Visual Segmentation-Based Data Record Extraction from Web Documents

Longzhuang Li
Dept. of Computing Sciences
Texas A&M Uni.-Corpus Christi
lli@sci.tamucc.edu

Yonghuai Liu
Dept. of Computer Science
Uni. of Wales, berystwyth
yyl@aber.ac.uk

Abel Obregon,
Matt Weatherston
Dept. of Computing Sciences
Texas A&M Uni.-Corpus Christi

Abstract

Semi-structured data records contained in the Web pages provide useful information for shopping agents and metasearch engines. In this paper, we present a visual segmentation-based data record extraction (VSDR) method to extract data records from those Web pages. VSDR method first segments a Web page into semantic blocks using the spatial closeness and visual resemblance of data records, then neighboring and non-neighboring data records are extracted based on a compress and collapse technique. Experimental results show that unlike the existing methods which only generate good results on their test domains, VSDR is a general data record extraction method that is able to produce quite stable and good results on a wide range of Web pages.

1. Introduction

The World Wide Web has become one of the most important information resources. Although most of the information is in the form of unstructured text, a large amount of semi-structured objects, called data records, are contained on the Web. These Web pages are typically created dynamically from the back-end database of the host site. Identifying and extracting those data records in Web pages becomes valuable because it not only enables us to combine information from multiple Web sites to provide useful service, such as comparative shopping and meta search, but also rebuild the back-end database of the host Web site.

Existing Web data extraction methods can be classified into three categories according to their degree of automation. The three categories are manual, semi-automatic, and fully automatic methods. The manual methods identify each data record by writing a program, called a wrapper, which is based on a Web page's specific presentation layout or contents. These methods can not scale to a large number of pages. The semi-automatic methods rely on machine learning and human assistance to separate objects. Human assistance makes these semi-automatic methods highly time consuming and unsuitable for the fast-changing Web pages. In addition, the induced wrapper is not generalizeable to other Web pages with different structures. More recently, fully automatic methods without human involvement have attracted more attention.

The current fully automatic methods, such as MDR [3], ViNTs [5], and STAVIES [4], perform well in the Web pages of their test domains, but may produce poor results on third party data sets, where most web pages are semi-structured and the organization of web pages is extremely different from each other. In this paper, a new effective visual segmentation-based data record (VSDR) extraction method is developed to automate the process. Data record extraction based on visual segmentation is highly feasible because people often view a Web page as different semantic objects, such as navigational links, advertisement bar, and data record, etc. Moreover, semantically similar objects are usually clustered together and resemble each other in the sense of human perception. As a result, the spatial closeness and visual resemblance make it possible to automatically segment the Web page into several semantic parts [1]. VSDR is a quite general data extraction method which produces good results on a wide range of Web pages.

Given a Web page, the VSDR method consists of following steps: (1) segment the page into visually and semantically similar blocks; then build the hierarchical visual structural tree of blocks; (2) remove the noisy blocks such as navigational bar blocks, dropdown menu blocks, etc.; and (3) in the remaining blocks, identify the data records. Next, the above steps are discussed in more detail.

2. Semantic Web pages segmentation

In the paper, we adapt the VIPS algorithm [1] to perform the initial semantic partition, and store the results in a XML file. Similar to [1], a block is defined as a node or a set of nodes in the DOM tree. The VIPS algorithm employs HTML tag information to partition a Web page into a set of blocks with each block containing related information. In particular, various visual cues, such as font, color, and size, are taken into account to achieve an accurate content structure on the semantic level. VIPS first extracts all the suitable nodes from the HTML DOM tree, and then finds the separators between these nodes.

In Figure 1, a Web page is divided into two blocks VB1-1(6) and VB1-2(9) (the number inside the parentheses is the degree of coherence of the block and will be ignored in the following discussion. For more detail, please see [1]), and VB1-1 is further divided into VB1-1-1, VB1-1-2, and VB1-1-3. VB1-1-1 mainly has a dropdown menu and a text form for search query input. VB1-1-2 consists of six clickable text links such as *computer/Electronics* and *Video Game*, etc. VB1-1-3 holds two blocks, VB1-1-3-1 and VB1-1-3-2, each corresponding to a data record. The corresponding block tree is shown in Figure 3 and more detail will be discussed in section 2.1.

