Int J Software Informatics, Volume 6, Issue 3 (2012), pp. 453–472
International Journal of Software and Informatics, ISSN 1673-7288

E-mail: ijsi@iscas.ac.cn
http://www.ijsi.org
Tel: +86-10-62661040

Web Data Extraction from Query Result Pages Based on Visual and Content Features

Daiyue Weng, Jun Hong, and David A. Bell

(School of Electronics, Electrical Engineering and Computer Science,
Queen's University Belfast, Belfast BT7 1NN, UK)

Abstract A rapidly increasing number of Web databases are now become accessible via their HTML form-based query interfaces. Query result pages are dynamically generated in response to user queries, which encode structured data and are displayed for human use. Query result pages usually contain other types of information in addition to query results, e.g., advertisements, navigation bar etc. The problem of extracting structured data from query result pages is critical for web data integration applications, such as comparison shopping, meta-search engines etc, and has been intensively studied. A number of approaches have been proposed. As the structures of Web pages become more and more complex, the existing approaches start to fail, and most of them do not remove irrelevant contents which may affect the accuracy of data record extraction. We propose an automated approach for Web data extraction. First, it makes use of visual features and query terms to identify data sections and extracts data records in these sections. We also represent several content and visual features of visual blocks in a data section, and use them to filter out noisy blocks. Second, it measures similarity between data items in different data records based on their visual and content features, and aligns them into different groups so that the data in the same group have the same semantics. The results of our experiments with a large set of Web query result pages in different domains show that our proposed approaches are highly effective.

Key words: web data mining; web data extraction; data record extraction; web data alignment

Weng DY, Hong J, Bell DA. Web data extraction from query result pages based on visual and content features. ***Int J Software Informatics*, Vol.6, No.3 (2012): 453–472. http://www.ijsi.org/1673-7288/6/i133.htm**

1 Introduction

The volume of structured data on the Web has been increasing enormously. Such data is usually returned from the back-end databases in response to specific user queries, and presented in the form of data records, which are enwrapped in query result pages. In the literature, the contents stored in web databases are referred to as the deep Web and query result pages are referred to as deep web pages. A study[12] in 2004 reveals that the scale of web databases that are 'hidden' on the Web is well in the order of 10^5 and continues expanding rapidly. Another estimate is that the

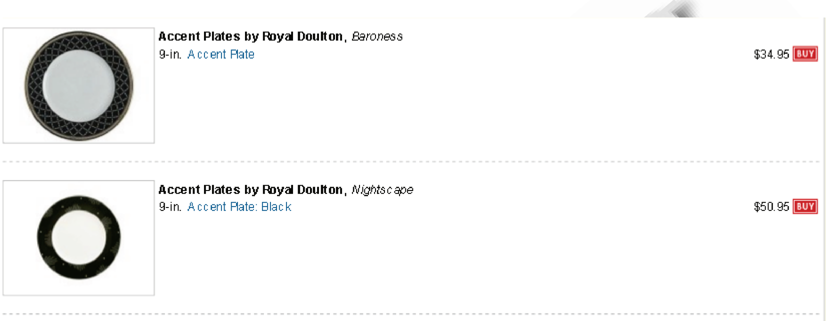
---

Corresponding author: Daiyue Weng, Email: dweng01@qub.ac.uk

Received 2011-11-14; Revised 2012-07-19; Accepted 2012-08-21; Published online 2012-09-18.

number of Web databases has reached 25 millions[27]. Many e-commerce sites are supported by web databases.

In general, majority of query result pages are list pages. A list page contains several data records in multiple columns with each row on each column representing a data record. For example, Fig. 1 shows a snapshot of a list page from cooking.com, which has a single column containing two data records. Each record represents a plate with several data items e.g. name, price, model etc.



Accent Plates by Royal Doulton, *Baroness*
9-in. Accent Plate
$34.95 BUY

Accent Plates by Royal Doulton, *Nightscape*
9-in. Accent Plate: Black
$50.95 BUY

Figure 1. An example of Web data records in a list Page

Extracting data records from query result pages enables integrating data from various Web databases to provide value-added web applications, such as, price comparison sites and meta-search engines etc. Query result pages are dynamically generated from back-end databases in response to user queries and encoded in HTML using pre-defined templates or script programs. These pages are semi-structured and displayed for human use, rather than for processing by programs. How to automatically extract data records is a very challenging problem.

