ISSN:0975-9646

Deep Web Data Extraction by Using Vision-Based Item and Data Extraction Algorithms

B.Sailaja
II nd year M.Tech,
Dept of CSE,AIET

Ch.Kodanda Ramu
Asst. Professor,
Dept of CSE, AIET

Y.Ramesh Kumar
Assoc. Professor,
Dept of CSE, AIET

Abstract: Deep Web contents are accessed by queries submitted to Web databases and the returned data records are enwrapped in dynamically generated Web pages (they will be called deep Web pages in this paper). Extracting structured data from deep Web pages is a challenging problem due to the underlying intricate structures of such pages. Until now, a large number of techniques have been proposed to address this problem, but all of them have inherent limitations because they are Web-page-programming-language dependent. As the popular two-dimensional media, the contents on Web pages are always displayed regularly for users to browse. This motivates us to seek a different way for deep Web data extraction to overcome the limitations of previous works by utilizing some interesting common visual features on the deep Web pages.

In this, a novel vision-based approach that is Web-page programming- language-independent is proposed. This approach primarily utilizes the visual features on the deep Web pages to implement deep Web data extraction, including data record extraction and data item extraction. We also propose a new evaluation measure revision to capture the amount of human effort needed to produce perfect extraction. Our experiments on a large set of Web databases show that the proposed vision-based approach is highly effective for deep Web data extraction.

Keywords: Deep web extraction, vision based approach, enwrapped, item extraction, record extraction.

INTRODUCTION

The World Wide Web has more and more online Webatabases which can be searched through their WebQuery interfaces. The number of Web databases hasReached 25 millions according to a recent survey [21].All the Web databases make up the deep Web (hiddenWeb or invisible Web). Often the retrieved information(query results) is enwrapped in Web pages in the form of Data records. These special Web pages are generated Dynamically and are hard to index by traditional crawlerbased Search engines, such as Google and Yahoo. In this Paper, we call this kind of special Web pages deep Web Pages. Each data record on the deep Web pages corresponds To an object. For instance, Fig. 1 shows a typical Deep Web page from Amazon.com. On this page, the Books are presented in the form of data records, and each Data record contains some data items such as title, author, Etc. In order to ease the consumption by human users, Most Web databases display data records and data items Regularly on Web browsers.However, to make the data records and data items in Them machine processable, which is needed in many Applications such as deep Web crawling and metasearching, The structured data need to be extracted from the deep Web Pages. In this paper, we study the problem of automatically Extracting the structured data, including data records and Data items, from the deep Web pages. The problem of Web data extraction has received a lot of attention in recent years and most of the proposed solutions are based on analyzing the HTML source code or the tag Trees of the Web pages (see Section 2 for a review of these Works). These solutions have the following main limitations: First, they are Web-page-programming-languagedependent, Or more precisely, HTML-dependent. As most Web pages are written in HTML, it is not surprising that all Previous solutions are based on analyzing the HTML source Code of Web pages. However, HTML itself is still evolving (from version 2.0 to the current version 4.01, and version 5.0 Is being drafted [14]) and when new versions or new tags Are introduced, the previous works will have to be Amended repeatedly to adapt to new versions or new tags. Furthermore, HTML is no longer the exclusive Web page Programming language, and other languages have been Introduced, such as XHTML and XML (combined with XSLT and CSS)..

The previous solutions now face the Following dilemma: should they be significantly revised or Even abandoned? Or should other approaches be proposed To accommodate the new languages? Second, they are Incapable of handling the ever-increasing complexity of HTML source code of Web pages. Most previous works Have not considered the scripts, such as javascript and CSS,In the HTML files. In order to make Web pages vivid and Colorful, Web page designers are using more and more Complex javascript and CSS.Based on our observation from A large number of real Web pages, especially deep WebPages, the underlying structure of current Web pages is More complicated than ever and is far different from their Layouts on Web browsers. This makes it more difficult for existing solutions to infer the regularity of the structure of Web pages by only analyzing the tag structures. Meanwhile, to ease human users' consumption of the information retrieved from search engines, good template designers of deep Web pages always arrange the data records and the data items with visual regularity to meet the reading habits of human beings.