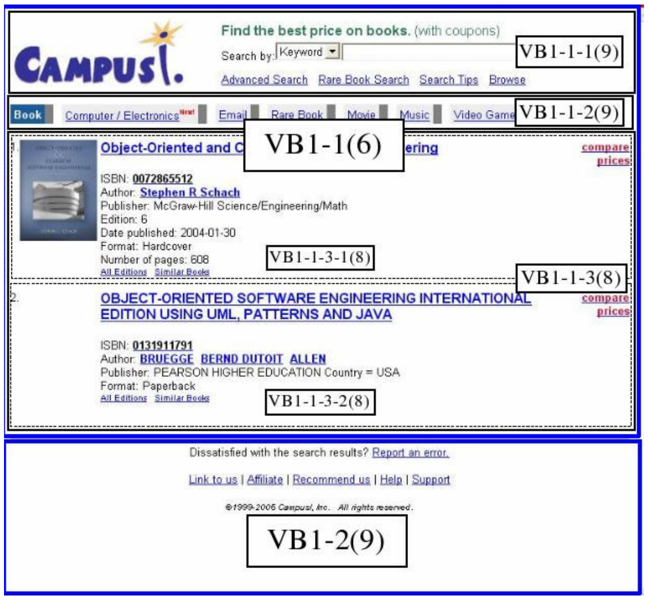

CAMPUS!
Find the best price on books. (with coupons)
Search by: Keyword
Advanced Search Rare Book Search Search Tips Browse
VB1-1-1(9)
Book Computer / Electronics Email Rare Book Movie Music Video Game
VB1-1-2(9)
VB1-1(6)
1. Object-Oriented and Classical Software Engineering
compare prices
ISBN: 0072865512
Author: Stephen R Schach
Publisher: McGraw-Hill Science/Engineering/Math
Edition: 6
Date published: 2004-01-30
Format: Hardcover
Number of pages: 608
All Editions Similar Books
VB1-1-3-1(8)
VB1-1-3(8)
2. OBJECT-ORIENTED SOFTWARE ENGINEERING INTERNATIONAL EDITION USING UML, PATTERNS AND JAVA
compare prices
ISBN: 0131911791
Author: BRUEGGE BERND DUTOIT ALLEN
Publisher: PEARSON HIGHER EDUCATION Country = USA
Format: Paperback
All Editions Similar Books
VB1-1-3-2(8)
Dissatisfied with the search results? Report an error.
Link to us | Affiliate | Recommend us | Help | Support
©1999-2006 Campus!, Inc. All rights reserved.
VB1-2(9)

Figure 1. Web page segmentation.

The VIPS algorithm can not determine the data regions or data record boundaries because it is not developed for this purpose, but the VIPS block tree provides the important semantic partition information of a Web page:

1. Similar data records are typically presented in one or more contiguous regions, with one major region containing most data records.

2. In a VIPS block tree, similar data records usually are siblings, and a leaf or terminal block is not a data record because a data record can be further partitioned into more than one subblocks.

3. In a VIPS block tree, a data record is usually self-contained in a subtree and contains at least two different types of blocks.

2.1 Build the block tree

In this paper, a VIPS segmentation block is categorized as one of the following eight types: text block (T), text link block (TL) (which may include the non-anchor text around the URL as well as several URLs), image block (I), image link block (IL), dropdown menu block (DM), text box block (TB), action button block (AB) including checkboxes and radio buttons, and visual block (VB). In a VIPS block tree, the first seven types of blocks (T, TL, I, IL, DM, TB, and AB) are leaf or terminal nodes which can not be further partitioned. The last type of blocks (VB) are internal nodes which are further divided into several smaller VBs, terminal blocks, or both. For example, in Figure 1, the blocks, such as VB1-1, VB1-2, VB1-1-1, VB1-1-2, VB1-1-3, VB1-1-3-1, and VB1-1-3-2, are VBs.