There has been intensive research on fully-automatic approaches[3−11,13−16,18] for extracting data from query result pages. These approaches first represent input pages as either various kinds of tree structures or token strings. They then try to identify and extract data by exploring the regularities within these representations and optionally the regularities of visual features that are inherently exhibited in the pages. The work in Ref. [18] transforms a page into a Visual Block tree[19] and primarily uses the visual features of the page to identify data sections, extract data records and align data items. The methods in Refs. [13–16] extract data from query result pages by identifying repetitive patterns or templates in the token strings of the result pages. Among these approaches, those in Refs. [3-10] represent the current technical trend of query result extraction. First, they identify a data section, which contains a set of data records. Second, they identify data records from each data section. Finally, they extract data by aligning the corresponding attributes of different records, producing a relational table[4, 5, 8,10]. The approach in Ref. [11] also extracts labels as attribute names for extracted attributes. This is a mainstream extraction strategy that has proved very efficient and effective based on their experimental results.

However, the existing fully-automatic approaches to query result extraction have the following limitations. First, Web pages are becoming more and more complex. Their tag structures are growing more and more complex since HTML itself is evolving, and other technologies like JavaScript and CSS are widely deployed to make result pages more dynamic. This may make the layouts of result pages different from their DOM tree or token string representations. Thus the existing approaches that rely on such representations may fail to extract data records from the pages. Second, many current techniques employ a similarity measure on page segments of DOM trees to identify data records. However, data records may not be extracted correctly if the sibling DOM tree segments of the same root are dissimilar to each other. This also makes it impossible to extract a single record in a data section. The work in Ref. [7] can extract one or more data records within a data section by identifying tag forest separators which are used to partition the data section. However the wrappers it generates for extracting multiple data records require that the tag structures of all the records in the same data section must be siblings under the same sub-tree. Some data sections may contain tag structures that are not siblings. Third, most of current approaches do not filter out noisy contents or the result of eliminating noisy contents is not satisfactory. Noisy contents refer to any part of a query result page that is not part of any data record. For example, banner advertisements, navigation bar, copyright notice, record statistical information etc are noisy contents. In other words, we are most interested in the part of a result page which contains all the data records with little noisy content which often affects the accuracy of data record extraction. Thus it is very important to remove any noisy content before data record extraction.

In this paper, we focus on the following problem: Given a query result page that contains a single column of one or multiple data record(s), automatically identify the data section and data records in the page; automatically extract and align data items from the data records. We propose an approach that consists of two phases.

In the first phase, given a query result page, our approach first identifies data records in the page based on a visual block partition of the page in terms of visual adjacency. In particular, the method for data record extraction performs in three steps:

Step 1: The approach transforms a query result page into a Visual Block tree, which represents a visual partition of the Web page. Such a representation reflects the content organization of the page enforced by visual cues so that content related parts will be represented in the same branch of the Visual Block tree. For example, Fig. 2 shows a visual partition of a result page (Fig. 2(a) and (b)) and the corresponding Visual Block tree (Fig. 2(c)) created by the VIPS algorithm[19]. We can also get visual features (e.g. position, width, height etc) of each block on the Visual Block tree.

Step 2: We observe that a data section in a result page, which contains all the data records, usually occupies a significant area of the result page. We also observe that query terms usually re-appear in the data records. We use these two observations to identify the data section by exploiting the sizes of the blocks of the Visual Block tree of the result page, and counting the occurrences of query terms in them. For example, block $b_{1\_1\_2\_2\_3\_4}$ in the Visual Block tree in Fig. 2(c) is the data section since it occupies a large portion of the result page in Fig. 2(a) and contains a number of occurrences of query terms (e.g."Accent Plates"). To get query terms, we make use of the query interface and assume that the result pages are generated in response to the queries containing query terms.