For example, all the data ecords in Fig. 1 are clearly separated, and the data items of the same semantic in different data records are similar on layout and font. In this paper, we explore the visual regularity of the data records and data items on deep Web pages and propose a novel vision-based approach, Vision-based Data extractor (ViDE), to extract structured results from deep Web pages automatically. ViDE is primarily based on the visual features human users can capture on the deep Web pages while also utilizing some simple nonvisual information such as data types and frequent symbols to make the solution more robust. ViDE consists of two main

components, Visionbased Data Record extractor (ViDRE) and Vision-based Data Item extractor (ViDIE). By using visual features for data extraction, ViDE avoids the limitations of those solutions that need to analyze complex Web page source files. Our approach employs a four-step strategy.

First, given a sample deep Web page from a Web database, obtain its visual representation and transform it into a Visual Block tree which will be introduced later; second, extract data records from the Visual Block tree; third, partition extracted data records into data items and align the data items of the same semantic together; and fourth, generate visual wrappers (a set of visual extraction rules) for the Web database based on sample deep Web pages such that both data record extraction and data item extraction for new deep Web pages that are from the same Web database can be carried out more efficiently using the visual wrappers. To our best knowledge, although there are already some works [3], [4], [12], [6], [8] that pay attention to the visual information on Web pages, our work is the first to perform deep Web data extraction using primarily visual features

1. The problem of Web data extraction has received a lot of attention in recent years and most of the proposed solutions are based on analyzing the HTML source code or the tag trees of the Web pages.
2. These solutions have the following main limitations: First, they are Web-page-programming-language dependent, or more precisely, HTML-dependent.
3. As most Web pages are written in HTML, it is not surprising that all previous solutions are based on analyzing the HTML source code of Web pages.
4. However, HTML itself is still evolving and when new versions or new tags are introduced, the previous works will have to be amended repeatedly to adapt to new versions or new tags.

2. METHODOLOGY

We explore the visual regularity of the data records and data items on deep Web pages and propose a novel vision-based approach, Vision-based Data Extractor (ViDE), to extract structured results from deep Web pages automatically. ViDE is primarily based on the visual features human users can capture on the deep Web pages while also utilizing some simple non visual information such as data types and frequent symbols to make the solution more robust. ViDE consists of two main components, Vision based Data Record extractor (ViDRE) and Vision-based Data Item extractor (ViDIE).

2.2 Strategies:

1. a sample deep Web page from a Web database, obtain its visual representation and transform it into a Visual Block tree which will be introduced later;
2. Extract data records from the Visual Block tree;
3. Partition extracted data records into data items and align the data items of the same semantic together; and Generate visual wrappers (a set of visual extraction rules) for the Web database based on sample deep Web pages.

Our approach is independent of any specific Web page programming language. Although our current implementation uses the VIPS algorithm [4] to obtain a deep Web page's Visual Block tree and VIPS needs to analyze the HTML source code of the page, our solution is independent of any specific method used to obtain the Visual Block tree in the sense that any tool that can segment the Web pages into a tree structure based on the visual information, not HTML source code, can be used to replace VIPS in the implementation of ViDE. In this paper, we also propose a new measure, revision, to evaluate the performance of Web data extraction tools.

It is the percentage of the Web databases whose data records or data items cannot be perfectly extracted (i.e., at least one of the precision and recall is not 100 percent). For these Web databases, manual revision of the extraction rules is needed to achieve perfect extraction. In summary, this paper has the following contributions:

1) A novel technique is proposed to perform data extraction from deep Web pages using primarily visual features. We open a promising research direction where the visual features are utilized to extract deep Web data automatically.
2) A new performance measure, revision, is proposed to evaluate Web data extraction tools. This measure reflects how likely a tool will fail to generate a perfect wrapper for a site.
3) A large data set consisting of 1,000 Web databases across 42 domains is used in our experimental study. In contrast, the data sets used in previous works seldom had more than 100 Web databases. Our experimental results indicate that our approach is very effective.

3. VISUAL BLOCK TREE AND VISUAL FEATURES

Before the main techniques of our approach are presented, we describe the basic concepts and visual features that our approach needs.