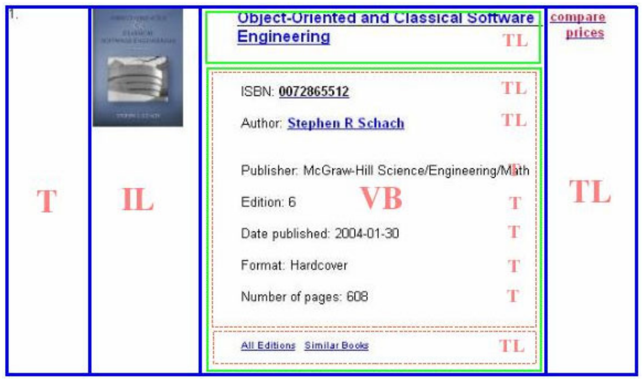

1.
Object-Oriented and Classical Software Engineering
compare prices
ISBN: 0072865512
Author: Stephen R Schach
Publisher: McGraw-Hill Science/Engineering/Math
Edition: 6
Date published: 2004-01-30
Format: Hardcover
Number of pages: 608
All Editions Similar Books

Figure 2. Segmentation of VB1-1-3-1 in Figure 1.

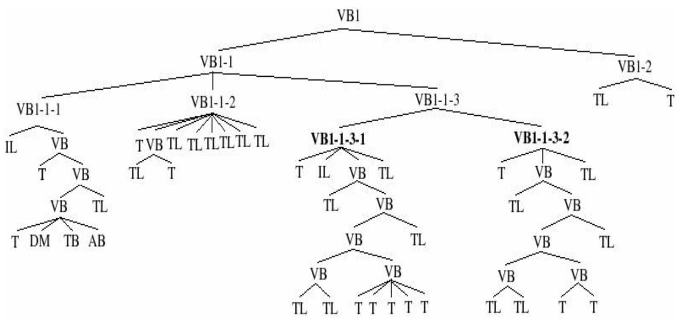


Figure 3. VIPS block tree of the Web page in Figure 1.

Before we discuss how to discover the data records in a VIPS block tree, let us get in more details on the partition of data record block VB1-1-3-1 in Figure 1. In Figure 2, VB1-1-3-1 is partitioned into four blocks in left-to-right order, a T with only a single digit *1*, an IL, a VB, and a TL with the anchor text *compare prices*. Of the partition of the VB (from top), the first three and the last blocks are TLs, the five blocks in the middle are Ts. The second and the third TLs contain both the anchor text and the non-anchor text. For example, the third TL consists of the non-anchor text *Author:* and the anchor text *Stephen R. Schach*. The last TL holds two links, *All Editions* and *Similar Books*. Under the block tree hierarchy, all but the first blocks form a VB, the second and the third TLs compose a VB, and the middle five Ts form another VB. According to the above analysis, the corresponding block subtree for VB1-1-3-1 is shown in Figure 3. Similarly, the block subtree for VB1-1-3-2 can be built (see Figure 3).

2.2 Post-process the output of VIPS algorithm

There are two problems with VIPS algorithm: (1) VIPS stops partitioning process earlier when it should continue on some Web pages; and (2) VIPS may partition two visually similar data records into different block tree structures.

We call the first problem "the pre-maturity problem". One example of the pre-maturity problem in VIPS lies in the fact that it may regard a complex block containing multiple different block types, such as T and TL, as a leaf node. In this case, a leaf node may consist of only one data record or contain several data records. Our solution is to scan through each leaf node, and divide the leaf node into one or several smaller regions according to their relative physical location, especially the distance between two neighboring lines. The heuristic used here is that the content of the same data record should be closely presented together. Then a divided region with multiple block types is considered as an internal node and every block type is a leaf node. For example, Figure 4(a) presents a leaf node, which contains two data records. By calculating the distance between two neighboring content lines, two spikes are discovered with each corresponding to a potential separator. Accordingly, three regions are identified (see Figure 4(b)).

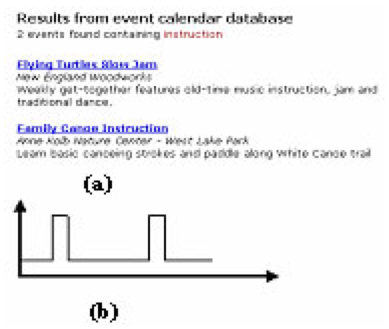

Results from event calendar database
2 events found containing instruction
Flying Turtles Slow Jam
New England Woodworks
Weekly get-together features old-time music instruction, jam and traditional dance.
Family Canoe Instruction
Anne Kolb Nature Center - West Lake Park
Learn basic canoeing strokes and paddle along white canoe trail
(a)
(b)

Figure 4. data region clustering.