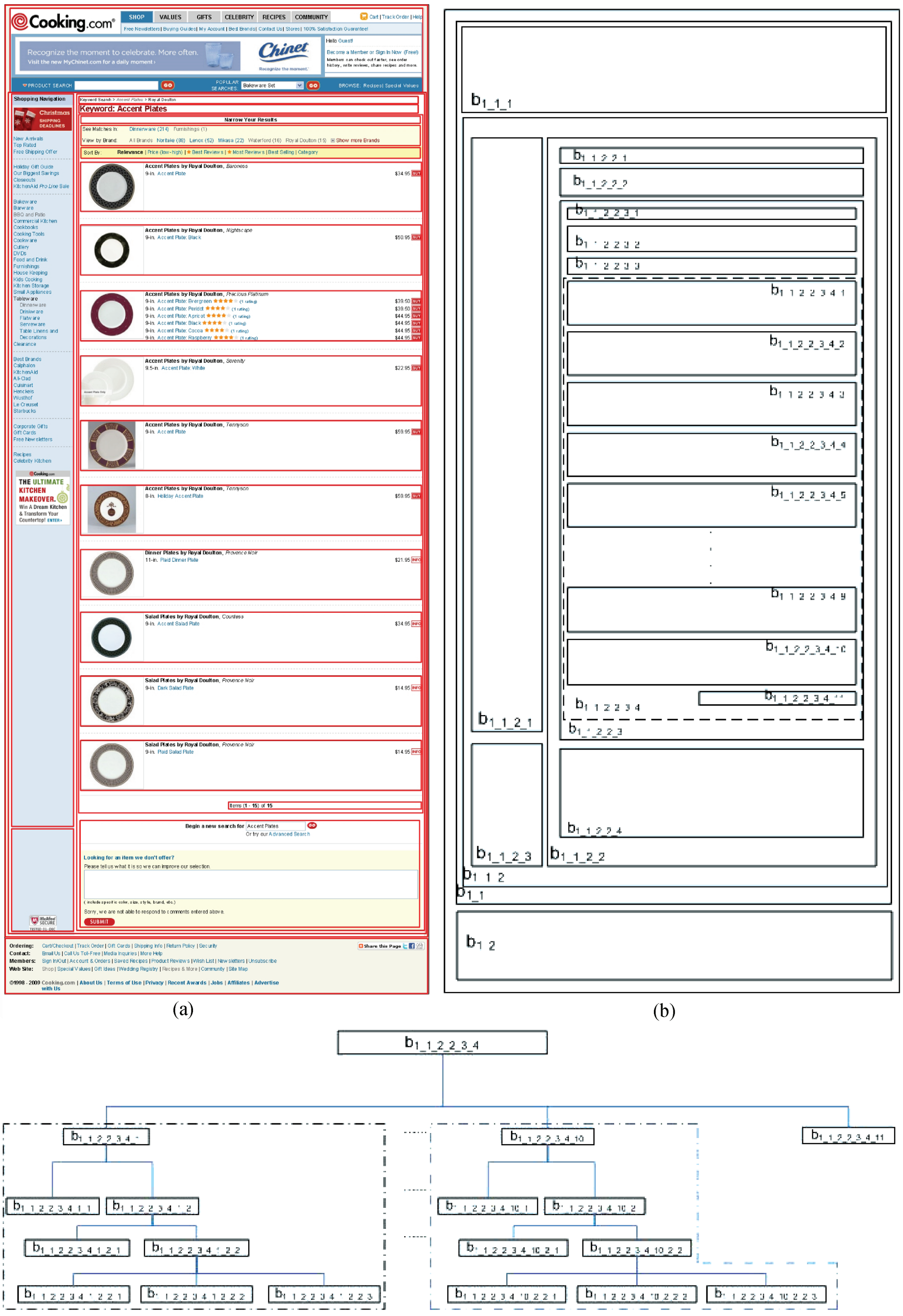
The identified data section often contains noisy blocks. Data records are obviously more vivid in content than noisy blocks, have one or more links or some images. To filter out noisy blocks, we use a number of content and visual features to characterize each block within the data section. These features provide statistical information about texts, block areas, links and images in the block. The overall importance of a block for a data record is apparently higher than noisy blocks. We set up a threshold of importance to evaluate the importance of each block. Less important blocks are noisy blocks that can be removed. For example, as shown in Fig. 2(a) there are ten data record blocks and a block containing information about data records shown ("Items (1 - 15) of 15") which is a noisy block.

Step 3: We observe that each data record contains semantically related data units of a data object, which reside in the leaf nodes of the Visual Block trees and are visually adjacent to each other. We identify data records by purely using the positional information of the rendering boxes of the leaf nodes in the data section. For example, the data units of each data record shown in Fig. 1 are adjacent to each other and relatively far away from the data units of the other data records. Thus we can group data units that are adjacent to each other so that each group representing a data record.

In the second phase, a similarity-based clustering method is proposed for aligning data items from the extracted data records, and put the data items into a relational table.

