amount of web data available on web pages are used by various advanced applications now a days. Web data can be obtained, when we search through query interface provided on the website. The information retrieved as result of query is called 'deep data' or 'hidden data'. These data are enwrapped in HTML pages in the form of data records. These dynamically generated web pages are special pages called Search Engine Resultant Pages (SERPs). SERPs are presented to users in the form of web documents along with other content. These web pages contain enormous amount of data and are virtual gold mine for business if they mined effectively. Web wrappers or Web Data Extraction Systems (WDES) are software applications used to interact with web sources like web pages and extract information stored in it. The data extracted from the web source are converted into structured format and stored in database for future usage. This study explains the development of a web wrapper which takes SERs as input and extracts deep data stored in it and converts them into structured format. Secondly, this study explains the steps used for wrapper generation. Thirdly, this paper explains the implementation of a tree structure algorithm called 'Improved Tree Matching (ITM)' algorithm, used for data extraction. Towards the end of this study, explained how the extracted data is stored in database. Implementation results show that compared to other existing algorithms ITM works with less complexity.

**Key words:** Web Data Extraction (WDE), wrappers, Document Object Model (DOM), Improved Tree Matching (ITM) algorithm, compared, complexity

---

### INTRODUCTION

World Wide Web (WWW) is considered as the world's largest database, holding enormous amount of various types of data that would be consumed by us for various needs. The data available on web are unstructured raw data. It is a virtual gold mine for business, if it is mined properly. Web mining (Singh and Singh, 2010) is a process that aims to discover useful information or knowledge from the web page contents, hyperlink structure and usage logs. Depending on the type of data used in the mining process, web mining (Kosala and Blockeel, 2000; Bai *et al.*, 2009) tasks are categorized into three such as Web Content Mining (WCM), Web Structure Mining (WSM) and Web Usage Mining (WUM) as shown in Fig. 1.

Web content mining (Johnson and Gupta, 2012) is used to mine information from different data like text, image, audio and video contents from the web pages. Web structure mining (Sangeetha and Joseph, 2014; Liu and Lin, 2007) is used to obtain the structure of the web site using hyperlinks, through which they provide better search results for the user. Web usage mining (Omar *et al.*, 2014; Malviya and Agrawal, 2015) is

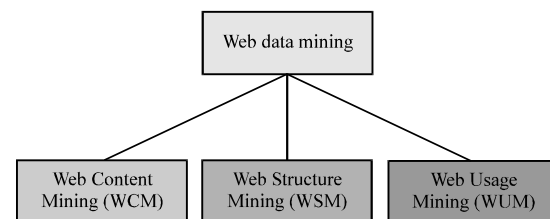


Fig. 1: Web data mining basic classifications

process of mining useful information based on the server log files, user log files and frequently accessed paths.

### MATERIALS AND METHODS

**Web data extraction systems:** Web data extraction methods are one type of Web Content mining. Web content mining is classified into Page Content Mining (PCM) and Search Result Mining (SERs) as shown in Fig. 2. Page content mining discovers useful data like text and multimedia contents from the web pages. Search result mining discovers hidden data or deep data from the Search Engine Result pages (SERs). SERs are special web pages which are generated dynamically when the

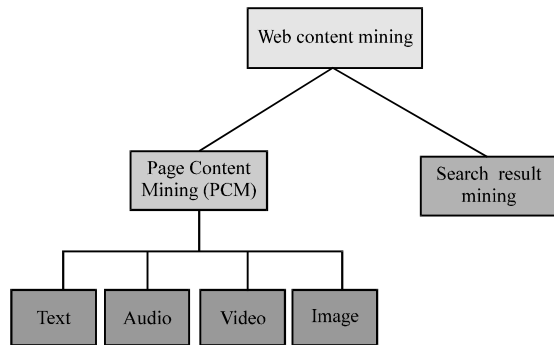


Fig. 2: Web content mining taxonomy

user executes the web query through web query interface provided in the website. The search result mining is otherwise called as web data extraction method or web wrapper generation method.

Web data extraction systems or web wrappers (Wang and Zhang, 2010; Kolkur and Jayamalini, 2013) are software systems used to extract information from web sources. Wrapper programs normally interact with a web source, extracts data stored in it and converts the extracted data in to structured format.

World wide web is considered as world's largest database, contains various types of data that we would like to consume for our needs. These data can be searched through web query interfaces provided in the website. The information retrieved as results of query is also called as deep data or hidden data. Deep data is mixed with other information like formatting tags, website titles, advertisement and navigation links, headers, footers, scripts and comments and displayed to users as web page. The other elements in the webpage make the page more human-friendly but not machine-friendly. The deep data is mixed with other elements and displayed in the form of data records. The SERPs which contains these data records are generated dynamically and represented to users in the form of HTML pages along with other contents. These webpages are virtual gold mine of information for business if mined effectively. Web wrapper (Chang *et al.*, 2006) is a program that extracts content of HTML pages and translates it into a relational form.

**Review of literature:** There are many approaches used for web data extraction in literature. Data extraction methods are generally classified (Wang and Zhang, 2010; Kolkur and Jayamalini, 2013) into following five categories.