There are two solutions for the second problem. The first solution is to re-partition the data records to a similar block tree structure. The second solution is that we keep the data record partitions as they are, but we compute their resemblance based on the string edit distance [2] by skipping the internal nodes. In this paper, we choose the second solution (For more detail, see section 4.2).

3. Identify noisy blocks

In the VSDR method, several heuristics are employed to identify noisy blocks. The cleaning process begins with the first-level subtrees, such as VB1-1 and VB1-2 in Figure 3. We only retain the first-level subtrees containing at least two sibling internal nodes, because it is observed that a data record usually itself forms a VB and we need at least two data records to do the extraction. The first-level subtree such as VB1-2 in Figure 3 can be safely eliminated because it has no internal nodes as its children and is unlikely to contain any data records.

As for the rest of the first-level subtrees, they may still only contain noise such as navigational bars/links, text boxes, dropdown menus, action buttons, etc. The navigational bar blocks, such as VB1-1-2 (see Figure 1 and Figure 3), can be identified if the number of contiguous link blocks is at least 5 and the ratio of the number of link blocks (including both text links and image links) to the number of all blocks in the same tree level is greater than a specific threshold. The ratio is preset to 0.75 in our experiments. These noisy subtrees are removed before extracting data records.

Similarly, the visual blocks containing text boxes, dropdown menus, and/or action buttons, and occupying a relatively small area of the whole page, will also be deleted as noise. For example, VB1-1-1 in Figure 1 is located at the head area of the page and mainly consists of the combination of dropdown menu, text boxes, action button, as well as a list of links. The simple heuristics used in this stage have been proved to be highly effective. With these heuristics we can remove most noisy blocks to speed up the data extraction procedure. As a matter of fact, there is only one first-level subtree left after deleting noise from many Web pages. In our case, only VB1-1-3 is left.

4. Data record extraction

4.1 Leaf node reduction

It is very common that even two similar data records in the same segment may still have different number of attributes. For example, in Figure 1, the second data record (VB1-1-3-2) lacks the three attributes *Edition*, *Date published*, and *Number of pages* existing in the first data record (VB1-1-3-1). If the two subtrees VB1-1-3-1 and VB1-1-3-2 are compared directly without any node reduction, the similarity between them may become small.

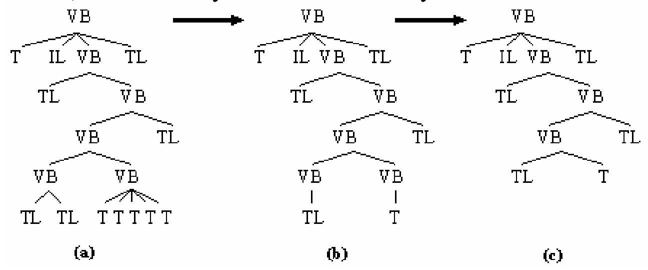


Figure 5. Reduction of block VB1-1-3-1 in Figure 3.

The leaf node reduction process includes two procedures: compress and collapse. In the compress stage, all but one of the sibling leaf nodes of the same type in the tree are removed. In the collapse stage, the internal nodes (VBs) with one child are cut off and the only child is raised up one level. This is a repeating process which will not stop until there are no blocks to be compressed and collapsed. For example, in Figure 4(a), the two sibling TLs and five sibling Ts at the lowest level of the subtree VB1-1-3-1 are compressed to one TL and one T, respectively (see Figure 4(b)). Then the two VBs with only one child are collapsed with the TL and the T attached to the parent of the two deleted VBs (see Figure 4(c)). Similarly, the subtree VB1-1-3-2 can be reduced.

4.2 Block similarity using edit distance

In this paper, we calculate the similarity between two blocks by normalizing their edit distance [2,3]. The edit distance of two blocks, B_1 and B_2, is defined as the minimum number of edit operations needed to change B_1 into B_2, where the allowed edit operations are insertions, deletions, and substitutions.

The similarity of two blocks is obtained by the following formula:

$$BS(B_1, B_2) = 1 - \frac{ED(B_1, B_2)}{Avg((Len(B_1) + Len(B_2))}$$

where the second term on the right hand side of the equation is the normalized edit distance (dividing the edit distance *ED(B₁, B₂)* by the average length of the two blocks).