3.1 Visual Information of Web Pages

The information on Web pages consists of both texts and images (static pictures, flash, video, etc.). The visual information of Web pages used in this paper includes mostly information related to Web page layout (location and size) and font.

3.1.1 Web Page Layout

A coordinate system can be built for every Web page. The origin locates at the top left corner of the Web page. The X-axis is horizontal left-right, and the Y-axis is vertical topdown. Suppose each text/image is contained in a minimum bounding rectangle with sides parallel to the axes. Then, a text/image can have an exact coordinate (x, y) on the Web page. Here, x refers to the horizontal distance between the origin and the left side of its corresponding rectangle, while

y refers to the vertical distance between the origin and the upper side of its corresponding box. The size of a text/ image is its height and width. The coordinates and sizes of texts/images on the Web page make up the Web page layout.

3.1.2 Font

The fonts of the texts on a Web page are also very useful visual information, which are determined by many attributes as shown in Table 1. Two fonts are considered to be the same only if they have the same value under each attribute.

3.2 Deep Web Page Representation

The visual information of Web pages, which has been introduced above, can be obtained through the programming interface provided by Web browsers (i.e., IE). In this paper, we employ the VIPS algorithm [4] to transform a deep Web page into a Visual Block tree and extract the visual information.AVisual Block tree is actually a segmentation of a Web page. The root block represents the whole page, and each block in the tree corresponds to a rectangular region on the Web page.

TABLE 1
Font Attributes and Examples

Font factor	Example	Font factor	Example
Size	A (10pt)	underline	A
face	A(Sans Serif)	italic	*A*
color	A (red)	weight	**A**
strikethrough	~~A~~	frame	A

The leaf blocks are the blocks that cannot be segmented further, and they represent the minimum semantic units, such as continuous texts or images. Fig. 2a shows a popular presentation structure of deep Web pages and Fig. 2b gives its corresponding Visual Block tree. The technical details of building Visual Block trees can be found in [4]. An actual Visual Block tree of a deep Web page may contain hundreds even thousands of blocks. Visual Block tree has three interesting properties.

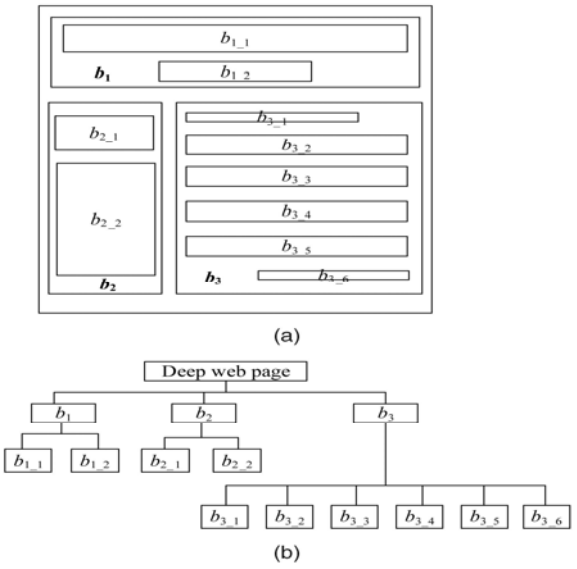


Fig. 2. (a) The presentation structure and (b) its Visual Block tree.

First, block a contains block b if a is an ancestor of b. Second, a and b do not overlap if they do not satisfy property one. Third, the blocks with the same parent are arranged in the tree according to the order of the corresponding nodes appearing on the page. These three properties are illustrated by the example in Fig. 2. The formal representations for internal blocks and leaf blocks in our approach are given below. Each internal block a is represented as a ¼ ðCS; P; S; FS; ISÞ, where CS is the set containing its child blocks (note that the order of blocks is also kept), P is the position of a (its coordinates on the Web page), S is its size (height and width), FS is the set of the fonts appearing in a, and IS is the number of images in a. Each leaf block b is represented as b ¼ ðP; S; F; L; I;CÞ, where the meanings of P and S are the same as those of an inner block, F is the font it uses, L denotes whether it is a hyperlink text, I denotes whether it is an image, and C is its content if it is a text.