**Methods based on ontology:** In ontology based methods, the complete knowledge base for a specific domain is

constructed and the information is extracted from the web page using the relevant rules in the knowledge base. This method mainly depends on powerful domain-specific ontology and less on structure of the webpage. This method also requires heavy workload. Shanmugasundaram studied about this method for information extraction.

**Wrappers based on visual features:** In this type of methods, visual features are used to locate and extract correct data region. These methods generate wrappers to calculate the boundary and location of a data region and take data region which is large and centrally located as the correct data region. Examples for visual based wrappers are ViDE (Liu *et al.*, 2010) VSDR and ViPER.

**Natural language processing:** In natural language processing, following steps are carried out. The structures and relationship between clauses and phrases are analyzed. Based on the syntax and semantics, the rules for extraction are generated. The source documents contain a lot of text, especially, grammatical text and then only this method is applicable. The classical systems RAPIER (Califf and Mooney, 1999) WHISK (Kushmerick *et al.*, 1997) are that are based on this principle.

**Wrapper summing up the rules:** Here, the information extraction systems makes use of machine learning techniques to learn structural features from a number of web pages and then sums up the extraction rules using the structural features. Usually, one wrapper can only handle a specific source. To extract information from different sources, a series of wrapper libraries are needed which requires a huge workload. Extraction tools like WIEN (Kushmerick *et al.*, 1997) and SoftMealy (Hsu and Dung, 1998) are based on this method.

**Wrappers based on HTML structure:** The method based on HTML structure extracts information based on structural features of HTML documents. XWRAP (Liu *et al.*, 2000) RoadRunner and W4F (Saiiuguet and Azavant, 2001) are examples for such systems.

The value-added services like competitive intelligence, comparative shopping, database integration and domain search have motivated the research on data extraction methods in recent years.

This study deals with a wrapper generation method also called as deep data mining, a method for converting semi-structured data into structured data. This system performs following steps: takes SERs as input pre-processes these SERs identifies the correct data region constructs DOM tree generates template checks the similarity between template and other webpages using

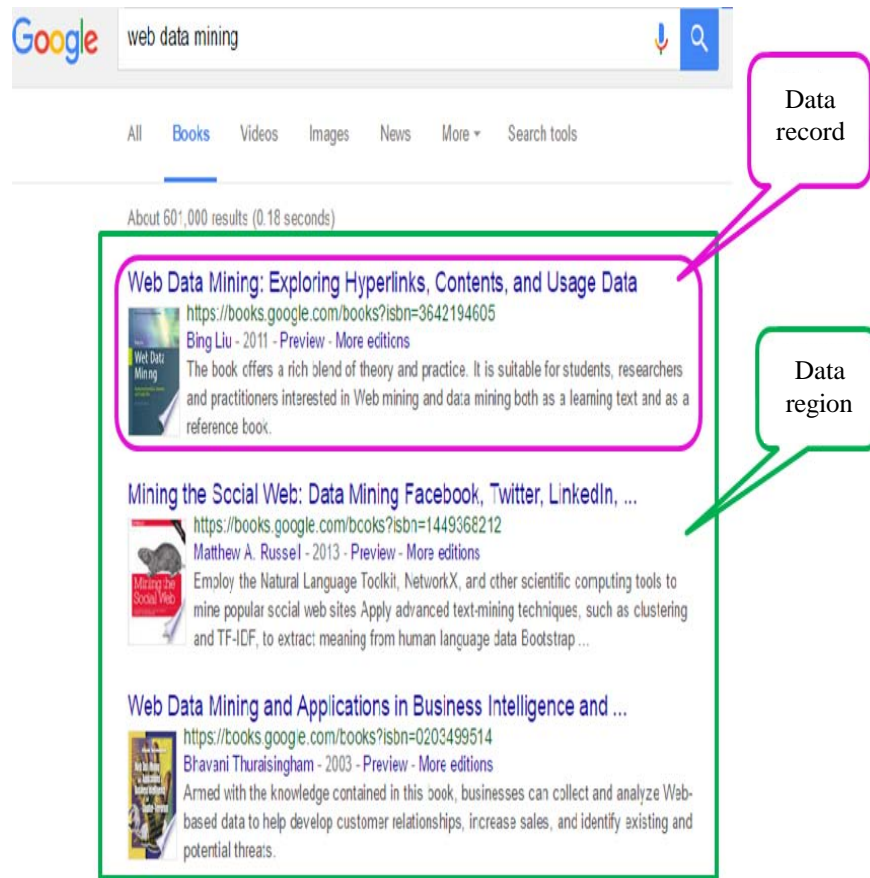


Fig. 3: Sample SERP from www.books.google.com

Improved Tree Matching (ITM) algorithm (Kolkur and Jayamalini, 2013) converts the data into structured format store extracted data in database.

XPath is a language that is used to address parts of an XML documents (i.e.,) to find and locate the information that the users are interested in structured documents. It is designed to use by both XSLT and Xpointer.

For this resarcher, SERs (Search Engine Result pages) are collected from the website 'www.books.google.com'. A sample SERP with data records (purple rounded rectangle box) and data region (a collection of data records) is shown in (green rectangular box) (Fig. 3).

