International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:3, 2009

Extraction of Data from Web Pages : A Vision Based Approach

P. S. Hiremath, Siddu P. Algur

***Abstract*—**With the explosive growth of information sources available on the World Wide Web, it has become increasingly difficult to identify the relevant pieces of information, since web pages are often cluttered with irrelevant content like advertisements, navigation-panels, copyright notices etc., surrounding the main content of the web page. Hence, tools for the mining of data regions, data records and data items need to be developed in order to provide value-added services. Currently available automatic techniques to mine data regions from web pages are still unsatisfactory because of their poor performance and tag-dependence. In this paper a novel method to extract data items from the web pages automatically is proposed. It comprises of two steps: (1) Identification and Extraction of the data regions based on visual clues information. (2) Identification of data records and extraction of data items from a data region. For step1, a novel and more effective method is proposed based on visual clues, which finds the data regions formed by all types of tags using visual clues. For step2 a more effective method namely, Extraction of Data Items from web Pages (EDIP), is adopted to mine data items. The EDIP technique is a list-based approach in which the list is a linear data structure. The proposed technique is able to mine the non-contiguous data records and can correctly identify data regions, irrespective of the type of tag in which it is bound. Our experimental results show that the proposed technique performs better than the existing techniques.

***Keywords*—**Web data records, Web data regions, Web mining,.

I. Introduction

WEB information extraction is an important task for information integration, because multiple web pages may present the same or similar information using completely different formats or syntaxes, which makes integration of information a challenging task. Due to the heterogeneity and lack of structure of web data, automated discovery of targeted information becomes a complex task. A typical web page consists of many blocks or areas, e.g., main content areas, navigation areas, advertisements, etc. For a particular application, only part of the information is useful, and the rest are noises. Hence, it is useful to separate these areas automatically. Pages in data-intensive web sites are usually automatically generated from the back-end DBMS using scripts [6]. Hence, the structured data on the web are often very important since they represent their host page's essential information, e.g., details about the list of products and services.

In order to provide value added services, by extracting and making use of information from multiple sites, one needs to semantically integrate information from multiple sources. This problem has been studied by researchers in AI, database and data mining, and web communities [9]. There are several approaches [2], [4], [5], [6], [14] for structured data extraction, which is also called wrapper generation.

The first approach [2] is to manually write an extraction program for each web site based on observed format patterns of the site. This manual approach is very labor intensive and time consuming. Hence, it does not scale to a large number of sites.

The second approach [4] is wrapper induction or wrapper learning, which is currently the main technique. Wrapper learning works as follows: The user first manually labels a set of trained pages. A learning system then generates rules from the training pages. The resulting rules are then applied to extract target items from web pages. These methods either require prior syntactic knowledge or substantial manual efforts. AN example of wrapper induction systems is WEIN [1].

The third approach [5] is the automatic approach. The structured data objects on a web are normally database records retrieved from underlying web databases and displayed in web pages with some fixed templates. Automatic methods aim to find patterns/grammars from the web pages and then use them to extract data. Examples of automatic systems are IEPAD [5], ROADRUNNER [6], MDR [10], DEPTA [15] and VIPS [11]. These systems make use of the Patricia (PAT) tree for discovering the record boundaries automatically and a pattern-based extraction rule to extract the web data. This method has a poor performance due to the various limitations of the PAT tree. ROADRUNNER [6] extracts a template by analyzing a pair of web pages of the same class at a time. It uses one page to derive an initial template and then tries to match the second page with the template. Deriving of the initial template has to be again done manually, which is a major limitation of this approach.

Another problem with the existing automatic approaches is their assumption that the relevant information of a data record is contained in a contiguous segment of HTML code, which is not always true. MDR [10] basically exploits the regularities

P S Hiremath is with the Dept. of Computer Science, Gulbarga University, Gulbarga, Karnataka, India (e-mail: hirematheps@yhotmail.com).

Siddu P Algur, is with Dept. of Information Science & Engineering., BVB College of Engineering & Technology., Hubli, Karnataka, India (e-mail: algursp@bvb.edu, siddu_p_algur@hotmail.com).

in the HTML tag structure directly. It is often very difficult to derive accurate wrappers entirely based on HTML tags. MDR algorithm makes use of the HTML tag tree of the web page to extract data records from the page. However, an incorrect tag tree may be constructed due to the misuse of HTML tags, which in turn makes it impossible to extract data records correctly. DEPTA [15] uses visual information (locations on the screen at which the tags are rendered) to find data records. Rather than analyzing the HTML code, the visual information is utilized to infer the structural relationship among tags and to construct a tag tree. But this method of constructing a tag tree has the limitation that, the tag tree can be built correctly only as long as the browser is able to render the page correctly. The computation time for constructing the tag tree is also an overhead. However, this method also fails to identify some of the data records. VIPS [11] is based on certain visual cues within the content of a node of a parse tree. Yu et al. [11] suggest that tags such as <HR>, which is used to create a horizontal line, as well as attributes that indicate a change in background color can indicate the beginning or end of a segment. These segmentation cues are seen as "visual separators" and are classified as horizontal or vertical lines. After the VIPS algorithm has parsed the HTML code, visual separators are detected in the parse tree. The separators receive weights which are adjusted depending on constraints based on separator dimension, adjacent tags such as <HR> (a line), and surrounding font sizes. The separators divide sub-trees further, depending on if they cross a "visual" block of information or not. Finally, the content structure of the page is created, by merging "visual" blocks that are not divided by separators. Knowledge on document structure is used to enhance the quality of query expansion terms in information retrieval. VIPS also has certain limitations. It is dependent on number of heuristic rules which cannot be applied for most of the web pages. Further, it does not correctly identify the data regions.