3.3 Visual Features of Deep Web Pages

Web pages are used to publish information to users, similar to other kinds of media, such as newspaper and TV. The designers often associate different types of information with distinct visual characteristics (such as font, position, etc.) to make the information on Web pages easy to understand. As a result, visual features are important for identifying special information on Web pages. Deep Web pages are special Web pages that contain data records retrieved from Web
databases, and we hypothesize that there are some distinct visual features for data records and data items. Our observation based on a large number of deep Web pages is consistent with this hypothesis. We describe the main visual features in this section and show the statistics about the accuracy of these features at the end of this Section 3.3.

Position features (PFs). These features indicate the location of the data region on a deep Web page.

. PF1: Data regions are always centered horizontally.

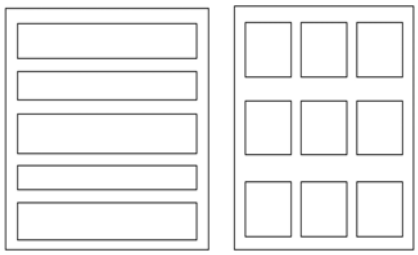
. PF2: The size of the data region is usually large relative to the area size of the whole page.

Since the data records are the contents in focus on deep Web pages, Web page designers always have the region containing the data records centrally and conspicuously placed on pages to capture the user's attention. By investigating a large number of deep Web pages, we found two interesting facts. First, data regions are always located in the middle section horizontally on deep Web pages. Second, the size of a data region is usually large when there are enough data records in the data region. The actual size of a data region may change greatly because it is not only influenced by the number of data records retrieved, but also by what information is included in each data record. Therefore, our approach uses the ratio of the size of the data region to the size
of whole deep Web page instead of the actual size. In our experiments in Section 7, the threshold of the ratio is set at 0.4, that is, if the ratio of the horizontally centered region is greater than or equal to 0.4, then the region is recognized as the data region.

Layout features (LFs). These features indicate how the data records in the data region are typically arranged.

. LF1: The data records are usually aligned flush left in the data region.
. LF2: All data records are adjoining.
. LF3: Adjoining data records do not overlap, and the space between any two adjoining records is the same.
Data records are usually presented in one of the two layout models shown in Fig. 3. In Model 1, the data records are arranged in a single column evenly, though they may be different in width and height. LF1 implies that the data records have the same distance to the left boundary of the data region. In Model 2, data records are arranged in

Fig. 3. Layout models of data records on deep Web pages.



multiple columns, and the data records in the same column have the same distance to the left boundary of the data region. Because most deep Web pages follow the first model, we only focus on the first model in this paper, and the second model can be addressed with minor implementation expansion to our current approach. In addition, data records do not overlap, which means that the regions of different data records can be separated. Appearance features (AFs). These features capture the visual features within data records.
. AF1: Data records are very similar in their appearances, and the similarity includes the sizes of the images they contain and the fonts they use.
. AF2: The data items of the same semantic in different data records have similar presentations
with respect to position, size (image data item), and font (text data item).
. AF3: The neighboring text data items of different semantics often (not always) use distinguishable fonts.
AF1 describes the visual similarity at the data record level. Generally, there are three types of data contents in data records, i.e., images, plain texts (the texts without hyperlinks), and link texts (the texts with hyperlinks). Table 2 shows the information on the three aspects for the data records in Fig. 1. We can see that these five data records are very close on the three aspects.

TABLE 2
Relevant Visual Information about the Top Five Data Records in Fig. 1

	Images (pixel)	plain texts		link texts	
		Total font number	Shared font number	Total font number	Shared font number
record1	115*115	5	5	2	2
record2	115*115	5	5	2	2
record3	115*110	5	5	2	2
record4	115*115	5	5	2	2
record5	115*115	5	5	2	2