In this system, SERPs and template pages are represented in the form of DOM tree. This system would be tested using a set of web pages from www.books.google.com. Improved Tree Matching (ITM) algorithm (Kolkur and Jayamalini, 2013) is used in this work for checking similarity. Web data is extracted from the pages which are giving similarity value greater than or equal to the threshold value are stored in database for future use.

**Overview of proposed system:** The block diagram of the proposed web data extraction system is given in Fig. 4. For this system, the SERPs are given as input. The data extraction procedure is divided into SERPs pre-processing, identifying correct data region, construction of DOM tree, compression of dOM tree, template generation, similarity check using improved tree matching algorithm, data extraction and storing extracted data.

The value of similarity between template and the web page is greater than or equal to threshold value is considered as page for data extraction and data canbe extracted from that page. The extracted data is stored in the database in a structured format for future use.

**Improved tree matching algorithm:** By Wang and Zhang, (2010) simple tree matching algorithm and extended tree matching algorithm were proposed by resarchers. Both the algorithms calculate the maximum number of node-pair between two trees using dynamic programming. These algorithms find the maximum matching pair between the

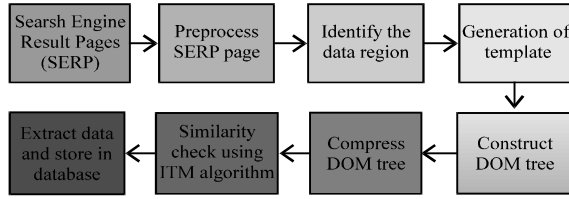


Fig. 4: Block diagram of the proposed system

first-level sub-trees of the two trees (T1, T2) by recursively comparing the sub-trees of T1 with all the sub-trees of T2.

Kolkur and Jayamalini (2013) researchers proposed Improved Tree Matching (ITM) algorithm to calculate the similarity between two SERPs. The SERPs follow similar structure because a pre-defined template generates them automatically and dynamically. This algorithm checks the similarity of two documents with reduced complexity when compared with STM and ESTM proposed by Wang and Zhang (2010), Kushmerick *et al.* (1997).

Let A and B are two trees and  $R_A$  and  $R_B$  are root nodes of A and B. It is defined as  $A_- = R_A: \langle A_1, A_m \rangle$  and  $B_- = R_B: \langle B_1, B_n \rangle$ . Each tree has many sublevels and each sublevels have sub sublevels and so on.  $A_i$  and  $B_j$  be the  $i$ th and  $j$ th node of the first-level sub trees of tree A and tree B, respectively. If the root nodes  $R_A$  and  $R_B$  are the same, then ITM calculates the maximum matching node pairs between Tree A and B is calculated by comparing the nodes in sub-tree 1 of tree A with nodes in sub-tree 1 of tree B. Each time, if the match is found, the value of matrix  $Max. [i, j]$  will be incremented by 1. This process will continue for all the sub trees of tree A and tree B at all the levels. If  $R_A$  and  $R_B$  contain distinct symbols, then  $Max. [i, j] = 0$ . The improved tree matching algorithm is given below.

#### Algorithm 1; ITM [A, B]:

```

If  $R_A = R_B$  then
    return 0
Else
    m = Size of Tree A
    n = Size of Tree B
    M [i, 0] = 0
    for i = 0, ..., m
        M [0, j] = 0 for j = 0, ..., n
    for i = 1 to m do
        for j = 1 to n do
            If  $(A[i][j] == B[j][i])$ 
                matching_node_pair++
            M [i, j] = matching_node_pair++
        end for
    end for
    return Maximum (M)
end if
    
```

To understand the working of ITM algorithm, take two trees shown in Fig. 5a, b. First, the root nodes

