Vision-based Web Data Records Extraction

Wei Liu, Xiaofeng Meng
School of Information
Renmin University of China
Beijing, 100872, China
{gue2, xfmeng}@ruc.edu.cn

Weiyi Meng
Dept. of Computer Science
SUNY at Binghamton
Binghamton, NY 13902
meng@cs.binghamton.edu

ABSTRACT

This paper studies the problem of extracting data records on the response pages returned from web databases or search engines. Existing solutions to this problem are based primarily on analyzing the HTML DOM trees and tags of the response pages. While these solutions can achieve good results, they are too heavily dependent on the specifics of HTML and they may have to be changed should the response pages are written in a totally different markup language. In this paper, we propose a novel and language independent technique to solve the data extraction problem. Our proposed solution performs the extraction using only the visual information of the response pages when they are rendered on web browsers. We analyze several types of visual features in this paper. We also propose a new measure ***revision*** to evaluate the extraction performance. This measure reflects perfect extraction ratio among all response pages. Our experimental results indicate that this vision-based approach can achieve very high extraction accuracy.

Keywords

Web DB, response page, data record

1. INTRODUCTION

The World Wide Web has close to one million searchable information sources according to a recent survey[1]. These searchable information sources include both search engines and Web databases. By posting queries to the search interfaces of these information sources, useful information from them can be retrieved. Often the retrieved information (query results) is wrapped on response pages returned by these systems in the form of data records, each of which corresponds to an entity such as a document or a book. Data records are usually displayed visually neatly on Web browsers to ease the consumption of human users. In Figure 1, a number of book records are listed on a response page from Amazon.com.

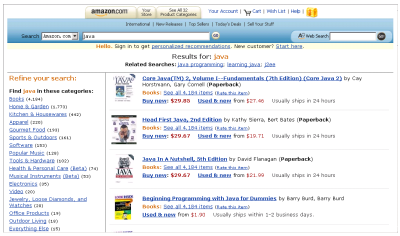

Figure 1: A response page from Amazon

However, to make the retrieved data records machine processable, which is needed in many applications such as deep web crawling and metasearching, they need to be extracted from the response pages. In this paper, we study the problem of automatically extracting the data records from the response pages of web-based search systems.

The problem of web data extraction has received a lot of attention in recent years[2][5][6][7][8]. The existing solutions are mainly based on analyzing the HTML source files of the response pages. Although they can achieve reasonably high accuracies in the reported experimental results, the current studies of this problem have several limitations. First, HTML-based approaches suffer from the following problems: (1) HTML itself is still evolving and when new versions or new tags appear, the previous solutions will have to be amended repeatedly to adapt to new specifications and new tags. (2) Most previous solutions only considered the HTML files that do not include scripts such as JavaScript and CSS. As more and more web pages use more complex JavaScript and CSS to influence the structure of web pages, the applicability of the existing solutions will become lower. (3) If HTML is replaced by a new language in the future, then previous solutions will have to be revised greatly or even abandoned, and other approaches must be proposed to accommodate the new language. Second, traditional performance measures, precision and recall, do not fully reflect the quality of the extraction. Third, most performance studies used small data sets, which is inadequate in assuring the impartiality of the experimental results.

There are already some works [9][12] that analyze the layout structure of web pages. They try to effectively represent and understand the presentation structure of web pages, which are physical structure independent. But the research on vision-based web data extraction is still at its infancy. It is well known that web pages are used to publish information for humans to browse, and not designed for computers to extract information automatically. Based on such consideration, in this paper we propose a novel approach to

extract data records automatically based on the visual representation of web pages. Like [8][7], our approach also aims at the response pages that have multiple data records. Our approach employs a three-step strategy to achieve this objective. First, given a response page, transform it into a Visual Block tree based on its visual representation; second, discover the region (data region) which contains all the data records in the Visual Block tree; third, extract data records from the data region.

This paper has the following contributions:

1. We believe this is the first work that utilizes only the visual content features on the response page as displayed on a browser to extract data records automatically.

