ISSN:0975-9646

Visual Model for Structured data Extraction Using Position Details

B.Venkat Ramana[#1] A.Damodaram[*2]

[#1] *Department of CSE, MIPGS, Hyderabad-59*
[*2] *Department of CSE, JNTUH, Hyderabad*

Abstract-- **The Web contains more online data which can be searched through their web query interface. Extracting structured data from deep Web pages is a challenging task due to the underlying complicate structures of such pages. Many techniques have been proposed to address this problem of extraction, but all of them have some limitations due to their dependency on the Web-page-programming-language. 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 paper, a new vision-based approach that is Web-page programming- language-independent is proposed. This new method finds the matching pattern between the attribute values of the two sites and ignores unwanted portions of the attribute. Automatic pattern discovery along with position details method with tree matching is used as structured data extraction method. The main advantage of the method is that it requires less human intervention. 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. Experimental results show that almost all extraction targets can be successfully extracted by the developed extractor.**

Keywords: **Web Information Extraction, Data Mining, Pattern Recognition, web content, record Extraction**

I. INTRODUCTION

The World Wide Web has more and more online Web databases which can be searched through their Web query interfaces. The number of Web databases has reached 25 millions according to a recent survey .All the Web databases make up the deep Web (hidden Web 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 crawler based search engines, such as Google and Yahoo. In this paper, we call this kind of special Web pages as deep Web pages. Each data record on the deep Web pages corresponds to an object. However, processing the data records and data items in the machine that is necessary in applications such as deep Web crawling and metasearching, the structured data need to be extracted from the deep Web pages. Here the structured data, including data records and data items can be extracted automatically from the deep Web pages.

Data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information - information that can be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases. Data are any facts, numbers, or text that can be processed by a computer. Today, organizations are accumulating vast and growing amounts of data in different formats and different databases. For example, analysis of retail point of sale transaction data can yield information on which products are selling and when. Information can be converted into knowledge about historical patterns and future trends. For example, summary information on retail supermarket sales can be analyzed in light of promotional efforts to provide knowledge of consumer buying behavior. Thus, a manufacturer or retailer could determine which items are most susceptible to promotional efforts.

In web mining web can be viewed as the largest database available and presents a challenging task for effective design and access. Thus, data mining applied to the Web has the potential to be quite beneficial. Web mining is mining of data related to the World Wide Web. This may be the data actually present in Web Pages or data related to Web activity. Web data can be classified into the following classes:

- Content of actual Web pages
- Intrapage structure includes the HTML or XML code for the page.
- Interpage structure is the actual linkage structure between Web pages.
- Usage data that describe how Web pages are accessed by visitors.

Web mining tasks can be divided into several classes. Web content mining examines the content of Web pages as well as results of Web searching. The content includes text as well as graphics data. Web content mining is further divided into Web page content mining and search results mining. The Web page content mining is traditional searching of Web pages via content, while the search results mining is a further search of pages found from a previous search. Thus, some mining activities have been built on top of traditional search engines, using their result as the data to be mined. With Web structure mining, information is obtained from the actual organization of pages on the Web. Content mining is similar to the work performed by basic IR techniques, but it usually goes farther than simply employing keyword searching. For example clustering may be applied to Web pages to identify similar pages. The intra page structure includes links within the page as well as the code (HTML, XML) for the page. Web usage mining looks at logs of Web access. General access pattern tracking is a type of usage mining that locks at a history of Web pages visited. This usage may be general or may be targeted to specific usage or users. For Example, patterns can be clustered based on their similarity. This in turn can be used to cluster users into groups based on similar access behavior. Web usage mining is a process of extracting useful information from server logs i.e. user's history. Web usage mining is the process of finding out what users are looking for on the Internet. Some users might be looking at only textual data, whereas some others might be interested in multimedia data. Web content mining is the process to discover useful information from text, image, audio or video data in the web. Web content mining sometimes is called web text mining, because the text content is the most widely researched area. Web structure mining is the process of using graph theory to analyze the node and connection structure of a web site. According to the type of web structural data, web structure mining can be divided into two kinds:

- Extracting patterns from hyperlinks in the web: a hyperlink is a structural component that connects the web page to a different location.
- Mining the document structure: analysis of the tree-like structure of page structures to describe HTML or XML tag usage.