When we compute the similarity between two blocks, we ignore all the internal nodes (VBs). For example, block VB1-1-3-1 in Figure 5(c) is expressed as the string (T IL TL TL T TL TL) in the depth-first traversal.

4.3 Identify data records in each block

This is a depth-first comparison process. The basic idea is to traverse down the tree and compute the similarity between each successive pair of internal nodes (VBs and their subtrees) using edit distance. The pairs with close resemblance are marked down as candidate data records. Internal nodes not similar to each other are further explored by going down one level and repeating the above pair wise comparison if the height of the internal node is greater than two. In the following discussion, comparing two internal nodes means comparing two subtrees with the two internal nodes as their roots respectively.

4.4 Extract neighboring data records

It is observed that similar data records are commonly contained in the same internal node, and in most cases those data records are siblings in the block tree. For similar data records which are not siblings, after applying the above leaf node reduction process to the internal node reduction, it is highly likely that they become siblings because the similar data records may be partitioned into visual blocks at different levels of the block tree and usually the level difference is as small as 1. For example, in Figure 6(a), VB1-1-1-1, VB1-1-1-2, and VB1-1-2 are three similar data records located at different levels of subtree VB1-1. After comparing the two sibling subtrees VB1-1-1-1 and VB1-1-1-2 and identifying them as two candidate data records, the two subtrees VB1-1-1-1 and VB1-1-1-2 are compressed to VB1-1-1-1(2) (see Figure 6(b), number 2 in the parentheses is the number of occurrences of block VB1-1-1-1) and collapsed to VB1-1-1 (see Figure 6(c)), then the two subtrees VB1-1-1 and VB1-1-2 are compared and reduced again. In particular, we keep tracking the number of occurrences of reduced internal nodes. This process is called the internal node reduction.

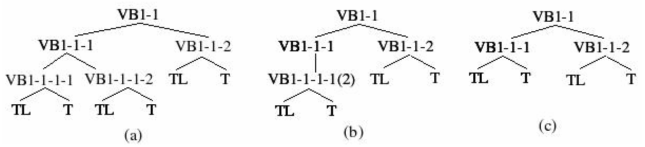


Figure 6. Reduction of visual blocks.

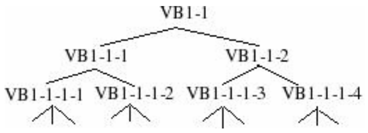


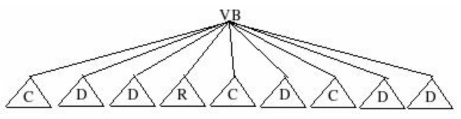
Figure 7. Similar blocks with more than one data record.

Sometimes a candidate data record may contain several real data records. For example, in Figure 7, four similar data records VB1-1-1-1~VB1-1-1-4 are partitioned under two internal nodes VB1-1-1 and VB1-1-2 with VB1-1-1-1~VB1-1-1-2 in VB1-1-1 and VB1-1-1-3~VB1-1-1-4 in VB1-1-2 respectively. In the sibling comparison procedure mentioned above, VB1-1-1 and VB1-1-2 are marked as the candidate data records. The heuristic we use for this case is if candidate data records do not have any leaf nodes as their children but have at least two internal child nodes, we only need to go down one more level and compare the siblings again. If all siblings are similar, they are real data records. Otherwise, their parent nodes are real data records. In Figure 7, VB1-1-1-1~V1-1-1-2 and V1-1-1-3~V1-1-1-4 are compared respectively and each of them is finally considered as a data record.

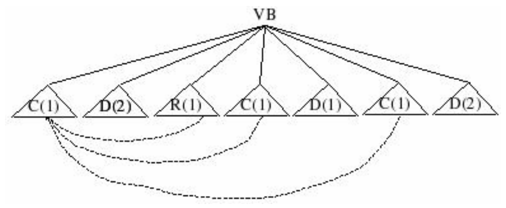
4.5 Extract non-neighboring data records

The current data extraction methods, such as MDR and ViNTs, can not generate satisfactory results for this scenario. Here, we have two cases to consider:

Case 1: Although similar data records are siblings, they are not all neighboring to each other and are separated by non-data record internal nodes or leaf nodes. For example, in Figure 8(a), the data record D immediately next to two Cs will be missing by simply applying the successive pair wise sibling comparison. In Figure 8, each triangle represents a subtree, and the letter inside the triangle is the subtree type: C stands for category, D for data record, and R for related content. Figure 8 can be interpreted as follows: under the first category C (from left), there are two date records Ds as well as some related information R. The rest of the subtrees can be similarly interpreted. The question here is how we discover which subtrees are data records. Our solution consists of two steps: (1) count and compress the neighboring subtrees of same type by using the method in section 4.4 (see Figure 8(b), the number inside the parentheses is the number of occurrence of neighboring subtrees of the same type); (2) add the number of occurrences (NOC) of each subtree type by additional pair wise comparison. The subtree type with the biggest NOC value is the one for data records. For example, in Figure 8 the NOC of C is 3, D 5, and R 1 respectively. As such, subtree type D are data records.



(a)



(b)

Figure 8. An example of separated data records.

Finding the NOC of each subtree type in step 2 is in fact very fast because we already know that the neighboring subtrees are of a different structure. In particular, we do not compare neighboring subtrees and exclude the similar subtrees from further comparison. For example, in Figure 8(b) starting from the left-most subtree C(1), we compare C(1)-R(1), C(1)-C(1), and C(1)-C(1) (see the dotted lines). Here, the similarity of the first C(1)-C(1) pair and the second C(1)-C(1) pair excludes D(1) and D(2), respectively. From the result of the first round of comparison, the NOC of C is 3 and the last two Cs are excluded from future consideration. Then we move to the second left-most subtree D(2), where we only have two more comparison D(2)-D(1) and D(2)-D(2). The NOC of D is added to 5 and again the last two Ds are removed from the comparison list. For the third left-most subtree R(1), we have no subtree left to compare.

A special scenario of this case is that we have the same NOC for all the subtree types. For example, when we have the interleaved occurrence of C and D under one internal node (such as CDCDCD), the combination of C and D is considered as a data record. An example of this scenario is that the title and body of a data record are partitioned as two neighboring sibling nodes.

Case 2: This is a more complex case, where data records may not be siblings as well as may not be at the same level, but they are all under the same parent node. For instance, in Figure 9, the triangle Ds are under different subtrees and can not be reduced to siblings by conducting the reduction process described in section 4.4. The basic idea for this case is to gradually flatten the subtrees from the top and count the NOC of subtrees for each type. This process will not stop until all the different subtrees have been reduced.

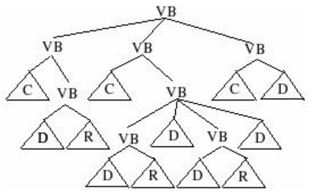


Figure 9. Another example of separated data records.

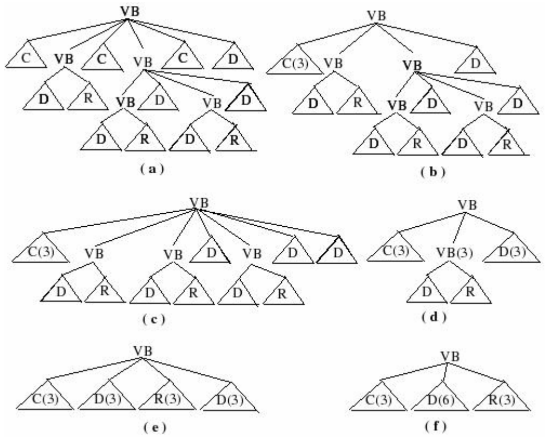


Figure 10. Identify data records by flattening subtrees.

Let us use Figure 9 as an example to illustrate the flattening process. First, after collapsing the first level of internal nodes (three sibling VBs) in Figure 9, we have the subtrees in Figure 10(a). Based on the pair wise comparison in Case 1 of section 4.4, we find that the NOC of C is 3 (see Figure 10(b). Then we choose the subtree with the largest height to flatten because it is most likely that this subtree contains data records (see Figure 10(c)). The NOC of different subtrees is shown in Figure 10(d). VB(3) is our next choice to flatten because it is deeper than both C(3) and D(3) and may still contain data records, so we contract VB(3) again (see Figure 10(e)) and receive the NOC of C, D, and R as 3, 6, and 3, respectively (see Figure 10(f)). At this stage, all Ds are identified as data records.

5. Experiments

In the experiments, we use three data sets to compare the performance of our data VSDR method with the two existing automatic systems, MDR and ViNTs.