2. A new performance measure, ***revision***, is proposed to evaluate the approaches for web data extraction. The measure ***revision*** is the percentage of the web sites whose records cannot be perfectly extracted (i.e., at least one of the precision and recall is not 100%). For these sites, manual revision of the extraction rules is needed.

3. A data set of 1,000 web databases and search engines is used in our experiment study. This is by far the largest data set used in similar studies (previous works seldom used 200 sites). Our experimental results indicate that our approach is very effective.

2. RELATED WORKS

Until now, many approaches have been reported in the literature for extracting information from Web pages. Recently, many automatic approaches [5][6][7][8] have been proposed instead of manual approaches [2] and semi-automatic approaches [3] [4]. For example, [6] find patterns or grammars from multiple pages in HTML DOM trees containing similar data records, and they require an initial set of pages containing similar data records. In [5], a string matching method is proposed, which is based on the observation that all the data records are placed in a specific region and this is reflected in the tag tree by the fact that they share the same path in DOM tree. The method DEPTA[7] used tree alignment instead of tag strings, which exploits nested tree structures to perform more accurate data extraction, so it can be considered as an improvement of MDR[8]. The only works that we are aware of that utilize some visual information to extract data records are [13][14]. However, in these approaches, tag structures are still the primary information utilized while visual information plays a small role. For example, in [13], when the visual information is not used, the recall and precision decrease by only 5%. In contrast, in this paper, our approach performs data record extraction completely based on visual information.

Although the works discussed above applied different techniques and theories, they have a common characteristic: they are all implemented based on HTML DOM trees and tags by parsing the HTML documents. In Section 1, we discussed the latent and inevitable limitations of them.

Since web pages are used to publish information for humans to browse and read, the desired information we want extracted must be visible, so the visual features of web pages can be very helpful for web information extraction. Currently, some works are proposed to process web pages based on their visual representation. For example, a web page segmentation algorithm VIPs is proposed in [9] which simulates how a user understands web layout structure based on his/her visual perception. Our approach is implemented based on VIPs. [10] is proposed to implement link analysis based on the layout and visual information of web pages. Until now, the layout and visual information is not effectively utilized to extract structural web information, and it is only considered as a heuristic accessorial means.

3. INTERESTING VISUAL OBSERVATIONS FOR RESPONSE PAGES

Web pages are used to publish information on the Web. To make the information on web pages easier to understand, web page designers often associate different types of information with distinct visual characteristics (such as font, color, layout, etc.). As a result, visual features are important for identifying special information on Web pages.

Response pages are special web pages that contain data records retrieved from Web information sources, and the data records contained in them also have some interesting distinct visual features according to our observation. Below we describe the main visual features our approach uses.

Position Features (PF): These features indicate the location of the data region on a response 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.

Though web pages are designed by different people, these designers all have the common consideration in placing the data region: the data records are the contents in focus on response pages, and they are always centered and conspicuous on web pages to catch the user's attention. By investigating a large number of response pages, we found two interesting facts. First, data regions are always located in the middle section horizontally on response 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 for different systems because it is not only influenced by the number of data records retrieved but also by what information is included in each data record, which is application dependent. Therefore, our approach does not use the actual size, instead it uses the ratio of the size of the data region to the size of whole response page.

Layout Features (LF): 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.

The designers of web pages always arrange the data records in some format in order to make them visually regular. The regularity can be presented by one of the two layout models.

In Model 1, The data records are arrayed 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 multiple columns, and the data records in the same column have the same distance to the left boundary of the data region. In addition, data records do not overlap, which means that the regions of different data records can be separated. Based on our observation, the response pages of all search engines follow Model 1 while the response pages of web databases may follow either

Table 1: Relevant visual information about the four data records in Fig. 1

	Images (pixel)	plain texts		link texts	
		Total font number	Shared font number	Total font number	Shared font number
Data record 1	944	5	4	3	3
Data record 2	1056	5	4	3	3
Data record 3	871	5	4	3	3
Data record 4	912	4	4	3	3

of the two models. Model 2 is a little bit more complicated than Model 1 in layout, and it can be processed with some extension to the techniques used to process Model 1. In this paper, we focus on dealing with Model 1 due to the limitation of paper length.