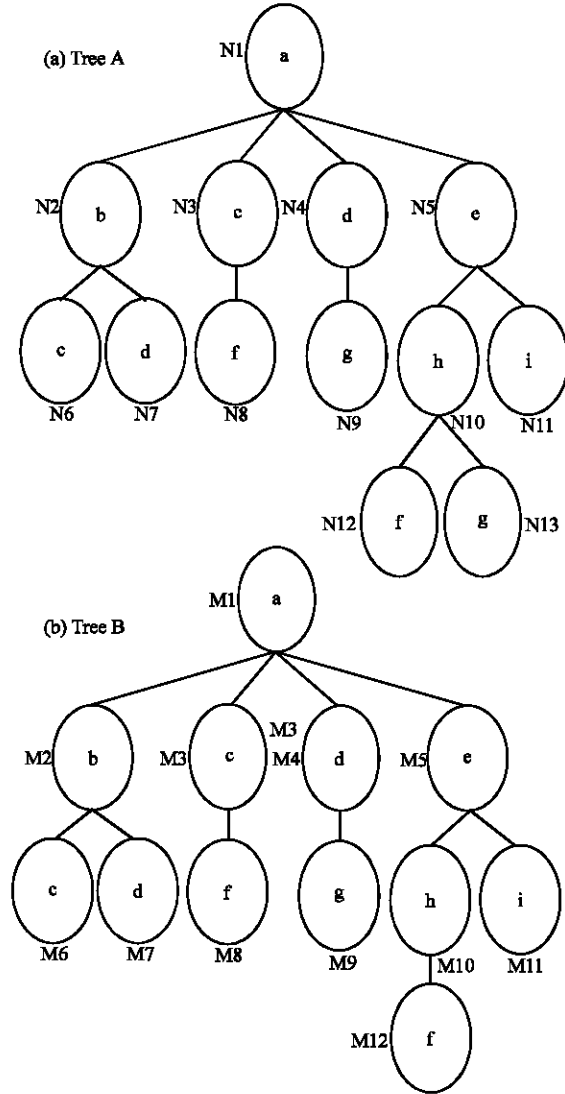


Fig. 5: Sample DOM trees

N1 and M1 of trees A and B are compared. If both the nodes have the same symbols, the ITM algorithm finds the maximum matching node pair between the first-level sub trees A and B and stores the matching score in the M matrix. If they contain distinct symbols, the two trees totally do not match. First, the roots, nodes N1 and N14 are compared. Since, the roots are same, the maximum matching at first level subtrees of A and B is calculated and saved in M matrix. In the above example, N2 of tree A is compared with M2 of tree B. The matching node-pairs between first sub-tree of both the trees is equal to 3 which is stored in M matrix. The construction of M matrix for the above example is shown in Table 1. The ITM algorithm returns 11 matching nodes pairs between Tree A and Tree B for the above trees given as examples.

## RESULTS AND DISCUSSION

### Deep data extraction-implementation steps and results:

The steps of wrapper generation can be divided into SERPs pre-processing, identification of correct data region, construct and compress DOM tree, selection of template page, similarity check using ITM algorithm, extracting data and store it in database.

**SERPs pre-processing:** Several mature HTML parsers like HTML-Cleaner, Neko HTML, HTML Parser, JSoup and J Tidy are used to clean the HTML pages which is normally found dirty, ill-formed and unsuitable for further processing on the web. The semi-structured nature of HTML language make the html source files non-standardized. For any serious consumption of these html documents, it is necessary to clean them up. These documents contain navigational bars, advertisements, other multimedia contents and formatting elements like labels, script, font and comments. They are considered

insecure code and these elements need to be filtered out in pre-processing phase. The original SERP from www.books.google.com is given in Fig. 6.

**Identification of correct data region:** All the web pages contain many data regions. Identification of data region which contains data of the user's interest is carried out at this phase. A HTML Parser called HTML-Cleaner was used for this purpose. HTML-Cleaner is an open-source HTML parser written in Java which can reorder individual elements of an HTML document and produce well-formed XML documents. The pre-processed web pages are converted into structured hierarchical tree (XML) using HTML-Cleaner. It also provides methods for tag filtering which are used to identify and extract the correct data region. The identified data region from a preprocessed page is shown in Fig. 7.

**Construct and compress DOM tree:** The DOM views HTML and XML documents as a tree-structure. All elements of these documents can be accessed through the DOM tree. The elements, their text and their attributes are all known as nodes. DOM tree is constructed for that specific data region using Apache's xercesImpl.jar file. The DOM tree for sample page 1 is shown in Fig. 8.