The objective of this paper is to extract the data items from a given webpage using visual clues and list based approach. The proposed technique is implemented in two steps to solve the problem:

i) Given a web page, the first method (or step) identifies and extracts the data region based on the visual clue (location of data region / data records / data items / on the screen at which the tags are rendered) information of web pages. The algorithm is called **VSAP** (**V**isual **S**tructure based **A**nalysis of web **P**ages [16]). It finds the data regions formed by all types of tags using visual clues. The VSAP technique is based on the observations which are found to be true in the experiment.

ii) Given a relevant data region (i.e. the output of VSAP algorithm), the second method (or step) identifies the data records and extract the data items from it. The algorithm is here after called **EDIP** (**E**xtraction of **D**ata **I**tems from Web **P**ages) which is a list based approach [17]. It finds the data items formed by all types of tags.

The paper is organized into six sections. The section 2 presents the related work reported in the literature. The section 3 deals with the proposed technique. The section 4 contains empirical evaluation of the proposed technique. The section 5 gives the experimental results. Finally the conclusions are given in section 6.

II. Related Work

Extracting the regularly structured data records from web pages is an important problem. So far, several attempts have been made to deal with the problem.

Related works, mainly in the area of mining data records in a web page automatically, are MDR [10], DEPTA [15].

The first approach, MDR [10] automatically mines all data records formed by table and form related tags i.e, <TABLE>, <FORM>, <TR>, <TD>, etc. assuming that a large majority of web data records are formed by them. The algorithm is based on two observations.

a) A group of data records are always presented in a contiguous region of the web page and are formatted using similar HTML tags. Such region is called a *Data Region*.
b) The nested structure of the HTML tags in a web page usually forms a tag tree and a set of similar data records are formed by some child sub-trees of the same parent node.

The algorithm works in three steps:

Step 1. Building the HTML tag tree by following the nested blocks of the HTML tags in the web page.
Step 2. Identifying the data regions by finding the existence of multiple similar generalized nodes of a tag node. A *generalized node* (or a *node combination*) is a collection of child nodes of a tag node, with the following two properties.

i) All the nodes have the same parent.
ii) The nodes are adjacent.

Then each generalized node is checked to decide if it contains multiple records or only one record. This is done by string comparison of all possible combinations of component nodes using Normalized Edit Distance method [1].

A *data region* is a collection of two or more generalized nodes with the following properties:

i) The generalized nodes all have the same parent.
ii) The generalized nodes all have the same length.
iii) The generalized nodes are all adjacent.
iv) The normalized edit distance (string comparison) between adjacent generalized nodes is less than a fixed threshold.

Step 3. Identifying the data records involves finding the data records from each generalized node in a data region.

All the three steps of MDR have certain serious limitations which will be discussed in the latter half of the paper.

The second approach, DEPTA (Data Extraction based on Partial Tree Alignment), also known as MDR-2, is similar to MDR in it's data region identification mechanism. DEPTA uses visual information to build the HTML tag tree, to eliminate the problems caused by HTML tag structure

irregularities. DEPTA uses tree edit distance for identifying similar data records. A single data record may be composed of multiple sub-trees due to noisy information. MDR may find wrong combinations of sub-trees. DEPTA [15] makes use of the visual gaps between data records to deal with the above problem.

However, as in [10], [15] it also requires content analysis to identify the main data region from the set of data regions obtained. Further, it also relies on TABLE tags for identifying data regions.

This technique proposes a two-step strategy.

Step 1. Identifying individual data records in a page.

Step 2. Aligning and extracting data items from the identified data records.

Specifically, this method also uses visual cues to find data records. Since DEPTA is tag dependent, it requires considerable time in building tag tree, traversing whole tag tree and string comparison

The above automatic methods are tag dependant, incorporate time-consuming tag tree construction, and make many assumptions which do not always hold good for all web pages. The proposed method does not make any such assumptions and can scale well for almost all web pages. It is also independent of the type of tags and eliminates the time-consuming tag tree construction procedure.

III. THE PROPOSED TECHNIQUE

The system model of the proposed technique which is a combination of VSAP and EDIP is shown schematically in Fig 1.

It consists of the following components.

VSAP:

- Parsing and Rendering Engine.
- Largest Rectangle Identifier
- Container Identifier
- Data Region Identifier

EDIP:

- Data record extractor.
- Item extractor.

The HTML page is input to the VSAP sub system, which extracts the relevant data regions and these data regions are input to the EDIP subsystem which extracts the data item from them. The output of each component is the input for the next component. This section focuses on the first step, identification and extraction of data regions.

A. Data Region Extraction

We propose a novel and more effective method to mine the data region in a web page automatically. The algorithm is called VSAP (Visual Structure based Analysis of web Pages). The visual information (i.e., the locations on the screen at which tags are rendered) helps the system in three ways.

a) It enables the system to identify gaps that separate data records, which helps to segment data records correctly, because the gap within a data record (if any) is typically smaller than that in between data records.

b) The visual or display information also contains information about the hierarchical structure of the tags.

c) By the visual structure analysis of the web pages , it can be analyzed that the relevant data region seems to occupy the major central portion of the web page.