We should note that feature LF1 is not always true as some data records on certain response pages of some sites (noticeably Google) may be indented. But the indented data records and the un-indented ones have very similar visual features. In this case, all data records that satisfy Model 1 are identified first, and then the indented data records are extracted utilizing the knowledge obtained from un-indented data records that have already been identified.

Appearance Features (AF): 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:* Data contents of the same type in different data records have similar presentations in three aspects: size of image, font of plain text and font of link text (The font of text is determined by font-size, font-color, font-weight and font-style).

Data records usually contain three types of data contents, i.e., images, plain texts (the texts without hyperlinks) and link texts (the texts with hyperlinks). Table 1 shows the information on the three aspects of data records in Figure 1, and we can find that the four data records are very close on the three aspects.

Our data record extraction solution is developed mainly based on the above three types of visual features. Feature PF is used to locate the region containing all the data records on a response page; feature LF and feature AF are combined together to extract the data records accurately.

Content Feature (CF): These features hint the regularity of the contents in data records.

- *CF1:* All data records have mandatory contents and some may have optional contents.
- *CF2:* The presentation of contents in a data record follows a fixed order.

The data records are the entities in real world, and they consist of data units with different semantic concepts. The data units can be classified into two kinds: mandatory and optional. Mandatory units are those that must appear in each data record. For example, if every book data record must have a title, then titles are mandatory data units. In contrast, optional units may be missing in some data records. For example, discounted price for products in e-commerce web sites is likely an optional unit because some products may not have discount price.

4. WEB DATA RECORD EXTRACTION

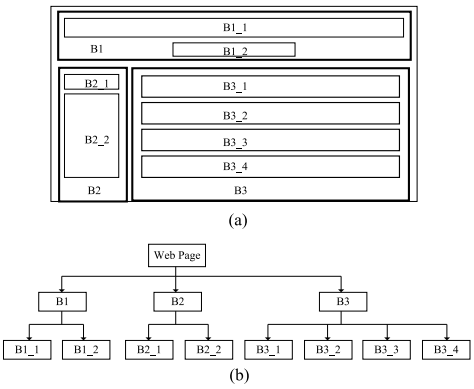

Figure 2: The content structure (a) and its Visual Block tree (b)

Based on the visual features introduced in the previous section, we propose a vision-based approach to extract data records from response pages. Our approach consists of three main steps. First, use the VIPs [9] algorithm to construct the Visual Block tree for each response page. Second, locate the data region in the Visual Block tree based on the PF features. Third, extract the data records from the data region based on the LF and AF features.

4.1 Building Visual Block tree

The Vision-based Page Segmentation (VIPs) algorithm aims to extract the content structure of a web page based on its visual presentation. Such content structure is a tree structure, and each node in the tree corresponds to a rectangular region on a web page. The leaf blocks are the blocks that cannot be segmented further, and they represent the minimum semantic units, such as continuous texts or images. There is a containment relationship between a parent node and a child node, i.e., the rectangle corresponding to a child node is contained in the rectangle corresponding to the parent node. We call this tree structure Visual Block tree in this paper. In our implementation we adopt the VIPS algorithm to build a Visual Block tree for each response page. Figure 2(a) shows the content structure of the response page shown in Figure 1 and Figure 2(b) gives its corresponding Visual Block tree. Actually, Visual Block tree is more complicated than what Figure 2 shows (there are often hundreds even thousands of blocks in a Visual Block tree).

For each block in the Visual Block tree, its position (the position on response page) and its size (width and height) are logged. The leaf blocks can be classified into three kinds: image block, plain text block and link text block, which represent three kinds of information in data records respectively. If a leaf block is a plain text block or a link text block, the font information is attached to it.