Table 1: M matrix construction for matching node pairs

Values	M2	M3	M4	M5
N2	3	-	-	-
N3	-	5	-	-
N4	-	-	7	-
N5	-	-	-	11

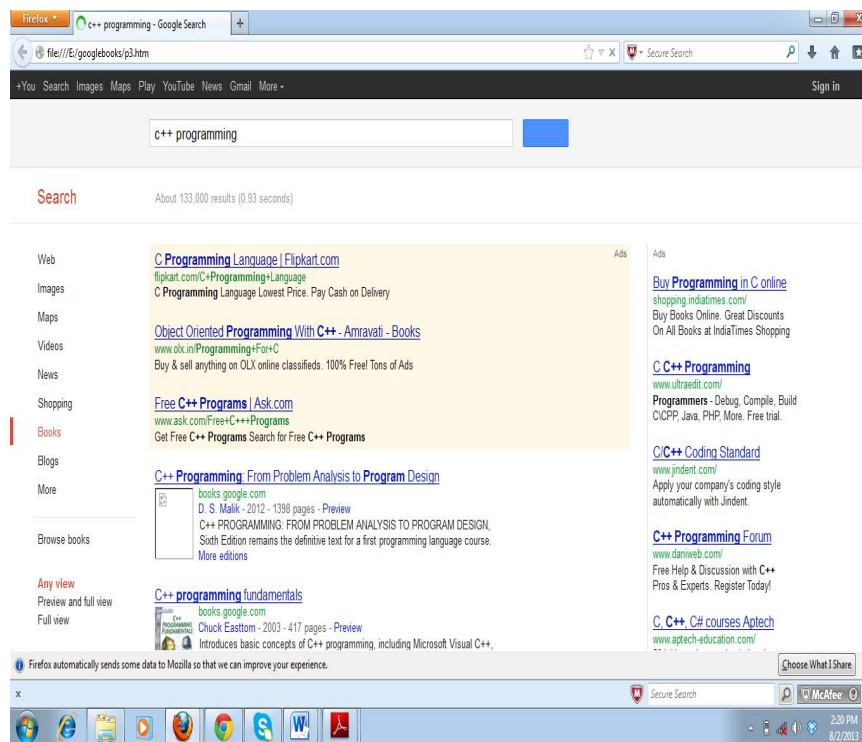


Fig. 6: Sample SERP form Google books before preprocessing

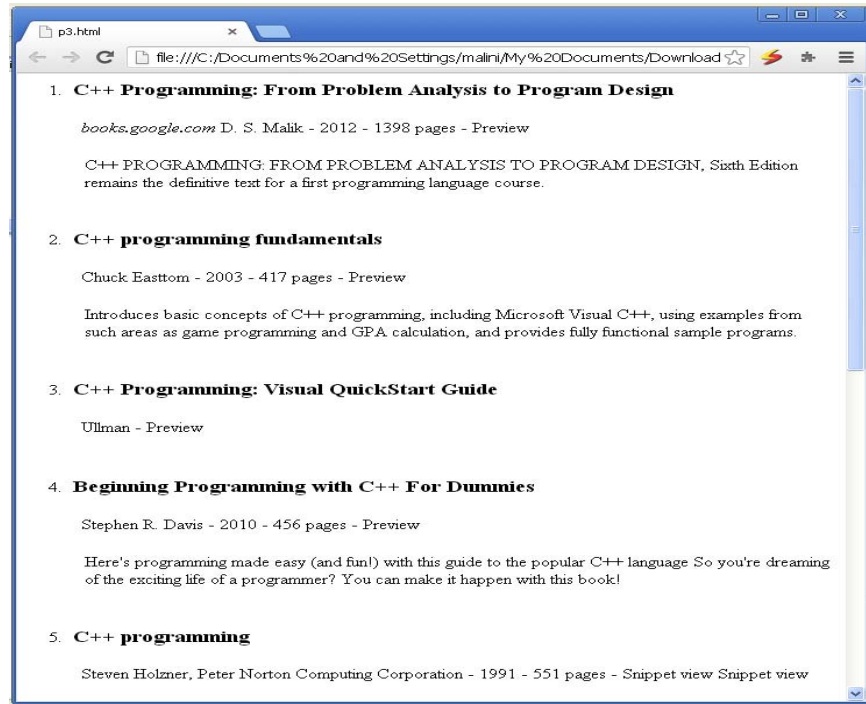


Fig. 7: Preprocessed and identified data region of sample SERP

Many formatting tags like <color>, <font>, <b>, <i>, <em>, <heading>, <table> etc are used by html designers while designing webpages to make them more user friendly. These tags massively increase the size of the DOM tree structure. Hence, to ensure optimal the size for the DOM tree, the unwanted tags to be removed.

**Template generation:** SERPs have very similar page structure because some pre-defined templates usually generate those pages automatically and dynamically and those templates handle data of the same data source with a fixed display format. In this approach, user would enter a query through a Web interface, the SERPs are obtained. For the required data region of the SERP, the DOM tree is constructed and randomly selected from the dataset. The selected DOM tree is the template tree, T, for the ITM extraction.

**Computing similarity check:** Improved Tree Matching Algorithm (ITM) (Kolkur and Jayamalini, 2013) was used for similarity check in this system. The formula for checking similarity between two trees A and B can be defined as follows Eq. 1:

$$\text{Similarity}(A, B) = \frac{\text{Improved tree matching}(A, B)}{(\text{Size}(A) + \text{Size}(B)) / 2} \quad (1)$$

where, improved tree matching (A, B) returns the number of maximum matching node pairs between tree A and tree B; sizes (A) and (B) returns the number of nodes in tree A and B, respectively. If the similarity value between two trees is, closer to 1, then tree A and B are very similar to each other and the HTML documents they represent are also very similar and the web data will be extracted from that page otherwise, the two trees are not similar. The threshold value is set in this approach was 0.6.

Table 2 shows the similarity value between thirty web pages and ten randomly selected template pages using Improved Tree Matching algorithm (ITM). The implementation results show that all the pages look like similar to each other and considered for data extraction.

**Data extraction and store in database:** Pages whose similarity value greater than or equal to the threshold value are considered for data extraction. The pages from which the data was extracted are stored in oracle 10 g database in XML format for future use. A pagedetail table and a DOMtree table are used to store the details of pages and their corresponding DOM tree (Fig. 9 and 10).

The DOM tree structure of each page is stored in a table DOMtree. The schema for table DOMtree is shown in Fig. 11 and 12 the snapshot of table DOMtree.

Table 2: Similarity values of ITM

Page template	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10
P1	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P2	0.92	1.00	0.96	0.92	0.83	0.79	0.83	0.92	0.88	0.92
P3	0.96	0.96	1.00	0.96	0.79	0.79	0.79	0.96	0.92	0.96
P4	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P5	0.75	0.83	0.79	0.75	1.00	0.93	1.00	0.75	0.79	0.83
P6	0.75	0.79	0.79	0.75	0.93	1.00	0.93	0.75	0.76	0.83
P7	0.75	0.83	0.79	0.75	1.00	0.93	1.00	0.75	0.79	0.83
P8	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P9	0.96	0.88	0.92	0.96	0.79	0.76	0.79	0.96	1.00	0.88
P10	0.92	0.92	0.96	0.92	0.83	0.83	0.83	0.92	0.88	1.00
P11	0.96	0.88	0.92	0.96	0.79	0.79	0.79	0.96	0.92	0.88
P12	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P13	1.00	0.92	0.96	1.00	0.75	0.72	0.75	1.00	0.96	0.92
P14	0.16	0.16	0.16	0.16	0.15	0.15	0.15	0.16	0.16	0.16
P15	0.14	0.14	0.14	0.14	0.14	0.13	0.14	0.14	0.14	0.14
P16	0.15	0.15	0.15	0.15	0.14	0.14	0.14	0.15	0.15	0.15
P17	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P18	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P19	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P20	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P21	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P22	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P23	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P24	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P25	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P26	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P27	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P28	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P29	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92
P30	1.00	0.92	0.96	1.00	0.75	0.75	0.75	1.00	0.96	0.92