The deep Web refers to World Wide Web content that is not part of the surface Web, which is indexed by standard search engines. In analogy to search engines over the "crawlable" web, we argue that one way to unlock the Deep Web is to employ a fully automated approach to extracting, indexing, and searching the query-related information-rich regions from dynamic web pages. Extracting the interesting information from a Deep Web site requires many things: including scalable and robust methods for analyzing dynamic web pages of a given web site, discovering and locating the query-related information-rich content regions, and extracting itemized objects within each region. By full automation, we mean that the extraction algorithms should be designed independently of the presentation features or specific content of the web pages, such as the specific ways in which the query-related information is laid out or the specific locations where the navigational links and advertisement information are placed in the web pages. The Deep Web comprises all information that resides in autonomous databases behind portals and information providers' web front-ends. Web pages in the Deep Web are dynamically-generated in response to a query through a web site's search form and often contain rich content.

Data Extraction from Web Extractors Web even those web sites with some static links that are "crawlable" by a search engine often have much more information available only through a query interface. Unlocking this vast deep web content presents a major research challenge. A Web crawler is a computer program that browses the World Wide Web in a methodical, automated manner or in an orderly fashion. Other terms for Web crawlers are ants, automatic indexers, bots, Web spiders, Web robots, or—especially in the FOAF community—Web scutters. This process is called Web crawling or spidering. The behavior of a Web crawler is the outcome of a combination of policies:

- a selection policy that states which pages to download,
- a re-visit policy that states when to check for changes to the pages,
- a politeness policy that states how to avoid overloading Web sites, and
- a parallelization policy that states how to coordinate distributed Web crawlers.

Objective of this work is to 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). By using visual features for data extraction, ViDE avoids the limitations of those solutions that need to analyze complex Web page source files. In Section II describes the Data Records Extraction, Section III contains the Data item Extraction Section IV contains Visual Wrapper Generation Section V describes about conclusion Section VI contains Future Works.

II. Data Records Extraction

Data record extraction aims to discover the boundary of data records and extract them from the deep Web pages. An ideal record extractor should achieve the following: 1) all data records in the data region are extracted and 2) for each extracted data record, no data item is missed and no incorrect data item is included. Instead of extracting data records from the deep Web page directly, we first locate the data region, and then, extract data records from the data region. PF1 and PF2 indicate that the data records are the primary content on the deep Web pages and the data region is centrally located on these pages. The data region corresponds to a block in the Visual Block tree. We locate the data region by finding the block that satisfies the two position features. Each feature can be considered as a rule or a requirement. Though very simple, this method can find the data region in the Visual Block tree accurately and efficiently.

Each data record corresponds to one or more sub trees in the Visual Block tree, which are just the child blocks of the data region. So, we only need to focus on the child blocks of the data region. In order to extract data records from the data region accurately, two facts must be considered. First, there may be blocks that do not belong to any data record, such as the statistical information (e.g., about 2,038 matching results for java) and annotation about data records (e.g., 1, 2, 3, 4, 5 (Next)). These blocks are called noise blocks in this paper. Noise blocks may appear in the data region because they are often close to the data records. According to LF2, noise blocks cannot appear between data records. They always appear at the top or the bottom of the data region. Second, one data record may correspond to one or more blocks in the Visual Block tree, and the total number of blocks in which one data record contains is not fixed.

Data record extraction is to discover the boundary of data records based on the LF and AF features. We achieve this in the following three phases:

Phase 1: Noise Blocks Filtering

Because noise blocks are always at the top or bottom, we check the blocks located at the two positions according to LF1. If a block at these positions is not aligned flush left, it will be removed as a noise block. This step does not guarantee the removal of all noise blocks.