The three data sets come from different resources. The first data set (Data 1) is the DataSet 3[1] used by ViNTs. The second data set (Data 2) is downloaded from the manually labeled *Testbed for Information Extraction from Deep Web TBDW* ver. 1.02[2]. TBDW holds query results from 51 search engines, and there are five query result pages for

[1] http://www.data.binghamton.edu:8080/vints/testbed.html

[2] http://daisen.cc.kyushu-u.ac.jp/TBDW

each search engine. In Data 2, we only collect the first result page (1.html) of each search engine. We gather the third data set (Data 3) from the home pages listed in the MDR paper [3] (MDR paper does not provide the URLs of real data it tested). The number of Web pages for each of the three data sets is shown in the third row of Table 1.

The performance measures we use to compare the three methods are *recall* = E_c/N_t and *precision* = E_c/E_t, where E_c is the total number of correctly extracted data records, E_t the total number of records extracted, and N_t the total number of data records contained in all the Web pages of a data set.

Table 1. Performance comparison of MDR, ViNTs, and VSDR on three data sets.

	Data 1			Data 2			Data 3		
	MDR	ViNTs	VSDR	MDR	ViNTs	VSDR	MDR	ViNTs	VSD R
# Web pages	41	41	41	46	46	46	33	33	33
# DRs	833	833	833	1004	1004	1004	605	605	605
# Extracted DRs	1068	809	861	1130	962	1077	786	627	698
# Correct DRs	686	800	808	609	940	964	532	318	604
Recall	82.4%	96.0%	97.0%	60.7%	93.6%	96.0%	87.9%	52.6%	99.8%
Precision	64.2%	98.9%	93.8%	53.9%	97.7%	89.5%	67.7%	50.7%	86.5%

Experimental results are shown in Table 1. For all three data sets, VSDR achieves the best recall values, which are 97.0%, 96.0%, and 99.8% respectively. ViNTs method comes second for Data 1 and Data 2, and third for Data 3 for the recall value. VSDR outperforms both MDR and ViNTs because VSDR method considers multiple data regions and extract non-neighboring data records as well (see section 4.5). ViNTs only extracts data records from one major data region and MDR misses many data records although it searches through multiple data regions.

ViNTs has the best precision rate on Data 1 and Data 2, but the worst on Data 3. VSDR ranks the second on Data 1 and Data 2, and first on Data 3. The reason is that the data records in Data 1 and Data 2 are mostly presented in one major data region. MDR suffers from generating too much noise for all three data sets.

It is shown from Table 1 that none of the three methods performs the best at all times in terms of recall and precision. But VSDR produces very stable and good results through a wide range of Web domains.

6. Related Works

MDR [3] directly compares the resemblance of HTML tag strings to decide the data region. One problem with the MDR method is that the result page may contain a lot of noisy data such as link blocks which should be removed. Another problem is that in some cases it cannot correctly divide data records. Instead, it returns the whole data regions. ViNTs [5] works best on search engine results. The method assumes that the data records are in a contiguous region. If there is noise (advertisement) in the middle, the method classifies either the noise as data records or entirely misses the data records because of noise interruption. STAVIES [4] employs clustering techniques to segment the Web pages and locate the region that contains the data records as well as the boundaries separating them. This method is restricted by using the cardinality of common ancestors of two nodes as the similarity measure.

7. Conclusion

In this paper, we discuss a visual segmentation-based method to extract data records from Web pages. Our experimental results show that none of the three methods works the best all the time, but the VSDR method can achieve stable and good results on a wide range of Web pages.

8. References

[1] Cai, D., Yu, S., Wen, J., and Ma, W. *VIPS: A vision-based page segmentation algorithm.* Microsoft Technical Report MSR-TR-2003-79, 2003.

[2] Gusfield, D. *Algorithms on strings, trees, and sequence*, Cambridge. 1997.

[3] Liu, B., Grossman, R., and Zhai, Y. Mining data records in web pages. In *Proc. of the ACM SIGKDD*, 2003.

[4] Papadakis, N. K., etc. A system for information extraction from unknown web data sources through automatic web wrapper generation using clusting techniques. *IEEE TKDE,* 17, 12 (Dec. 2005).

[5] Zhao, H., etc. Fully automatic wrapper generation for search engines. In *the 14th Intl. Conf. on WWW, 2005.*