We observe that data items with the same semantics have similar visual and content features. They usually share certain keywords (e.g., the product name "Accent Plates" in Fig. 1), and have the same data type (e.g., the product names are strings). They have similar presentation styles (e.g., the product names in Fig. 1 have the same font face, font size and font color etc) and have approximately same block size (e.g., in Fig. 4(a), the blocks that wrap product names have somewhat identical sizes). Moreover, they are usually encoded by the same tag template (e.g., the product names in Fig. 1 are encoded in "$\langle TD\rangle\langle B\rangle text\langle /B\rangle\langle B\rangle text\ \langle /B\rangle\langle I\rangle text\langle /I\rangle\langle /TD\rangle$", where "text" represents the data enwraped). Therefore, all the data items in the data records can be clustered into different groups based on similarity measures on the above features, so that the data items contained in each group have the same semantics (e.g., all product names are aligned into the same group).

In summary, we make the following contributions. For data record extraction: (1) we propose an approach for identifying data sections based on the visual features of the blocks and the occurrences of query terms, even as small as containing only one data record. (2) Based on content and visual features of visual blocks, our solution for removing noisy blocks can eliminate most of the noisy blocks. (3) We propose an approach for identifying data records based on an observation that the data units of a data record are visually close to each other and distant from the data units of the other data records on the page. By grouping data units that are adjacent to each other, we will not miss any record that is dissimilar to its siblings, and we are also able to extract data records from pages, each of which containing one record only. Readers can refer to Ref. [28] for details. For data alignment: (1) We propose a method for aligning data items based on a number of visual and content features so that data items from different data records with the same semantics are clustered together. (2) We devise an algorithm for constructing relational tables from aligned data items considering optional data items and data items from nested data records.



(a)

(b)

(c)

Figure 2. (a) An example query result page; (b)The visual block layout of the page;

(c) Part of the Visual Block tree for $b_{1\_1\_2\_2\_3\_4}$

The rest of this paper is organized as follows. Section 2 presents web page representation, problem definition and an overview of our approach. Section 3 describes our approaches for identifying data sections, removing noisy blocks and identifying data records. Section 4 presents approach for aligning data items in the data records. Experimental results are given in section 5. Section 6 discusses related work. Section 7 concludes the paper.

2 Fundamentals and Overview

In this section, we first introduce Visual Block trees and give a formal definition of the rendering box model of Web pages based on the Visual Block tree, which is the basis of our approach. We then define the problem of web data extraction and present an overview of our approach.

2.1 Visual representation of query result pages

The content of a query result page is typically organized into different regions to make it easy for human use, e.g., advertisements, menu bars, sponsor links, query results and so on. Each region contains semantically related contents. Visual cues (e.g. lines, spaces, font sizes, background colours etc) can be used to distinguish regions from each other. To make use of visual features for data record extraction, we employ the VIPS[19] algorithm to represent a query result page as a Visual Block tree. The root of the tree represents the entire page and each node represents a visual block on the page. A leaf node represents a block containing a basic semantic unit that cannot be further decomposed, e.g., a text or image. Node a is an ancestor of node b if the block that a represents contains the block that b represents on the page. The blocks represented by nodes at the same level of the tree do not overlap. The order of the child nodes with the same parent follows the order of the blocks they represent on the page, i.e., top-down, left-right. For example, Fig. 2(a) shows a query result page from cooking.com; Fig. 2(b) shows the visual block layout of the result page produced by the VIPS algorithm. For example, b_1 represents the body of the page, $b_{1\_1\_2\_1}$ represents the block containing the category links on the page, b_{1_2} contains the website information and $b_{1\_1\_2\_2\_3\_4}$ contains all data records denoted as $b_{1\_1\_2\_2\_3\_4\_1}$ to $b_{1\_1\_2\_2\_3\_4\_10}$. Figure 2(c) shows part of the Visual Block tree for $b_{1\_1\_2\_2\_3\_4}$.

2.2 Overview of our approach

The approach takes as input a query result page from a specific Web database and parses it into a Visual Block tree. The output is a relational table that contains all aligned data items from the extracted data records. Our approach is composed of the following modules and processes result pages in order.

1. Data Section Identification: It traverses the Visual Block tree in a depth-first fashion to identify a block that contains all the data records and treats it as a data section.
2. Noisy Block Remover: It tries to remove any noisy information remaining in the data section to improve the accuracy of data record extraction.

3. Data Record Identification: It extracts and groups leaf nodes of the data section in the Visual Block tree into data records based on the positions of their corresponding visual blocks.