Phase 2: Blocks Clustering

The remaining blocks in the data region are clustered based on their appearance similarity. Since there may be three kinds of information in data records, i.e., images, plain text, and link text, the appearance similarity between blocks is computed from the three aspects. For images, we care about the size; for plain text and link text, we care about the shared fonts. Intuitively, if two blocks are more similar on image size and font, they should be more similar in appearance.

Phase 3: Blocks Regrouping

The clusters obtained in the previous step do not correspond to data records. On the contrary, the blocks in the same cluster all come from different data records. According to AF2, the blocks in the same cluster have the same type of contents of the data records. The blocks need to be regrouped such that the blocks belonging to the same data record form a group. Our basic idea of blocks regrouping is as follows: According to CF1, the first data item in each data record is always mandatory. Clearly, the cluster that contains the blocks for the first items has the maximum number of blocks possible; let n be this maximum number. It is easy to see that if a cluster contains n blocks, these blocks contain mandatory data items. Our regrouping method first selects a cluster with n blocks and uses these blocks as seeds to form data records. Next, given a block b, we determine which record b belongs to according to CF2.

a. Algorithm Block Regrouping

```
Input: C1, C2, Cm; a group of clusters generated by blocks clustering from a given sample deep web page P
Output: G1, G2… Gm; each of them corresponds to a data record on P
Begin
//Step 1: sort the blocks in Ci according to their positions in the page (from top to bottom and then from left to right)
1 for each cluster Ci do
2 for any two blocks $b_{i,j}$ and $b_{i,k}$ in Ci //$1 \leq j < k \leq |Ci|$
3 if $b_{i,j}$ and $b_{i,k}$ are in different lines on P,and bi,k is above $b_{i,j}$
4 $b_{i,j} \leftrightarrow b_{i,k}$; //exchange their orders in Ci;
5 else if $b_{i,j}$ and $b_{i,k}$ are in the same line on P, and $b_{i,k}$ is in front of $b_{i,j}$
6 $b_{i,j} \leftrightarrow b_{i,k}$;
7 end until no exchange occurs;
8 from the minimum-bounding rectangle Reci for Ci;
//Step 2: initialize n groups, and n is the number of data records on P
9 Cmax= {Ci| |Ci|=max {|C1|,|C2|,…,|Cm|}}; // n=|Cmax|
10 for each block $b_{max,j}$ in Cmax
11 Initialize group Gi;
12 Put $b_{max,I}$ into Gi;
//Step 3: put the blocks into the right groups, and each group corresponds to a data record
13 For each cluster Ci
14 if Reci overlaps with Recmax on P
15 if Reci is ahead of (behind) Recmax
16 for each blocks $b_{i,j}$ in Ci
17 find the nearest block bmax,k in Cmax that is behind (ahead of) $b_{i,j}$ on the web page; place $b_{i,j}$ into group Gi; End
```

III. DATA ITEM EXTRACTION

A data record can be regarded as the description of its corresponding object, which consists of a group of data items and some static template texts. In real applications, these extracted structured data records are stored (often in relational tables) at data item level and the data items of the same semantic must be placed under the same column. When introducing CF, we mentioned that there are three types of data items in data records: mandatory data items, optional data items, and static data items. We extract all three types of data items. Note that static data items are often annotations to data and are useful for future applications, such as Web data annotation. Below, we focus on the problems of segmenting the data records into a sequence of data items and aligning the data items of the same semantics together. Note that data item extraction is different from data record extraction; the former focuses on the leaf nodes.

3.1 Data Record Segmentation

AF3 indicates that composite data items cannot be segmented any more in the Visual Block tree. So, given a data record, we can collect its leaf nodes in the Visual Block tree in left to right order to carry out data record segmentation. Each composite data item also corresponds to a leaf node. We can treat it as a regular data item initially, and then, segment it into the real data items with the heuristic rules mentioned in AF3 after the initial data item alignment. The Visual Block tree, while the latter focuses on the child blocks of the data region in the Visual Block tree.