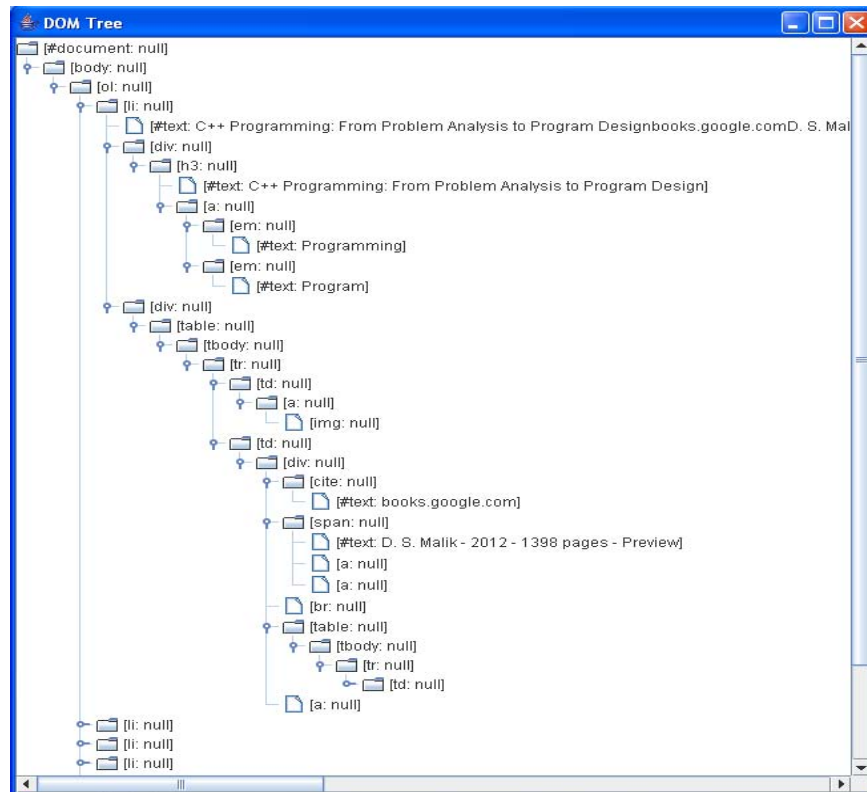
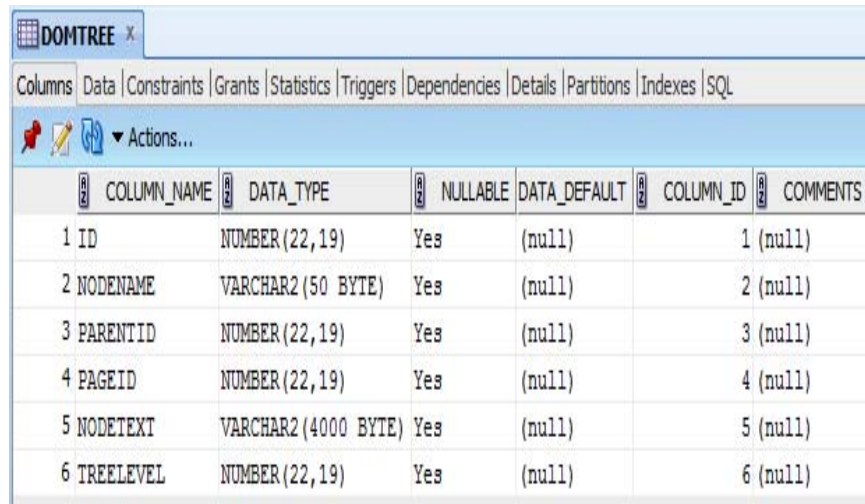