AF2 and AF3 describe the visual similarity at the data item level. The text data items of the same semantic always use the same font, and the image data items of the same semantic are often similar in size. The positions of data items in their respective data records can be classified into two kinds: absolute position and relative position. The former means that the positions of the data items of certain semantic are fixed in the line they belong to, while the latter refers to the position of a data item relative to the data item ahead of it. Furthermore, the items of the same semantic from different data records share the same kind of position. AF3 indicates that the neighboring text data items of different semantics often use distinguishable fonts. However, AF3 is not a robust feature because some neighboring data items may use the same font. Neighboring data items with the same font are treated as a composite data item. Composite data items have very simple string patterns and the real data items in them can often be separated by a limited number of symbols, such as “,”, “/,” etc. In addition, the composite data items of the same semantics share the same string pattern. Hence, it’s easy to break composite data items into real data items using some predefined separating symbols. For example, in Fig. 4, four data items, such as publisher, publishing date, edition, and ISBN, form a composite data item, and they are separated by commas. According to our observation to deep Web pages, the granularity of the data items extracted is not larger than what HTML tags can separate, because a composite data item is always included in one leaf node in the tag tree.
Content feature (CF). These features hint the regularity of the contents in data records.
. CF1: The first data item in each data record is always of a mandatory type.
. CF2: The presentation of data items in data records follows a fixed order.
. CF3: There are often some fixed static texts in data records, which are not from the underlying Web database.
The data records correspond to the entities in real world, and they consist of data items with different semantics that describe the attribute values of the entities. The data items can be classified into two kinds: mandatory and optional. Mandatory data items appear in all data records. For example, if every data record must have a title, then titles are mandatory data items. In contrast, optional items may be missing in some data records. For example, “discounted price” for products is likely an optional unit. The order of different types of data items from the same Web database is always fixed in data records. For example, the order of attributes of data records from Bookpool.com in Fig. 4 is “title,” “author,” “publisher,” “publish time,” “edition,” “ISBN,” “discount price,” “save money,” “availability,” etc. Fixed static texts refer to the texts that appear in every data record. Most of them are meaningful labels that can help users understand the semantics of data items, such as “Buy new” in Fig. 4. We call these static texts static items, which are part of the record template. Our deep Web data extraction solution is developed mainly based on the above four types of visual features. PF is used to locate the region containing all the data records on a deep Web page; LF and

AF are combined together to extract the data records and data items.

Statistics on the visual features. To verify the robustness of these visual features we observed, we examined these features on 1,000 deep Web pages of different Web databases from the General Data Set (GDS) used in our experiments (see Section 7 for more information about GDS). The results are shown in Table 3.

TABLE 3
The Statistics on the Visual Features

Feature type		Statistics	Feature type		Statistics
Position Features	*PF1*	99.9%	Appearance Features	*AF1*	99.5%
	PF2	99.9%		*AF2*	100%
				AF3	92.8%
Layout Features	*LF1*	99.3%	Content Features	*CF1*	100%
	LF2	100%		*CF2*	100%
	LF3	100%		*CF3*	6.5%

For most features (except AF3 and CF3), their corresponding statistics are the percentages of the deep Web pages that satisfy them. For example, the statistics of 99.9 percent for PF1 means that for 99.9 percent of the deep Web pages, PF1 feature "data regions are always centered horizontally" is true. From the statistics, we can conclude that these visual features are very robust and can be reliably applied to general deep Web pages. For AF3, 92.8 percent is the percentage of the data items that have different font from their following data items. For CF3, 6.5 percent is the percentage of the static data items over all data items. We should point out that when a feature is not satisfied by a page, it does not mean that ViDE will fail to process this page. For example, our experiments using the data sets to be described in Section 7 show that among the pages that violate LF3, 71.4 percent can still be processed successfully by ViDE, and among the pages that violate AF1, 80 percent can still be correctly processed.

Fig. 4. Illustrating visual features of deep Web pages.

3.4 Special Supplementary Information

Several types of simple nonvisual information are also used in our approach in this paper. They are same text, frequent symbol, and data type, as explained in Table 4. Obviously, the above information is very useful to determine whether the data items in different data records from the same Web database belong to the same semantic. The above information can be captured easily from the Web pages using some simple heuristic rules without the need to analyze the HTML source code or the tag trees of the Web pages. Furthermore, they are specific language (i.e., English, French, etc.) independent.