4. Data Alignment: It clusters and aligns all data items in the data records of the data section based on visual and content features. The output is a relational table that all the data items are aligned in the table.

3 Identification of Data Sections and Data Records

In this section, we briefly describe our approaches for identifying data sections, removing noisy blocks and extracting data records. The details of these approaches refer to Ref. [28].

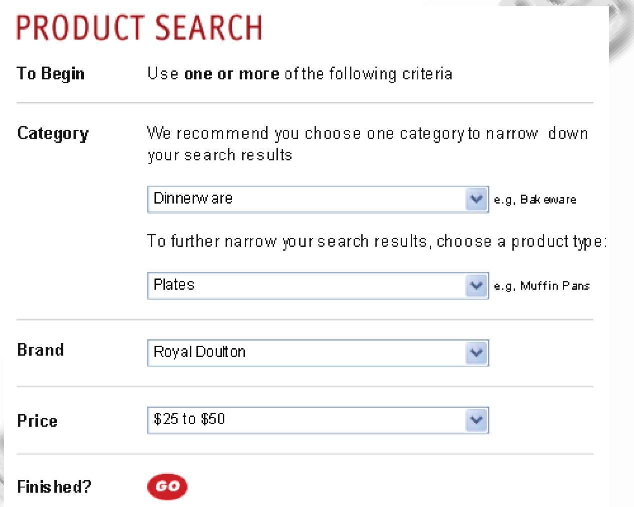
3.1 Identifying data sections

We identify a *data section* as a node in the Visual Block tree, which represents a rectangular box in the result page that contains all the data record blocks and as few noisy blocks as possible. our approach is based on two observations: (1) the size of a data section is usually large relative to the size of the whole page; (2) the query terms often re-appear in the data records.

To utilize the observations, we first select the blocks, so that area ratios between the sizes of the blocks and the whole page are greater than a threshold[18], and identify them as candidate data section blocks. For example, after applying the area ratio constraint, we can identify b_{1_1}, $b_{1\_1\_2}$, $b_{1\_1\_2\_2}$, $b_{1\_1\_2\_2\_3}$ and $b_{1\_1\_2\_2\_3\_4}$ in Fig. 2(b), as candidate data sections. To determine the real data section, we make use of query terms that are used in queries over query interfaces. For example "Dinnerware" "Plates" "Royal Doulton" and "\$25 to \$50" are query terms used for the input elements on the query interface, as shown in Fig. 3, and re-appear in data records shown in Fig. 2(a). The more query terms occur in a block, the more likely the block is a data section. Given a set of query terms q_i for $i = 1, 2, ..., n$, and a candidate block, the importance of the block is measured as $R = \sum_{i=1}^{n} f_i$, where f_i represents the frequency of query term i in the candidate block. The block that has the maximum number of occurrences of query terms among all the candidate blocks is identified as the data section. For example, after applying the second constraint to the candidate data sections, $b_{1\_1\_2\_2\_3\_4}$, as shown in Fig. 2(c), is identified as the data section.

3.2 Removing noisy blocks

The identified data section usually contains noisy blocks that are not part of any data record[18], such as data record numbers (e.g., "Items (1-15) of 15" in Fig. 2(a). We observe that a data record typically contains images, texts and links for the data object, and occupies a significant area on the page. Moreover, the noisy blocks usually exist as first-level child blocks in the Visual Block tree of the data section. Specifically, we evaluate the importance of each first-level child block within the data section is defined as $ImBlk = w_1 \times ImgNum + w_2 \times LinkNum + w_3 \times LinkTextLen + w_4 \times TextLen + w_5 \times Area$, where w_1, w_2, w_3, w_4 and w_5 are real numbers so that $w_1 + w_2 + w_3 + w_4 + w_5 = 1$, and $0 \leqslant ImBLk \leqslant 1$. *LinkTextLen* (the anchor text length of the block) and *TextLen* (the text length of the block) are considered as the most important features for differentiating data record blocks from noisy blocks. *ImgNum* represents the number of images in the block, *LinkNum* represents the number of links in the block, and *Area* represents the rendering area of the block. Note that these content features are provided by the Visual Block tree and are normalized across the whole data section block. When the *ImBlk* of a block is greater than the given threshold θ, it is very likely that the block is a data record. Otherwise the block is taken as a noisy block. The threshold can be trained using sample pages.