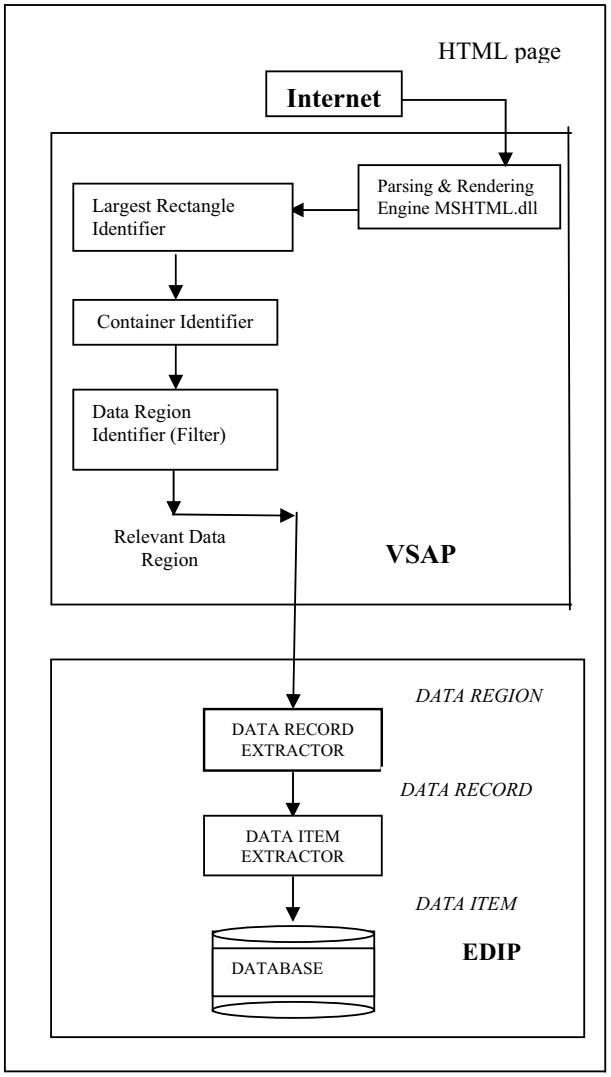

Fig. 1 Proposed System Model

The VSAP technique is based on three observations:

a) A group of data records, that contains descriptions of a set of similar objects, is typically presented in a contiguous region of a page .

b) The area covered by a rectangle that bounds the data region is more than the area covered by rectangles bounding other regions, E.g. Advertisements and links.

c) The height of an irrelevant data record within a collection of data records is less than the average height of relevant data records within that region.

The experiments show that these observations are true.

Definition 1: A data region is defined as the most relevant portion of a web page.

E.g. A region on the product-related web-site that contains a list of products forms the data region.

Definition 2: A data record is defined as a collection of data that together represents a meaningful independent entity.

E.g. A product listed inside a data region on a product-related website is a data record.

Fig 2 illustrates an example, which is a segment of a web page that shows a data region containing list of four books. The full description of each book is a data record.

Fig. 2 An example of a Data Region containing 4 data records

The VSAP algorithm for the proposed technique is as follows:

Algorithm VSAP(*HTML document*)
1. Set maxRect=NULL
2. Set dataRegion=NULL
3. FindMaxRect(*BODY*);
4. FindDataRegion(*maxRect*);
5. FilterDataRegion(*dataRegion*);

end

The lines 1 and 2 specify initializations. The line 3 finds the largest rectangle within the container. Line 4 identifies the data region which consists of the relevant data region and some irrelevant regions also. Line 5 identifies the actual relevant data region by filtering the bounding irrelevant regions.

The two main steps of VSAP algorithm, namely, i) determination of bounding rectangles and ii) identification of data regions are explained below.

B. Determination of Bounding Rectangles

In the first step of the proposed technique, we determine the co-ordinates of all the bounding rectangles in the web page. The VSAP approach uses the MSHTML parsing and rendering engine of Microsoft Internet Explorer 6.0. This parsing and rendering engine of the web browser gives us these co-ordinates of a bounding rectangle.

The MSHTML parsing and rendering engine is the main HTML component of the Microsoft Internet Explorer web browser. The IE rendering engine is nothing more than a COM component. The rendering Engine of the browser produces the boundary coordinates.

We scan the HTML file for tags. For each tag encountered, we determine the co-ordinate of the top-left corner, height and width of the *bounding rectangle* of that tag.

Definition: Every HTML tag specifies a method for rendering the information contained within it. For each tag, there exists an associated rectangular area on the screen. Any information contained within this rectangular area obeys the rendering rules associated with the tag. This rectangle is called the *bounding rectangle* for the particular tag.

A bounding rectangle is constructed by obtaining the co-ordinate of the top-left corner of the tag, the height and the width of that tag. The left and top co-ordinates of the tag are obtained from the offsetLeft and offsetTop properties of the HTMLObjectElement. These values are with respect to it's parent tag. The height and width of that tag are available from the offsetHeight and offsetWidth properties of the HTMLObjectElement Class.

The Fig. 3 shows a sample web page of a product-related website, which contains the list of books and their descriptions which form the data records inside the data region.

Fig. 3 A Sample Web Page of a product related web-site

For each HTML tag on the web page, there exists an associated rectangular area on the screen, which forms the bounding rectangle for that specific tag. Fig 4 shows the bounding rectangles for the <TD> tags of the web page shown in Fig 3.

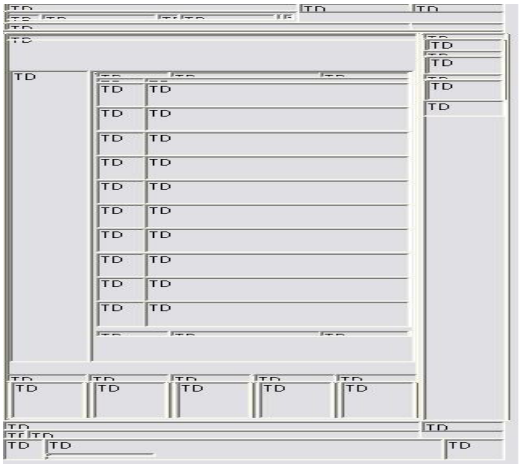

Fig. 4 Bounding rectangles for <TD> tag corresponding to the web page in Fig. 3

Figure 5 shows the top-left corner's co-ordinate, height and width obtained for constructing the bounding rectangle of a <TD> tag. The bounding rectangle of that tag is shaded. The co-ordinates (80, 45), specify the corresponding height and width of the bounding rectangle of the <TD> tag, respectively.