3.2 Data Item Alignment

CF1 indicates that we cannot align data items directly due to the existence of optional data items. It is natural for data records to miss some data items in some domains. For example, some books have discount price, while some do not. Data item alignment focuses on the problem of how to align the data items of the same semantic together and also keep the order of the data items in each data record. In the following, we first define visual matching of data items, and then, propose an algorithm for data item alignment.

3.3 Visual Matching of Data Items

AF2 indicates that if two data items from different data records belong to the same semantic, they must have consistent font and position, including both absolute position and relative position. The first four lines of the algorithm say that two data items are matched only if they have the same absolute position in addition to having the same font. Here, absolute position is the distance between the left side of the data region and the left side of a data item. When two data items do not have the same absolute position, they can still be matched if they have the same relative position. For match on relative position, the data items immediately before the two input data items should be matched.

3.3.1 Algorithm-Data Item Matching

```
Input: item1, item2: two data items
Output: matched or unmatched: the match result (Boolean)
Begin
1 if (font (item1) ≠ font (item2))
2 Return unmatched;
3 if (position (item1) =position (item2))
4 Return matched;
5 if (itemp1 and itemp2 are matched) //itemp1 and itemp2 are the data items immediately in front of item1 and item2 respectively
6 Return matched;
7 else
8 Return unmatched;
9 End
```

3.4 Data Item Alignment

CF2 says that the order of data items in data records is fixed. Thus, each data record can be treated as a sequence of data items. We can utilize this feature to align data items. Our goal is to place the data items of the same semantic in the same column. If an optional data item does not appear in a data record, we will fill the vacant position with a predefined blank item. Based on this insight, we propose a multi alignment algorithm that can process all extracted data records holistically step by step. The basic idea of this algorithm is described as follows: Initially, all the data items are unaligned. We align data items by the order in their corresponding data records. When we encounter optional data items that do not appear in some data records, these vacant positions will be filled with the predefined blank item. This ensures that all data records are aligned and have the same number of data items at the end.

3.4.1 Algorithm-Data Item Alignment

```
Input: a set of extracted data records {n|1≤j≤n}
Output: a set of data records {n|1≤j≤n} with all the data items aligned
Begin
1. Currentitemset=φ;
2. Currentitemcluster=φ;
3. //put the first unaligned data item of each n into currentItemSet:
//Itemiu(i) refers to the first unaligned item of th ith data record
Currentitemset↔itemiu(i)(1≤i≤n) ;
4. While currentItemSet≠φ
5. use the data item matching algorithm to group the data items
In the currentItemSet into k clusters {$c_i$|1≤i≤k} (k≤n);
6. for each cluster ci for each r1 that does not have a data item in ci
7. if Itemju(j)+kis matched with data items in ci
8. Log position k;
9. else
10. Log position 0;
11. Pi=max value of these logged position for Ci;
12./*Till now, each cluster Ci has a positionPi*/ if any PL==0
currentCluster=CL
13.Else
14 current cluster=cl whose pl is max{p1,p2,….pk};
  for each $r_j$ whose item I U(j) is in current cluster cl
15 Remove itemiU(j) from current item set;
16 If itemiU(j)-1 exists in rj
17 Put itemiU(i)+1 into current item set;
18 For each rj that has no item in current cluster cl
19 Insert a blank item ahead of itemiU(j) in rj;
20 U(j)++;
21 end
```

IV. Visual Wrapper Generation

ViDE has two components: ViDRE and ViDIE. There are two problems with them. First, the complex extraction processes are too slow in supporting real-time applications. Second, the extraction processes would fail if there is only one data record on the page. Since all deep Web pages from the same Web database share the same visual template, once the data records and data items on a deep Web page have been extracted, we can use these extracted data records and data items to generate the extraction wrapper for the Web database so that new deep Web pages from the same Web database can be processed using the wrappers quickly without reapplying the entire extraction process. Our wrappers include data record wrapper and data item wrapper. They are the programs that do data record extraction and data item extraction with a set of parameter obtained from sample pages. For each Web database, we use a normal deep Web page containing the maximum number of data records to generate the wrappers. The wrappers of previous works mainly depend on the structures or the locations of the data records and data items in the tag tree, such as tag path. In contrast, we mainly use the visual information to generate our wrappers. Note that some other kinds of information are also utilized to enhance the performances of the wrappers, such as the data types of the data items and the frequent symbols appearing in the data items. But they are easy to obtain from the Web pages. We describe the basic ideas of our wrappers below.