4.2 Data region discovery

PF1 and PF2 indicate that the data records are the primary content on the response pages and the data region is centrally located on these pages. The data region corresponds to a block in the Visual Block tree (in this paper we only consider response pages that have only a single data region). We locate the data region by finding the block that satisfies the two PF features. Each feature can be considered a rule or a requirement. The first rule can be applied directly, while the second rule can be represented

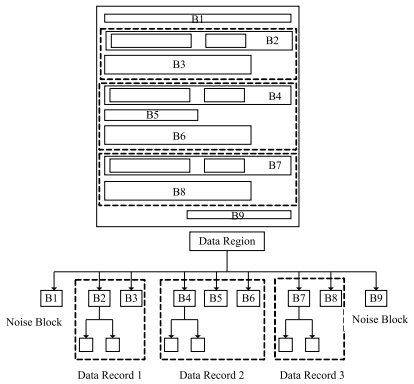

Figure 3: A general case of data region

by $(area_b/area_{responsepage}) \geq T_{dataregion}$, where $area_b$ is the area of block b, $area_{responsepage}$ is the area of the response page, and $T_{dataregion}$ is the threshold used to judge whether b is sufficiently large relative to $area_{responsepage}$. The threshold is trained from sample response pages collected from different real web sites. For the blocks that satisfy both rules, we select the block at the lowest level in the Visual Block tree.

4.3 Data records extraction from 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 (about 2,038 matching results for java) and annotation about data records (1 2 3 4 5 [Next]). These blocks are called noise blocks in this paper. According to LF2, noise blocks cannot appear between data records and they can only 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 one data record contains is not fixed. For example, in Figure 1, "Buy new" price exists in all four data records, while "Used & new" price only exists in the first three data records. Figure 3 shows an example of a data region that has the above problems: Block B1 (statistical information) and B9 (annotation) are noise blocks; there are three data records (B2 and B3 form data record 1; B4, B5 and B6 form data record 2; B7 and B8 form data record 3), and the dashed boxes are the boundaries of data records.

This step is to discover the boundary of data records based on the LF and AF features. That is, we attempt to determine which blocks belong to the same data record. We achieve this with the following three sub-steps: *Sub-step1:* Filter out some noise blocks; *Sub-step2:* Cluster the remaining blocks by computing their appearance similarity; *Sub-step3:* Discover data record boundary by regrouping blocks.

4.3.1 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 is not aligned flush left, it will be removed from the data region as a noise block. In this sub-step, we cannot assure all noise blocks are removed. For example, in Figure 3, block B9 can be removed in this sub-step, while block B1 cannot be removed.

4.3.2 Blocks clustering

The remaining blocks in the data region are clustered based on their appearance similarity. Since there are three kinds of information in data records, i.e., images, plain text and link text, the appearance similarity of 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, font, they should be more similar in appearance. The appearance similarity formula between two blocks B1 and B2 is given below:

$$sim(B_1, B_2) = w_i \times simIMG(B_1, B_2)$$
$$+w_{pt} \times simPT(B_1, B_2) + w_{lt} \times simLT(B_1, B_2)$$

where simIMG(B_1, B_2) is the similarity based on image size, simPT(B_1, B_2) is the similarity on plain text font, and simLT(B_1, B_2) is the similarity on link text font. And w_i, w_{pt} and w_{lt} are the weights of these similarities, respectively. Table 2 gives the formulas to compute the component similarities and the weights in different cases. The weight of one type of contents is proportional to their total size relative to the total size of the two blocks.

A simple one-pass clustering algorithm is applied. The basic idea of this algorithm is as follows. First, starting from an arbitrary order of all the input blocks, take the first block from the list and use it to form a cluster. Next, for each of the remaining blocks, say B, compute its similarity with each existing cluster. Let C be the cluster that has the maximum similarity with A. If sim(B, C)$> T_{as}$ for some threshold T_{as}, which is to be trained by sample pages (generally, T_{as} is set to 0.8), then add B to C; otherwise, form a new cluster based on B. Function sim(B, C) is defined to be the average of the similarities between B and all blocks in C computed using the Formula above.

As an example, by applying this method to the blocks in Figure 1, the blocks containing the titles of the data records are clustered together after clustering, so are the prices of data records and other contents.