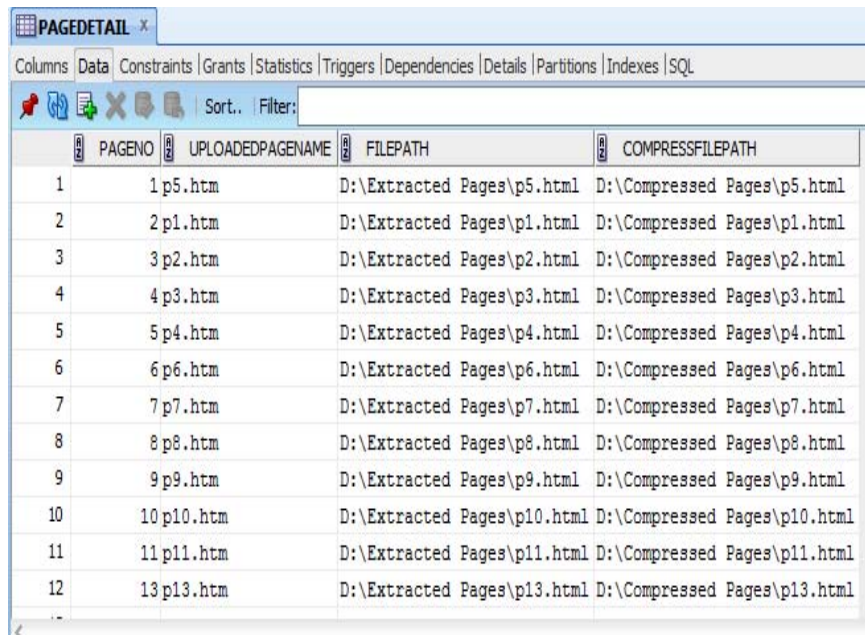
Fig. 8: DOM tree representation of sample SERP





	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	ID	NUMBER(22,19)	Yes	(null)	1	(null)
2	NODENAME	VARCHAR2(50 BYTE)	Yes	(null)	2	(null)
3	PARENTID	NUMBER(22,19)	Yes	(null)	3	(null)
4	PAGEID	NUMBER(22,19)	Yes	(null)	4	(null)
5	NODETEXT	VARCHAR2(4000 BYTE)	Yes	(null)	5	(null)
6	TREELEVEL	NUMBER(22,19)	Yes	(null)	6	(null)

Fig. 9: Schema for table PAGEDETAIL



	PAGENO	UPLOADEDPAGEName	FILEPATH	COMPRESSFILEPATH
1	1	p5.htm	D:\Extracted Pages\p5.html	D:\Compressed Pages\p5.html
2	2	p1.htm	D:\Extracted Pages\p1.html	D:\Compressed Pages\p1.html
3	3	p2.htm	D:\Extracted Pages\p2.html	D:\Compressed Pages\p2.html
4	4	p3.htm	D:\Extracted Pages\p3.html	D:\Compressed Pages\p3.html
5	5	p4.htm	D:\Extracted Pages\p4.html	D:\Compressed Pages\p4.html
6	6	p6.htm	D:\Extracted Pages\p6.html	D:\Compressed Pages\p6.html
7	7	p7.htm	D:\Extracted Pages\p7.html	D:\Compressed Pages\p7.html
8	8	p8.htm	D:\Extracted Pages\p8.html	D:\Compressed Pages\p8.html
9	9	p9.htm	D:\Extracted Pages\p9.html	D:\Compressed Pages\p9.html
10	10	p10.htm	D:\Extracted Pages\p10.html	D:\Compressed Pages\p10.html
11	11	p11.htm	D:\Extracted Pages\p11.html	D:\Compressed Pages\p11.html
12	12	p13.htm	D:\Extracted Pages\p13.html	D:\Compressed Pages\p13.html

Fig. 10: Snapshot of table PAGEDETAIL

**Evaluating deep data extraction method:** Recall and precision are two parameters used to evaluate information extraction method. Recall R is equal to the ratio of the properly extracted information taken in the right information while the precision P is equal to the ratio of the properly extracted information taken in the information extracted. The formulae for calculating precision and recall are given:

$$\text{Recall (R)} = \frac{\text{No. of page extracted}}{\text{Total No. of page}} \quad (2)$$

$$\text{Precision (P)} = \frac{\text{No. of pages correctly extracted}}{\text{No. of pages extracted}} \quad (3)$$

In this research:

- Number of pages used for testing = 30
- Number of pages extracted, i.e., similarity  $\geq 0.6 = 27$
- Number of pages correctly extracted
- (Manually Checked) = 27



	COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1	PAGENO	NUMBER(19,0)	Yes	(null)	1 (null)	
2	UPLOADEDPAGENAME	VARCHAR2(400 BYTE)	Yes	(null)	2 (null)	
3	FILEPATH	VARCHAR2(1000 BYTE)	Yes	(null)	3 (null)	
4	COMPRESSFILEPATH	VARCHAR2(1000 BYTE)	Yes	(null)	4 (null)	

Fig. 11: Schema for table DOMtree

ID	NODENAME	PARENTID	PAG...	NODETEXT	TREELEVEL
1	11 b	10	1 (null)		9
2	12 span	9	1 Gonzalez - 2009 - 954 pages - Preview		8
3	13 table	9	1 (null)		8
4	14 tbody	13	1 (null)		9
5	15 tr	14	1 (null)		10
6	16 td	15	1 (null)		11
7	17 span	16	1 Completely self-contained-and heavily illustrated...		12
8	18 li	2	1 (null)		3
9	19 h3	18	1		4
10	20 table	18	1 (null)		4
11	21 tbody	20	1 (null)		5
12	22 tr	21	1 (null)		6

Fig. 12: Snapshot of table DOMtree

$$\text{Recall (R)} = \frac{\text{No. of page extracted}}{\text{Total No. of page}} = 27/30 = 90\%$$

$$\text{Precision (P)} = \frac{\text{No. of pages correctly extracted}}{\text{No. of pages extracted}} = 27/27 = 100\%$$

## CONCLUSION

This study explains the various steps involved to extract data records from similar web pages effectively using improved tree matching algorithm. This research,

i.e., deep data extraction method based on improved tree matching algorithm integrates the structure similarity of web pages and its correlation with XPath. This algorithm takes full advantage of the structural similarity of the web pages generated by the predefined templates and effectively extract data records from similar web pages. Finally, the extracted data was stored in XML format for future use.