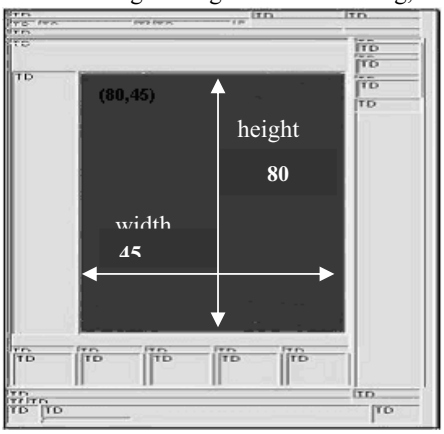

Fig. 5 Bounding rectangle of <TD> tag

C. Identification of Data regions

The second step of the proposed technique is to identify the data region of the web page. The data region is the most relevant portion of a web page that contains a list of data records.

The three steps involved in identifying the data region are:

Step i) Identify the largest rectangle.
Step ii) Identify the container within the largest rectangle.
Step iii) Identify the data region containing the data records within this container.

These steps are explained below

(i) Identification of the largest rectangle:

Based on the height and width of bounding rectangles obtained in the previous step, we determine the area of the bounding rectangles of each of the children of the BODY tag. We then determine the largest rectangle amongst these bounding rectangles. The reason for doing this is due to the observation that the largest bounding rectangle will always contain the most relevant data in that web page. Thus by determining the largest rectangle, we can thereby obtain a superset of the data region. In Fig 6 the largest rectangle is being shown with a dotted border.

The procedure *FindMaxRect* identifies the largest rectangle amongst all the bounding rectangles of the children of the BODY tag. It is as follows.

Procedure FindMaxRect (BODY)
for each child of BODY tag
Begin
find the co-ordinates of the bounding rectangle for the child
if the area of the bounding rectangle > area of maxRect
then
maxRect = child
endif
end

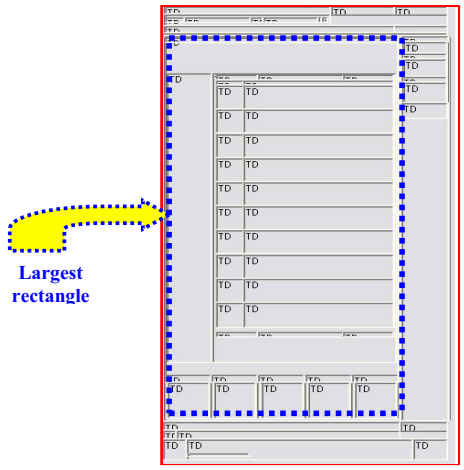

Fig. 6 Largest Rectangle amongst bounding rectangles of Children of BODY tag

(ii) Identification of the container within the largest rectangle:

Once we have obtained the largest rectangle, we form a set of all the bounding rectangles whose area is more than half the area of the largest rectangle. The rationale behind this is that the most important data of a web page must occupy a significant portion of the web page. Next, we determine the bounding rectangle having the smallest area in this set. The reason for determining the smallest rectangle within this set is that the smallest rectangle will only contain data records. Thus a container is obtained. It contains the data region and some irrelevant data.

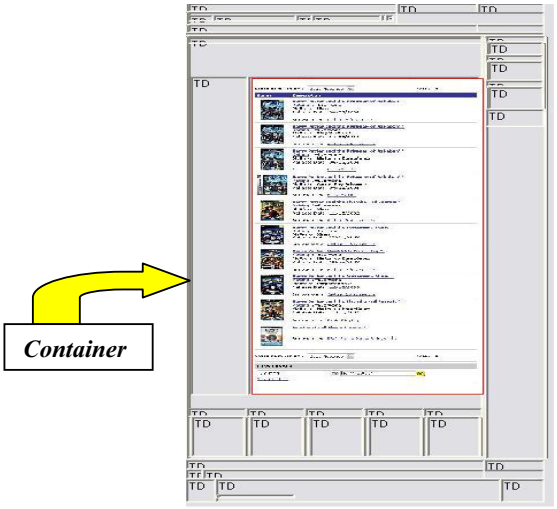

Fig. 7 The container identified from sample web page in Fig. 3

Definition: A container is a superset of the data region which may or may not contain irrelevant data.

For example, the irrelevant data contained in the container may include advertisements on the right and bottom of the

page and the links on the left side.

The Fig 7 shows the container identified from the web page shown in Fig 3.

The procedure *FindDataRegion* identifies the container in the web page which contains the relevant data region along with some irrelevant data also. It is as follows:

Procedure FindDataRegion (*maxRect*) ListChildren=depth first listing of the children of the tag associated with maxRect
for each tag in ListChildren
Begin
If area of bounding rectangle of tag > half the area of maxRect **then**
If area of bounding rectangle dataRegion > area of bounding rectangle of tag **then**
dataRegion = tag
endif.
endif
end

The Fig 8 shows the enlarged view of the container shown in the Fig 7. We note that there is some irrelevant data, both on the top as well as the bottom of the actual data region containing the data records.

Fig. 8 The Enlarged view of the Container shown in Fig. 7

(iii) Identification of data region containing data records within the container:

To filter the irrelevant data from the container, we use a filter. The filter determines the average heights of children within the container. Those children whose heights are less than the average height are identified as irrelevant data and are filtered off. The Fig 9 shows a filter applied on the container in Fig 8, in order to obtain the data region. We note that the irrelevant data in this case is on the top and bottom of the container, which are being removed by the filter.