4.3.3 Blocks regrouping

In 4.3.2, the blocks in the data region are grouped into several clusters. However, these clusters do not correspond to data records. On the contrary, the blocks in the same cluster likely 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 in the data region are regrouped, and the blocks belonging to the same data record form a group. This regrouping process has the following three phases:

Phase 1. For each cluster C_i, obtain its minimum-bounding box R_i, which is the smallest rectangle on the response page that can enclose all the blocks in C_i. We get the same number of boxes as the clusters. Reorder the blocks in C_i from top to bottom according to their positions in web browser. Thus, $B_{i,j}$ is above $B_{i,j+1}$ on web browser.

Phase 2. Suppose C_{max} is the cluster with the maximum number of blocks. If there are multiple such clusters, select the one whose box is positioned higher than the others on the web browser (here "higher position" is based on the highest point in each block). Let the number of blocks in C_{max} be n. Each block in C_{max} forms an initial group. So there are n initial groups (G_1, G_2, , G_n) with each group G_k having only one block $B_{max,k}$.

Phase 3. For each cluster C_i, if R_i overlaps with R_{max} on

Table 2: Formulas and remarks

	Formula	Remarks
1	$\text{simIMG(B1, B2)}=\frac{\text{Min}\{sa_i(B1), sa_i(B2)\}}{\text{Max}\{sa_i(B1), sa_i(B2)\}}$	$sa_i(B)$ is the total area of images in block B. $sa_B(B)$ is the total area of block B. $fn_{pt}(B)$ is the total number of the fonts of plain texts in block B. $sa_{pt}(B)$ is the total area of plain texts in block B. $fn_{lt}(B)$ is the total number of the fonts of link texts in block B. sa_{pt}(B) is the total area of link texts in block B.
2	$w_i=\frac{sa_i(B1)+sa_i(B2)}{sa_B(B1)+sa_B(B2)}$	
3	$\text{simPT(B1,B2)}=\frac{\text{Min}\{fn_{pt}(B1), fn_{pt}(B2)\}}{\text{Max}\{fn_{pt}(B1), fn_{pt}(B2)\}}$	
4	$w_{pt}=\frac{sa_{pt}(B1)+sa_{pt}(B2)}{sa_B(B1)+sa_B(B2)}$	
5	$\text{simLT(B1,B2)}=\frac{\text{Min}\{fn_{lt}(B1), fn_{lt}(B2)\}}{\text{Max}\{fn_{lt}(B1), fn_{lt}(B2)\}}$	
6	$w_{lt}=\frac{sa_{lt}(B1)+sa_{lt}(B2)}{sa_B(B1)+sa_B(B2)}$	

the web browser, process all the blocks in C_i. If R_i is lower (higher) than R_{max}, then for each block $B_{i,j}$ in C_i, find the nearest block $B_{max,k}$ in C_{max} that is higher (lower) than $B_{i,j}$ and put $B_{i,j}$ into G_k. When all clusters are processed, each group is a data record.

The basic idea of the process is as follows. According to LF2 and LF3, no noise block can appear between data records, and its corresponding box will not overlap with others. So the boxes that overlap with others enclose all the blocks that belong to data records. In sub-step2 (section 4.3.2), the blocks containing the data contents of the same type will be in the same cluster (e.g., for book records, the blocks containing titles will be clustered together). According to CF1, if a cluster has the maximum number of blocks, then the blocks in this cluster are the mandatory contents in data records, and the number of blocks in it is the number of data records. If there is more than one such cluster, we select one as C_{max} (generally, the one whose box is higher than the others on the web browser is selected). We select the blocks in C_{max} as the seeds of the data records, and each block forms an initial group. In each initial group G_k, there is only one block $B_{max,k}$. Then we try to put the blocks in other clusters into the right groups. That means if a block $B_{i,j}$ (in C_i, C_i is not C_{max}) and a block $B_{max,k}$ (in C_{max}) are in the same data record, then $B_{i,j}$ should be put into the group $B_{max,k}$ belongs to. In another word, the blocks in the same data record are also in the same group. According to LF3, no two adjoining data records overlap. So for $B_{max,k}$ in C_{max}, the blocks that belong to the same data record with $B_{max,k}$ must be below $B_{max,k-1}$ and above $B_{max,k+1}$. For each C_i, if R_i is lower (higher) than R_{max}, then the block on top of R_i is lower (higher) than the block on top of R_{max}. According to CF2, this determines $B_{i,j}$ is lower (higher) than $B_{max,k}$ if they belong to the same data record. So we can conclude that, if $B_{max,k}$ is the nearest block higher (lower) than $B_{i,j}$, then $B_{i,j}$ is put into the group $B_{max,k}$ belongs to.