TABLE 4
Nonvisual Information Use

Special complementary information	Remarks
Same text	Given two texts, we can determine whether or not they are the same.
Frequent symbol	Given the deep web pages of a web database, if some symbols/words (e.g., ISBN, $) appear in all the data items of an attribute, they are called frequent symbols.
Data type	They are predefined, including image, text, number, date, price, email, etc

Algorithm for item alignment:

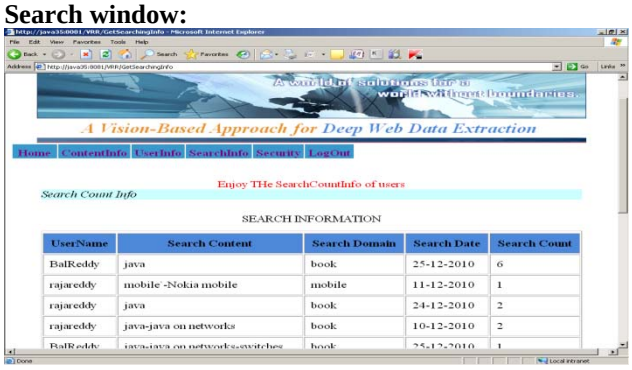
```
Algorithm data item alignment
Input: a set of extracted data records {r1 | 1≤i≤n}
Output: a set of data records {r1 | 1≤i≤n} with all the data items aligned
Begin
1  currentItemSet=φ;
2  currentCluster=φ;
//put the first unaligned data item of each ri into currentItemSet
// Item(k)i refers to the first unaligned item of the ith data record
3  currentItemSet ← Item(k)i (1≤i≤n);
4  while currentItemSet≠φ
5      use the data item matching algorithm to group the data items
in currentItemSet into k clusters {Ci | 1≤i≤k} (k≥1);
6      for each cluster Ci
7          for each ri that does not have a data item in Ci
8              if Item(k)i is matched with data items in Ci
9                  Log position k;
10         else
11                 Log position 0;
12     Pi = max value of these logged positions for Ci;
       /*till now, each cluster Ci has a position Pi */
13     if any Pi ≠0
14         currentCluster=Ci;
15     else
16         currentCluster=Ci whose Pi is max {P1, P2,...,Pk};
17     for each ri whose Item(k)i is in currentCluster Ci
18         remove Item(k)i from currentItemSet;
19         if Item(k+1)i exists in ri
20             put Item(k+1)i into currentItemSet;
21     for each ri that has no item in currentCluster Ci
22         insert a blank item ahead of Item(k)i in ri ;
24     Ui(k)++;
End
```

4. DATA ANALYSIS

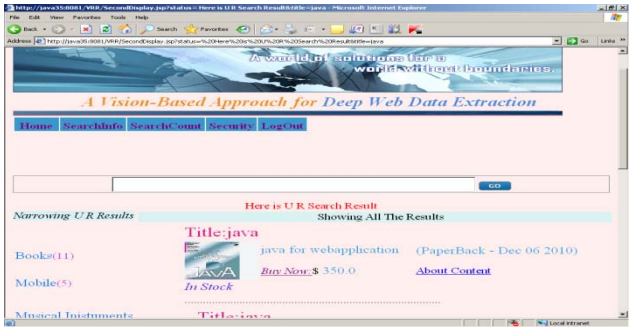
Collection of ip's:

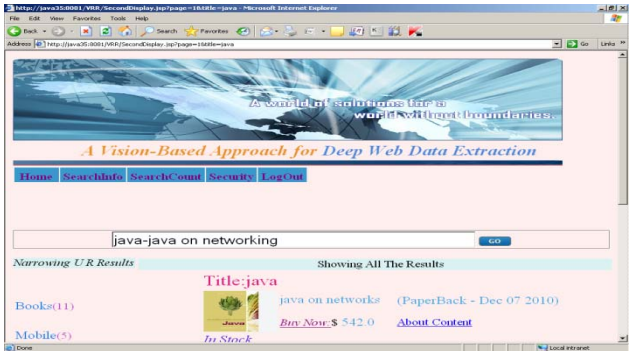
Login user display window:

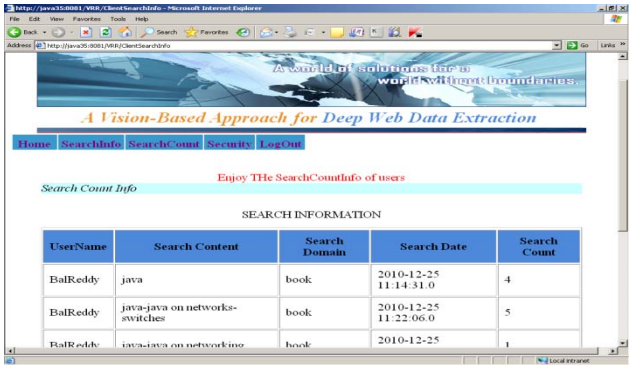
Search window:



Search result window:







5. CONCLUSION

With the flourish of the deep Web, users have a great opportunity to benefit from such abundant information in it. In general, the desired information is embedded in the deep Web pages in the form of data records returned by Web databases when they respond to users' queries. It focuses on the structured Web data extraction problem, including data record extraction and data item extraction. Based on observations of a large number of deep Web pages, this approach identifies a set of interesting common visual features that are useful for deep Web data extraction. Based on these visual features, a novel vision-based approach (ViDE) is proposed to extract structured data from deep Web pages.

This approach consists of four primary steps "Visual Block tree building, data record extraction, data item extraction, and visual wrapper generation". Visual Block tree building is to build the Visual Block tree for a given sample deep page using the VIPS algorithm. With the Visual Block tree, data record extraction and data item extraction are carried out based on the proposed visual features.. Visual wrapper generation is to generate the wrappers that can improve the efficiency of both data record extraction and data item extraction. Highly accurate experimental results provide strong evidence that rich visual features on deep Web pages can be used as the basis to design highly effective data extraction algorithms.

However, there are still some remaining issues and we plan to address them in the future. First, ViDE can only process deep Web pages containing one data region, while there is significant number of multidata-region deep Web pages. Though Zhao et al. [31] have attempted to address this problem, their solution is HTML-dependent and its performance has a large room for improvement. We intend to propose a vision-based approach to tackle this problem. Second, the efficiency of ViDE can be improved. In the current ViDE, the visual information of Web pages is obtained by calling the programming APIs of IE, which is a time-consuming process. To address this problem, we intend to develop a set of new APIs to obtain the visual information directly from the Web pages.

REFERENCES

[1] .G.O. Arocena and A.O. Mendelzon, "WebOQL: Restructuring Documents, Databases, and Webs," Proc. Int'l Conf. Data Eng. (ICDE), pp. 24-33, 1998.

[2] D. Buttler, L. Liu, and C. Pu, "A Fully Automated Object Extraction System for the World Wide Web," Proc. Int'l Conf. Distributed Computing Systems (ICDCS), pp. 361-370, 2001.

[3] D. Cai, X. He, J.-R. Wen, and W.-Y. Ma, "Block-Level Link Analysis," Proc. SIGIR, pp.

[4] D. Cai, S. Yu, J. Wen, and W. Ma, "Extracting Content Structure for Web Pages Based on Visual Representation," Proc. Asia Pacific Web Conf. (APWeb), pp. 406-417, 2003.

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

[6] C.-H. Chang, C.-N. Hsu, and S.-C. Lui, "Automatic Information Extraction from Semi-Structured Web Pages by Pattern Discovery," Decision Support Systems, vol. 35, no. 1, pp. 129-147, 2003.

[7] V. Crescenzi and G. Mecca, "Grammars Have Exceptions," Information Systems, vol. 23, no. 8, pp. 539-565, 1998.

[8] V. Crescenzi, G. Mecca, and P. Merialdo, "RoadRunner: Towards Automatic Data Extraction from Large Web Sites," Proc. Int'l Conf. Very Large Data Bases (VLDB), pp. 109-118, 2001.

[9] D.W. Embley, Y.S. Jiang, and Y.-K. Ng, "Record-Boundary Discovery in Web Documents," Proc. ACM SIGMOD, pp. 467- 478, 1999.

[10] W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krpl, and B. Pollak, "Towards Domain Independent Information Extraction from Web Tables," Proc. Int'l World Wide Web Conf. (WWW), pp. 71-80, 2007.

[11] J. Hammer, J. McHugh, and H. Garcia-Molina, "Semistructured Data: The TSIMMIS Experience," Proc. East-European Workshop Advances in Databases and Information Systems (ADBIS), pp. 1-8, 1997.