The procedure FilterDataRegion filters the irrelevant data from the container, and gives the actual data region as the output. It is as follows:

Procedure FilterDataRegion (dataRegion)
totalHeight = 0
for each child of dataRegion
totalHeight += height of the bounding rectangle of child

avgHeight = totalHeight / no of children of dataRegion
for each child of dataRegion
Begin
If height of child's bounding rectangle < avgHeight **then**
Remove child from dataRegion
endif
end

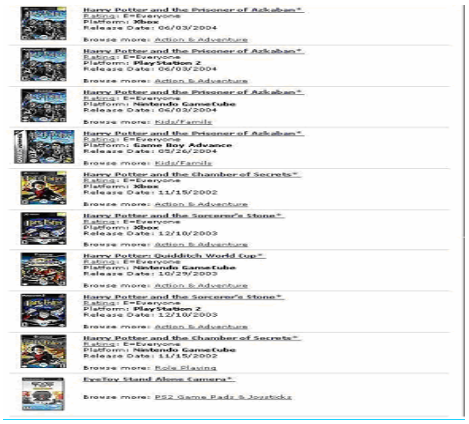

Fig. 9 Data Region obtained after filtering the Container in Fig. 8

The VSAP technique, as described above, is able to mine the relevant data region containing data records from the given web page efficiently. The extracted relevant data region will become an input to EDIP algorithm, the proposed technique for data extraction.

D. Data Extraction

We present a more effective method to extract data items from the data regions of a web page automatically. The algorithm is called EDIP (Extraction of Data Items from Web Pages) which is a list based approach. It finds the data items formed by all types of tags.

We describe the EDIP algorithm by considering a sample webpage shown in Fig. 10. When a web page having description of some products is given to VSAP, it identifies and extracts the data region based on visual clue information of web pages. All the noise on a given web page is eliminated using a filter. The filtered data region is shown in Fig. 11. This region is given as the input to the EDIP algorithm for data extraction.

The EDIP algorithm of the proposed technique for the extraction of data items from a data region is given below:

Algorithm EDIP (DATA REGION)
1. FindDataRecord(DataRegion);
2. FindDataItem(DataRecord);

end

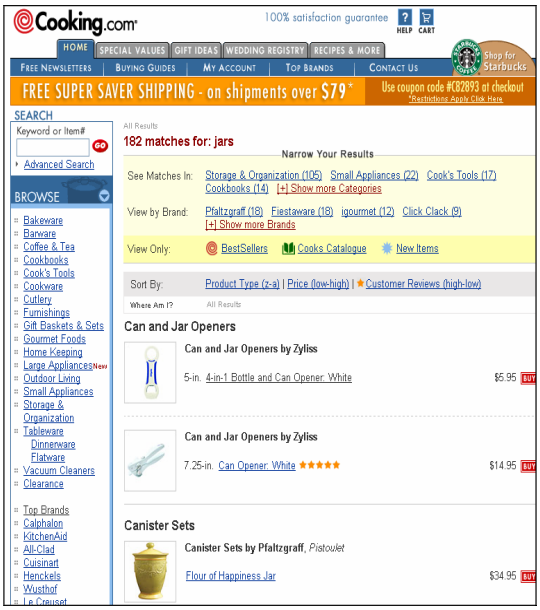

Fig. 10 A Sample Web Page of product related Web Site

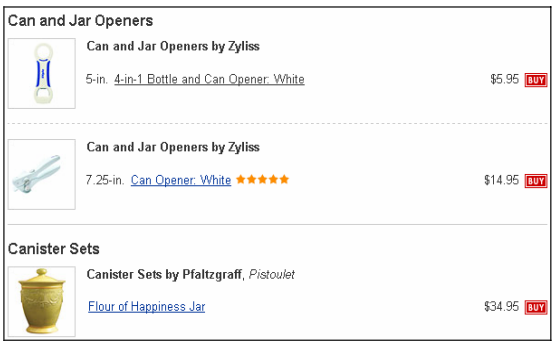

Fig. 11 Filter Data Region

The line 1 finds the data record from the data region. Line 2 identifies and extracts the data items from the data records. This algorithm has two main steps which are presented below.

E. Identification of Data record

The procedure FindDataRecord identifies the tags in the HTML code of the data region which contains the relevant records and is given below:

Procedure FindDataRecord(*DataRegion*)
for each tag in DataRegion till the **EOF**
If Tag = <**Table**> tag
Extract Contents till </Table> tag
Store it in a Record
ExtractDataItems(Record**)**
endif
end

The above procedure extracts a data record from the Data Region. The file containing the data region is scanned for the tags. If the tag is a <table> tag then this marks the beginning of a record and the procedure for extracting the data items from a data record is called.

F. Identification and extraction of data items

The second step is the item extraction from each data record. The algorithm for extracting data items is as follows:

Procedure ExtractDataItems(Record**)**
for each record of the DataRegion
If (FirstRecord)
creatList(firstRecord)
Else CompAndUpdateList()
Endif
Store Extracted Data Items into Database
End

In the ExtractDataItems procedure, the data items of each record are extracted. The technique initially takes one data record with data fields. The content following each <TD> tag or the content following the tag after <TD> tag will be taken as an individual data item. The different tags usually found in a web page after <TD> are <IMG>, <A>, <NOBR>, <B>, <I>, <SCRIPT>, <SPAN>, <STRIKE>, etc. A linked list is created containing all the data items. The createList procedure for creation of a list is given below:

Procedure creatList (firstRecord**)**
While (get character from first record != **EOF**)
{If (character =‘<‘)
{String=check next string till ‘>’;
If (String=“**TD**”)
{Store the string after the ‘>’ and which is not a tag in the node of the linked list; }
String2=**get next character**;
If (String2=“**IMG**”)
{**Extract** the **path** information and store it in a node ;}
Else if (String2=“**B**” or “**I**” or “**S**”)
{**Extract the information after it** and store it in a node; }
Else if (String2=“**A**”)
{**Extract** the content after the tag and store it in a node ;}
}
End

Initially the first record is extracted and a list is created as shown in Fig.12

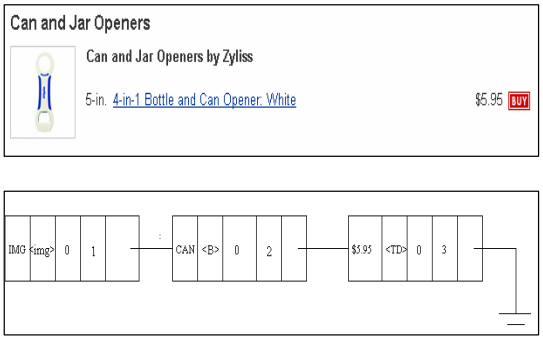

Fig. 12 Creation of a list of data items from the Record of the Data Region.

For subsequent records of the data region, as and when the items are extracted, they are compared with the list and updated accordingly. This step is carried out to ensure that none of the data items in any of the record are left unaddressed. At the end of this procedure, a master list of data items is obtained. The data items are then transferred into the database. The procedure for CompAndUpdateList is given below.

```
Procedure CompAndUpdateList()
{Temp=head
  If (compare temp->tag =new->tag)
{While(compare temp->tag!=new->tag&temp->link !=null)
        {If (temp->bit==1)
                temp=temp->link; }
                If (temp->link=NULL)
                Create node at the end ;
        }
        temp=temp->link;
}
    End
```

This procedure compares the tags of the data field of the second data record with that of the first data record. If the same tags are found then the bit value is checked whether it is set or not. If it is set then that node is skipped. If it is not set then the comparison process is continued till that tag is encountered. If no such tag is found then a new node containing that tag is created and appended to the end of the linked list. This is done so that generalized numbers of data fields are obtained.

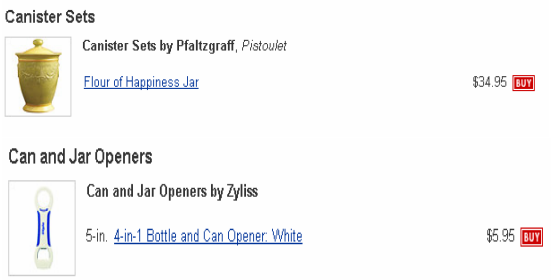

Fig. 13 Comparisons and Updation of a list of data items from the remaining Records of the Data Region.

The Fig.13 is an example to depict the comparison of two records of a data region. Here a new node is added after comparison since that data item is not present in the first record. The outcome of this step is a list that contains all the items of the data region.

IV. Empirical Evaluation

In this section, we evaluate the proposed technique. We also compare it with two state-of-the-art existing systems, DEPTA [15] (which is an improvement of [10]) and MDR [10]. We do not compare it with the method in [5] and the method in [6] here as it is shown in [10] that MDR is already more effective than them.

The evaluation consists of three aspects as discussed in the following:

1. *Data Region Extraction*:
   We compare the first step of [10] with our system for identifying the data regions.
   Both MDR [10] and DEPTA [15], are dependent on certain tags like <TABLE>, <TBODY>,etc. for identifying the data region. But, a data region need not be always contained only within specific tags like <TABLE>, <TBODY>, etc. A data region may also be contained within tags other than table-related tags like <P>, <LI>, <FORMS> etc.
   In the proposed VSAP system, the data region identification is independent of specific tags and forms. Unlike [10], where an incorrect tag tree may be constructed due to the misuse of HTML tags, there is no such possibility of erroneous tag tree construction in case of VSAP, because the hierarchy of tags is constructed based on the visual cues on the web page.
   In case of [10] and [15] , the entire tag tree needs to be scanned in order to mine data regions, but VSAP does'nt scan the entire tag tree, but it only scans the largest child of the <BODY> tag. Hence, this method proves very efficient in improving the time complexity compared to other contemporary algorithms.
2. *Data Record Extraction*:
   We compare the record extraction step of MDR [10] with VSAP:
   MDR identifies the data records based on keyword search (e.g , " $ "). But VSAP is purely dependent on the visual structure of the web page only. It does not make use of any text or content mining. This proves to be very advantageous as it overcomes the additional overhead of performing keyword search on the web page.
   MDR, not only identifies the relevant data region containing the search result records but also extracts records from all the other sections of the page, e.g, some advertisement records also, which are irrelevant.
   MDR also sets a similarity threshold value of 0.3 as the default value for the system to identify similar data records, which has been set based upon a number of training pages. Hence, it doesn't scale well with all web pages.
   In MDR, comparison of generalized nodes is based on string comparison using normalized edit distance method. In DEPTA, comparison of sub-trees is made by simple tree matching algorithm. However, both these methods are slow and inefficient as compared to VSAP where the comparison is purely numeric, since we are comparing the co-ordinates of the two bounding rectangles. It scales well with all the web pages.
   A single data record may be composed of multiple sub-trees. Due to noisy information, MDR may find wrong combinations of sub-trees. In VSAP system, visual gaps between data records help to deal with this problem.