5. EXPERIMENTS

We have built an operational prototype system based on our method, and we evaluate it in this section. This prototype system is implemented with C# on a Pentium 4 2GH PC. For response pages with no more than 20 data records, the whole process takes no more than 3 seconds.

5.1 Data set

Most previous works on web data extraction conducted experimental evaluations on relatively small data sets, and as a result, the experimental results are often not very reliable. Sometimes, the same system/approach yields very different experimental results depending on the data sets used (e.g., see the experimental comparisons reported in [8][13] about three approaches). In general, there are two reasons that may lead to this situation: first, the size of the data set used is too small, and second, the data set used is not sufficiently representative of the general situation.

In this paper, we use a much larger data set than those used in other similar studies to avoid the problems mentioned above. Our data set is collected from the Completeplanet web site (www.completeplanet.com). Complete-planet is currently the largest depository for deep web, which has collected the search entries of more than 70,000 web databases and search engines. These search systems are organized under 43 topics covering all the main domains in real world. We select 1,000 web sites from these topics (the top 10 to 30 web sites in each topic). During our selection, duplicates under different topics are not used. In addition, web sites that are powered by well-known search engines such as Google are not used. This is to maximize the diversity among the selected web sites. For each web site selected, we get at least five response pages by submitting different queries to reduce randomness. Only response pages containing at least two data records are used. In summary, our data set is much larger and more diverse than any data set used in related works. We plan to make the data set publicly available in the near future.

5.2 Performance measures

Two measures, precision and recall, are widely used to measure the performance of data record extraction algorithms in published literatures. Precision is the percentage of correctly extracted records among all extracted records and recall is the percentage of correctly extracted records among all records that exist on response pages. In our experiments, a data record is correctly extracted only if anything in it is not missed and anything not in it is not included.

Besides precision and recall, there is an important measure neglected by other researchers. It is the number of web sites with perfect precision and recall, i.e., both precision and recall are 100% at the same time. This measure has a

Table3: Comparison of ViDRE and MDR

	ViDRE	MDR
DR_r	85,497	
DR_e	84,198	53,323
DR_c	83,103	45,485
Total Web sites	1,000	
Correct web sites	876	448
precision	98.7%	85.3%
recall	97.2%	53.2%
revision	12.4%	55.2%

great meaning for web data extraction in real applications. We give a simple example to explain this. Suppose there are three approaches (A1, A2 and A3) which can extract data records from response pages, and they use the same data set (5 web sites, 10 data records in each web site). A1 extracts 9 records for each site and they are all correct. So the average precision and recall of A1 are 100% and 90%, respectively. A2 extracts 11 records for each site and 10 are correct. So the average precision and recall of A2 are 90.9% and 100%, respectively. A3 extracts 10 records for 4 of the 5 sites and they are all correct. For the 5th site, A3 extracts no records. So the average precision and recall of A3 are both 80%. Based on average precision and recall, A1 and A2 are better than A3. But in real applications A3 may be the best choice. The reason is that in order to make precision and recall 100%, A1 and A2 have to be manually tuned/adjusted for each web site, while A3 only needs to be manually tuned for one web site. In other words, A3 needs the minimum manual intervention.

In this paper we propose a new measure called ***revision***. Its definition is given below.

$$revision = \frac{WS_t - WS_c}{WS_t}$$

where WS_c is the total number of web sites whose precision and recall are both 100%, and WS_t is total number of web sites processed. This measure represents the degree of manual intervention required.