4.1 Vision-Based Data Record Wrapper

Given a deep Web page, vision-based data record wrapper first locates the data region in the Visual Block tree, and then, extracts the data records from the child blocks of the data region. After the data region R on a sample deep Web page P from site S is located by ViDRE, we save five parameters values (x; y; w; h; l), where (x; y) form the coordinate of R on P, w and h are the width and height of R, and l is the level of R in the Visual Block tree. Given a new deep Web page P_ from S, we first check the blocks at level l in the Visual Block tree for P_. The data region on P_ should be the block with the largest area overlap with R on P_. The overlap area can be computed using the coordinates and width/height information. Data record extraction. For each record, our visual data record wrapper aims to find the first block of each record and the last block of the last data record (denoted as blast). To achieve this goal, we save the visual information (the same as the information used in (1)) of the first block of each data record extracted from the sample page and the distance (denoted as d) between two data records. For the child blocks of the data region in a new page, we find the first block of each data record by the visual similarity with the saved visual information. Next, blast on the new page needs to be located. Based on our observation, in order to help the users differentiate data records easily, the vertical distance between any two neighboring blocks in one data record is always smaller than d and the vertical distance between blast and its next block is not smaller than d. Therefore, we recognize the first block whose distance with its next block is larger than d as blast.

4.2 Vision-Based Data Item Wrapper

The data alignment algorithm groups data items from different data records into columns or attributes such that data items under the same column have the same semantic. Table 6 lists useful information about each attribute obtained from the sample page that can help determine which attribute a data item belongs to. The basic idea of our vision-based data item wrapper is described as follows: Given a sequence of attributes fa1; a2; . . . ; a ng obtained from the sample page and a sequence of data items item1; item2; . . . ; item mg obtained from a new data record, the wrapper processes the data items in order to decide which attribute the current data item can be matched to. For item i and aj, if they are the same on f, l, and d, their match is recognized. The wrapper then judges whether itemiþ1 and ajþ1 are matched next, and if not, it judges i temi and ajþ1. Repeat this process until all data items are matched to their right attributes. Note that if an attribute on a new page did not appear on the sample page, the data item of the attribute cannot be matched to any attribute. To avoid such a problem, several sample pages may be used to generate the wrapper. This can increase the chance that every attribute appears on at least one of these sample pages.

V. 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. Therefore, it is an important task to extract the structured data from the deep Web pages. Here we focused on the structured Web data extraction problem, including data record extraction and data item extraction. First, surveyed previous works on Web data extraction and investigated their inherent limitations. The visual information of Web pages can help us implement Web data extraction. Based on our observations of a large number of deep Web pages, it is identified that a set of interesting common visual features that are useful for deep Web data extraction. Based on these visual features, we proposed a novel vision-based approach to extract structured data from deep Web pages, which can avoid the limitations of previous works. The main trait of this vision-based approach is that it primarily utilizes the visual features of deep Web pages.

VI. Future Work

There are still some remaining issues and we plan to address them in future. ViDE can only process deep web pages containing one data region, while there is a significant number of multidata-region deep web pages. We intend to propose vision based approach to tackle HTML dependent and its performance. The efficiency of ViDE can be improved. In the current ViDE, the visual information of web pages is obtained by calling the programming AIPs of IE, which is time consuming process. To address this problem, we intend to develop a a set of new AIPs to obtain the visual information directly from web pages.This process of extracting Deep Web Pages can be further improved by using the algorithm “EXTENDED COCITATION ALGORITHM”.

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. 440-447, 2004.

[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]. S. Oswalt Manoj, Nisha Soms and N.V. Shibu -Visual Architecture based Web Information Extraction, International Journal of Data Mining, Vol. 1, Special Issue, December 2011.