3. *Overall Time Complexity*:

Complexity of VSAP is much lesser than the existing algorithms. The existing algorithm MDR [10] has time complexity of the order O (NK) without considering string comparison, where N is the total number of nodes in the tag tree and K is the maximum number of tag nodes that a generalized node can have (which is normally a small number <10). DEPTA [15] has the time complexity of the order O (k2) without considering tree matching, where k is the number of trees. Our algorithm VSAP has a complexity of the order of O(n), where n is the number of tag-comparisons made. Although, the complexities of these algorithms are expressed in different parameters, the comparison of dominant operations in each of them leads to the conclusion that VSAP is a time saving algorithm.

V. Experimental Results

In running MDR, we used their default settings. MDR system was downloaded at http://www.cs.uic.edu/~liub/MDR/MDR-download.htm.

We use the *recall* and *precision* measures (which are widely used to evaluate information retrieval systems) to evaluate the performance of VSAP system for extracting data records. Recall and precision are defined below:

Recall = Ec / Nt and *Precision* = *Ec* / Et

where, Ec is the total number of correctly extracted records , Nt is the total number of records on the page , and Et is the total number of records extracted. *Recall* defines the correctness of the data records identified and *precision* is the percentage of the relevant data records identified from the web page.

The experimental results are given in Table 1. Correct results (Cor.) are relevant data records present on the page that have been correctly identified. Wrong results (Wr.) are irrelevant data records that have been incorrectly identified. The results obtained after running both MDR and VSAP are shown in table 1. The successes of both algorithms are compared in terms of precision and recall.

Column 1 : It lists the URL of each site. In some sites more than one page are tried (which have different data record formats). All our experimental web pages are selected randomly. Due to long URLs of most pages, we could not list them here.

Columns 2 and 4 : They give the numbers of correct (Cor.) records extracted by MDR and VSAP from the pages of each site respectively. These data records are the relevant ones (e.g., product lists). They do not include navigation areas, advertisements etc, which may also have regular patterns. MDR cannot handle nested data records (records within records), but VSAP is able to handle such data records also.

Columns 3 and 5: They give the numbers of data records extracted wrongly(Wr.) by MDR and VSAP , from the pages of each site respectively. *x/y* means that *x* is the number of extracted results that are wrong , and *y* is the number of results that are not extracted.

Table I
Comparison of VSAP and MDR

URL	MDR		VSAP	
	Cor.	Wr.	Cor.	Wr.
http://www.tigerdirect.com/….	8	36/ 0	8	0/0
http://www.amazon.com/….	0	17/25	25	1/0
http://www.cooking.com/….	17	13/3	20	0/0
http://www.ebay.com/…..	25	30/0	25	0/0
http://www.powells.com/…..	9	47/1	10	5/0
http://www.barnesandnoble.com/….	10	30/0	10	0/0
http://www.pricegrabber.com/…..	0	0/25	25	0/0
http://www.shoebuy.com/….	12	12/84	96	0/0
http://www.smartbuy.com/…..	0	15/10	10	0/0
http://www.reviews.cnet.com/…..	24	38/1	25	3/0
http://www.nothingbutsoftware.com/	10	12/0	10	0/0
http://www.refurbdepot.com/….	0	6/15	15	0/0
http://www.drugstore.com/….	15	14/0	15	0/0
http://www.bookpool.com/….	10	7/0	10	0/0
http://www.target.com/……	0	0/12	12	1/0
Total	140	277 / 176	316	10 / 0
Recall	33.5%		96.93%	
Precision	44.3%		100%	

The results shown are obtained without using any form of content search. All the URLs given are obtained from the search functionality in the browser, where the search query for all URLs is “watch” (except for test case 12 where “camera” was specified, test case 14 where “computer” was specified and 8 where “men’s” was specified).

We note that test cases. 7 and 15 hanged in case of MDR and hence have been considered as missing the correct data records. It has been experimentally found out that MDR fails to identify a single data record where as VSAP successfully identifies it for the URL,

http://www.nothingbutsoftware.com/SiteSearch.asp?db=12719&query=watches

The last three rows of Table 1 give the total of each column, the recall and precision of each system. For MDR and VSAP, the recall and precision are computed based on the total number of data records found in all pages and the actual number of data records in these pages. We see that the data record extraction is highly effective. We also observe that VSAP performs significantly better than MDR.

The results obtained after running EDIP are presented in Table 2 for few URL’s (however the proposed algorithm was applied for all URL’s proposed in DEPTA [16]. It has been found the results in all the cases are consistent with the conclusion that can be obtained from given data in table 2.). The success of both algorithms is measured in terms of precision and recall.

The DEPTA fails to identify a single data record for the URL, http://www.cooking.com/, while EDIP successfully identifies it. The last three rows of Table 2 give the total of each column, the recall and precision of each system. One can see that the data item extraction is highly effective with EDIP and is also observed that EDIP performs significantly better than DEPTA in all aspects.

TABLE II
COMPARISON OF DEPTA AND EDIP

URL	DEPTA		EDIP	
	Cor.	Wr.	Cor	Wr
http://www.tigerdirect.com/….	43	0/ 0	43	0/0
http://www.amazon.com/….	132	0/11	132	1/3
http://www.cooking.com/….	35	0/2	35	0/0
http://www.google.froogle.com/….	10	1/0	10	0/0
http://www.nothingbutsoftware.com/	260	4/0	260	2/0
http://www.alibris.com/….	163	15/0	163	5/0
Total	643	20/13	643	8/3
Recall	95.11%		98.31%	
Precision	96.98%		98.77%	

The success of EDIP is because of the following points: EDIP is independent of errors occurring due to misuse of HTML tags. The overall time complexity of the EDIP algorithm is much better than the existing approaches. Thus the EDIP method is shown to be better compared to earlier methods in terms of tag dependency and time complexity.