## RECOMMENDATIONS

To meet people's demand of intelligent information, we will plan to study how to reduce the manual involvement and establish full automatic intelligent

information retrieval systems. In future, we will plan to study extracting data from webpages which have different in structures.

### ACKNOWLEDGEMENTS

We wish to express our sincere thanks and deep sense of gratitude to all the staff members of computer department of Bharath University, Chennai for their support and co-ordination. We also thank to our family members for their support to complete this resarcher.

### REFERENCES

- Bai, S., Q. Han, Q. Liu and X. Gao, 2009. Research of an algorithm based on web usage mining. Proceedings of the International Workshop on Intelligent Systems and Applications (ISA'09), May 23-24, 2009, IEEE, Wuhan, China, ISBN:978-1-4244-3893-8, pp: 1-4.
- Califf, M.E. and R.J. Mooney, 1999. Relational learning of pattern-match rules for information extraction. Proceedings of the 16th National Conference on Artificial Intelligence and the 11th Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence, July 18-22, 1999, American Association for Artificial Intelligence, Menlo Park, CA., pp: 328-334.
- Chang, C.H., M. Kayed, R. Girgis and K.F. Shaalan, 2006. A survey of web information extraction system. Inst. Electr. Electron. Eng. Trans. Knowledge Data Eng., 18: 1411-1428
- Hsu, C.N. and M.T. Dung, 1998. Generating finite-state transducers for semi-structured data extraction from the web. Inf. Syst., 23: 521-538.
- Johnson, F. and S.K. Gupta, 2012. Web content mining techniques: A survey. Intl. J. Comput. Appl., 47: 44-50.
- Kolkur, S. and K. Jayamalini, 2013. Web data extraction using tree structure algorithms-a comparison. Intl. J. Recent Technol. Eng., 2: 35-39.
- Kosala, R. and H. Blockeel, 2000. Web mining research: A survey. ACM SIGKDD Explorat. Newsl., 2: 1-15.
- Kushmerick, N., D. Weld and R. Doorenbos, 1997. Wrapper induction for information extraction. Proceedings of the International Joint Conference on Artificial Intelligence, August 23-24, 1997, Nagoya, Japan, pp: 729-737.
- Liu, L., C. Pu and W. Han, 2000. XWRAP: An XML-enabled wrapper construction system for web information sources. Proceedings of the 16th International Conference on Data Engineering, March 3, 2000, IEEE, San Diego, California, pp: 611-621.
- Liu, W., X. Meng and W. Meng, 2010. Vide: A vision-based approach for deep web data extraction. IEEE. Trans. Knowl. Data Eng., 22: 447-460.
- Liu, Y. and Y. Lin, 2007. Supervised HITS algorithm for MEDLINE citation ranking. Proceedings of the 7th IEEE International Conference on Bioinformatics and Bioengineering (BIBE'07), October 14-17, 2007, IEEE, Boston, Massachusetts, ISBN:1-4244-1509-8, pp: 1323-1327.
- Malviya, B.K. and J. Agrawal, 2015. A study on web usage mining theory and applications. Proceedings of the 5th International Conference on Communication Systems and Network Technologies (CSNT'15), April 4-6, 2015, IEEE, Gwalior, India, ISBN:978-1-4799-1797-6, pp: 935-939.
- Omar, R., A.O.M. Tap and Z.S. Abdullah, 2014. Web usage mining: A review of recent works. Proceedings of the 5th IEEE International Conference on Information and Communication Technology for The Muslim World (ICT4M), November 17-18, 2014, IEEE, Kuching, Malaysia, ISBN:978-1-4799-6243-3, pp: 1-5.
- Saiiuguet, A. and F. Azavant, 2001. Building intelligent web applications using lightweight wrappers. Data Knowledge Eng., 36: 283-316.
- Sangeetha, M. and K.S. Joseph, 2014. Page ranking algorithms used in Web Mining. Proceedings of the IEEE International Conference on Information Communication and Embedded Systems (ICICES2014), February 27-28, 2014, IEEE, Chennai, India, ISBN:978-1-4799-3698-4, pp: 1-7.
- Singh, B. and H.K. Singh, 2010. Web data mining research: A survey. Proceedings of the IEEE International Conference on Computational Intelligence and Computing Research, December 28-29, 2010, IEEE, Coimbatore, India, ISBN: 978-1-4244-5965-0, pp: 1-10.
- Wang, H. and Y. Zhang, 2010. Web data extraction based on simple tree matching. Proceedings of the WASE International Conference on Information Engineering (ICIE'10) Vol. 2, August 14-15, 2010, IEEE, Beidaihe, Hebei, ISBN:978-1-4244-7506-3, pp: 15-18.