5.3 Experimental results

We evaluate our prototype system ViDRE and compare it with MDR. We choose MDR based on two considerations: first, it can be downloaded from web site and can run locally; second, it is very similar to ViDRE (a single page at a time; data extracted at record level). MDR has a similarity threshold, which is set at default value (60%) in our test, based on the suggestion of the authors of MDR. Our ViDRE also has a similarity threshold, which is set at 0.8. We show the experimental results in Table 3.

From Table 3, we can draw two conclusions. First, the performance of ViDRE is very good. That means vision-based approach can also reach a high accuracy (precision and recall). Second, ViDRE is much better than MDR on revision. MDR has to be revised for nearly half of the web sites tested, while ViDRE only need to be revised for less than one eighth of these sites.

6. CONCLUSION AND FUTURE WORK

In this paper, we presented a fully automated technique to extract search result data records from response pages dynamically generated by search engines or Web DBs. Our technique utilizes only the visual content features on the response page, which is HTML language or any other language independent. This differentiates our technique from other competing techniques for similar applications. Our experimental results on a large data set indicate that our technique can achieve high extraction accuracy.

In the future, we plan to address several issues and improve our vision-based approach further. First, if there is only one data record on a response page, our approach will fail. We intend to tackle this problem by comparing multiple response pages from one web site. Second, data record extraction is slow when the number of data records is large (say more than 50). We plan to look into the issue of improving the efficiency of our approach. Third, we plan to collect a set of response pages from real web sites which are not designed with HTML, and show our vision-based approach is really language independent.

7. ACKNOWLEDGMENTS

This research was partially supported by the grants from the NSFC under grant number 60573091, 60273018, China National Basic Research and Development Program's Semantic Grid Project (No. 2003CB317000), the Key Project of Ministry of Education of China under Grant No.03044, Program for New Century Excellent Talents in University (NCET), and US NSF grants IIS-0414981 and CNS-0454298.

8. REFERENCES

[1] K. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured Databases on the Web: Observations and Implications. In SIGMOD Record, 33(3), pages 61-70, 2004.

[2] G. O. Arocena, A. O. Mendelzon. WebOQL: Restructuring Documents, Databases, and Webs. In ICDE, pages 24-33, 1998.

[3] X. Meng, H. Lu, H. Wang. SG-WRAP: A Schema-Guided Wrapper Generation. In ICDE, pages 331-332, 2002.

[4] R. Baumgartner, S. Flesca, G. Gottlob. Visual Web Information Extraction with Lixto. In VLDB , pages 119-128, 2001.

[5] C. Chang, S. Lui. IEPAD: Information extraction based on pattern discovery. In WWW, pages 681-688, 2001.

[6] V. Crescenzi, G. Mecca, P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In VLDB, pages 109-118, 2001.

[7] Y. Zhai, B. Liu. Web data extraction based on partial tree alignment. In WWW, pages 76-85, 2005.

[8] B. Liu, R. L. Grossman, Yanhong Zhai. Mining data records in Web pages. In KDD, pages 601-606, 2003.

[9] D. Cai, S. Yu, J. Wen, W. Ma. Extracting Content Structure for Web Pages Based on Visual Representation. In APWeb, pages 406-417, 2003.

[10] D. Cai, X. He, J. Wen, W. Ma. Block-level link analysis. In SIGIR, pages 440-447, 2004.

[11] D. Cai, X. He, Z. Li, W. Ma, J. Wen. Hierarchical clustering of WWW image search results using visual, textual and link information. In ACM Multimedia, pages 952-959, 2004.

[12] X. Gu, J. Chen, W. Ma, G. Chen. Visual Based Content Understanding towards Web Adaptation. In AH, pages 164-173, 2002.

[13] H. Zhao, W. Meng, Z. Wu, V. Raghavan, C. T. Yu. Fully automatic wrapper generation for search engines. In WWW, pages 66-75, 2005.

[14] K. Simon, G. Lausen. ViPER: Augmenting Automatic Information Extraction with Visual Perceptions. In CIKM, pages 381-388, 2005.