VI. CONCLUSION

In this paper, we have proposed a new approach to extract structured data from web pages. Although the problem has been studied by several researchers, existing techniques are either inaccurate or make many strong assumptions. A novel and effective method based on a combination of VSAP and EDIP is proposed to mine the data region and data items in a web page automatically. It is a pure visual structure oriented and list based method that can correctly identify the data region and data items. The proposed technique exploits the merits of both VSAP and EDIP leading to better performance in data extraction from a webpage. The experimental results demonstrate the effectiveness of the proposed method.

SCOPE FOR FUTURE WORK

Extraction of the data fields from the data records contained in these mined data regions will be considered in the future work taking also into account the complexities such as the web pages featuring dynamic html, etc. The extracted data can be put in some suitable format and eventually stored back into a relational database. Thus, data extracted from each web-page can then be integrated into a single collection. This collection of data can be further used for various Knowledge Discovery Applications, E.g. making a comparative study of products from various companies, smart shopping, etc.

ACKNOWLEDGMENT

The authors are grateful to the referees for their suggestions and useful comments.

REFERENCES

[1] Baeza Yates, R. Algorithms for string matching: A survey. *ACM SIGIR Forum, 23(3-4): 34—58, 1989.*

[2] J. Hammer, H. Garcia Molina, J. Cho, and A. Crespo. Extracting semi-structured information from the web.*In Proc.of the Workshop on the Management of Semi-structured Data, 1997.*

[3] D. Embley, Y. Jiang, and Y. K. Ng. Record-boundary discovery in Web documents. *ACM SIGMOD Conference, 1999.*

[4] Kushmerick, N. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence*, 118:15-68, 2000. Clustering-based Approach to Integrating Source Query].

[5] Chang, C-H., Lui, S-L. IEPAD: Information Extraction Based on Pattern Discovery. *WWW-01*, 2001.]

[6] Crescenzi, V., Mecca, G. and Merialdo, P. ROADRUNNER: Towards Automatic Data Extraction from Large Web Sites. *VLDB-01*, 2001.]

[7] Eying, H. Zhang. HTML Page Analysis based on Visual Cues. *6th International Conference on Document Analysis and Recognition, 2001.*

[8] D. Buttler, L. Liu, C. Pu. A Fully Automated Object Extraction System for the World Wide Web. *International Conference on Distributed Computing Systems (ICDCS 2001), 2001.*

[9] Bing Liu , Kevin chen-chuan chang, Editorial: Special issue on web content mining, *WWW 02, 2002.*

[10] Liu, B., Grossman, R. and Zhai, Y. Mining Data Records in Web Pages. *KDD-03, 2003.*

[11] Cai, D., Yu, S., Wen, J.-R. and Ma, W.-Y. (2003). Extracting Content Structure for Web Pages based on Visual Representation, *Asia Pacific Web Conference (APWeb 2003), pp. 406417.*

[12] A. Arasu, H. Garcia-Molina, Extracting structured data from web pages, *ACM SIGMOD 2003, 2003.*

[13] J. Wang, F. H Lochovsky. Data Extraction and Label Assignment for Web Databases.*WWW conference, 2003.*

[14] H. Zhao, W. Meng, Z. Wu, Raghavan, Clement Yu. Fully Automatic Wrapper Generation For Search Engines, *International WWW conference 2005, May 10-14,2005, Japan. ACM 1-59593-046-9/05/005.*

[15] Zhai, Y., Liu, B. Web Data Extraction Based on Partial Tree Alignment, *WWW-05, 2005, May 10-14, 2005, Chiba, Japan. ACM 1-59593-046-9/05/00.*

[16] Hiremath P.S, Benchalli S.S, Algur Siddu P, Minig Data Regions from Web Pages, COMMAD 2005b.

[17] Algur Siddu P, Hiremath P.S, Extraction of Data from Web – Some Aspects, IICT – 2007

Dr. P.S. Hiremath Professor and Chairman, Department of P. G. Studies and Research in Computer Science, Gulbarga University, Gulbarga-585106, Karnataka, INDIA. He has obtained M.Sc. degree in 1973 and Ph.D. degree in 1978 in Applied Mathematics from Karnatak University, Dharwad. He had been in the Faculty of Mathematics and Computer Science of various Institutions in India, namely, National Institute of Technology, Surathkal (1977-79), Coimbatore Institute of Technology, Coimbatore (1979-80), National Institute of Technology, Tiruchinapalli (1980-86), Karnatak University, Dharwad (1986-1993) and has been presently working as Professor of Computer Science in Gulbarga University, Gulbarga (1993 onwards). His research areas of interest are Computational Fluid Dynamics, Optimization Techniques, Image Processing and Pattern Recognition, Data Mining ,Web Mining, Knowledge Discovery Techniques . He has published 52 research papers in peer reviewed International Journals.
Tel (off): +91 8472 249682, e-mail: hiremathps@hotmail.com, Fax: +91 8472 245927.

Siddu P. Algur is currently pursuing his Ph.d. in Computer Science in the Department of P.G. Studies and Research in Computer Science at Gulbarga University, Gulbarga. He received B.E. degree in Electrical and Electronics from Mysore University, Karnataka, India, in 1986. He received his M.E. degree in Information Science and Engineering from NIT, Allahabad, India, in 1991. Mr. Algur worked as Lecturer in the department of E & E at KLE Society's College of Engineering and Technology from September 1986 to October 1992. From October 1992 to August 2007 he worked as Lecturer and Assistant Professor in the department of Computer Science and Engineering at SDM College of Engineering and Technology, Dharwad. Presently he is working as a Professor in the department of Information Science and Engineering at BVB College of Engineering and Technology, Hubli. His research interest includes Data Mining, Web Mining, Information Retrieval from the web, Knowledge discovery techniques.