PRODUCT SEARCH

To Begin: Use **one or more** of the following criteria

Category: We recommend you choose one category to narrow down your search results

Dinnerware — e.g, Bakeware

To further narrow your search results, choose a product type:

Plates — e.g, Muffin Pans

Brand: Royal Doulton

Price: $25 to $50

Finished?

Figure 3. The query interface of cooking.com

3.3 Identifying data records

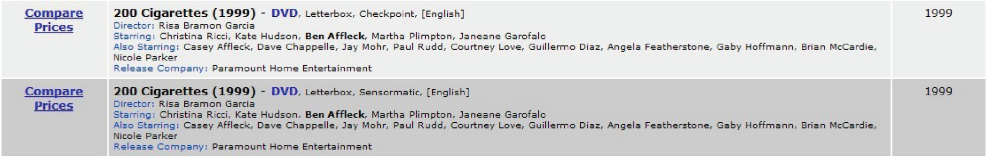
A data record represents a data object retrieved from the web database and consists of multiple data units that are semantically related. Data items are represented as leaf nodes on the Visual Block tree, and they are visually aligned with and adjacent to each other on query result pages. For example, as shown in Fig. 2(a), the data items of each record are the leaf nodes in the Visual Block tree, and they are visually aligned with and adjacent to each other on the web page. To identify data records, our approach takes as input a set of query terms and a data section block. It first identifies leaf nodes that are part of a data record and can be used as starting points, which are leaf nodes that contain query terms, for grouping other data items of the record. Given each of the starting points, our approach first group data items that are horizontally aligned with it to form a data item group based on the top positions of the visual blocks of the corresponding leaf nodes (i.e. they have similar top positions). It then groups data items that are horizontally aligned with each other to form leaf node groups. Finally, our approach progressively expands each data item group with other data item groups and leaf node groups that are vertically adjacent (i.e. the vertical gap between two groups are within a given number of pixels) to it until there is no vertically adjacent group. Each data item group thus identified as output a data record.

To illustrate how the algorithm works, we take the first two data records shown in Fig. 2(a) as an example. “Plates” has been used as a query term. Two leaf nodes representing text “Dinner Plates by” are identified as starting points. These two starting leaf nodes are used to initiate two data item groups. Those leaf nodes representing the second rows of these two data records form two leaf node groups. Each data item group is expanded with a leaf node group which is vertically adjacent to it. Each extracted data item group represents a data record.

4 Data Alignment

The objective of data alignment is to match and cluster data items with the same semantics together. In this section, we propose a new clustering-based approach to data alignment which utilizes a number of visual and content features.

We note that ViDE[18] also uses visual features in data item alignment. However, there are significant differences between ViDE and our approach. First, ViDE assumes that the order of data items is fixed in data records. Given a data record, ViDE extracts its leaf nodes in the Visual Block tree from left to right in sequence. It utilizes this inherent order along with visual features to align a set of the first unaligned data items in every data record. ViDE clusters the data items in such a set that data items with the same font and left position are clustered together, i.e. these data items have the same semantic. If there are multiple clusters, i.e. there are optional data items, ViDE fills blank items into those data records that do not have these optional data items, so that all data records will have the same number of data items. ViDE has two weaknesses. First, the conditions for deciding if the data items have the same semantic, which are purely based on visual features are too strong. Sometimes data items that have the same visual features may belong to different semantics. For example, in Fig. 4, “Director”, “Starring”, “Also Starring” and “Release Company” have the same left position and font, but they belong to different semantics. Hence, we need to use more evidence in addition to visual features. Second, ViDE does not work very well on extracting nested data records containing sub data records, i.e. multi-value attributes. It relies on the order of the data items and simply aligns those data items that expand horizontally and occupy more columns, hence it treats nested data records as flat ones. Therefore, we need to introduce a clustering algorithm that groups data items bottom-up, so that data items with the same semantics can be clustered together.



Compare Prices	**200 Cigarettes (1999)** - **DVD**, Letterbox, Checkpoint, [English] Director: Risa Bramon Garcia Starring: Christina Ricci, Kate Hudson, **Ben Affleck**, Martha Plimpton, Janeane Garofalo Also Starring: Casey Affleck, Dave Chappelle, Jay Mohr, Paul Rudd, Courtney Love, Guillermo Diaz, Angela Featherstone, Gaby Hoffmann, Brian McCardie, Nicole Parker Release Company: Paramount Home Entertainment	1999
Compare Prices	**200 Cigarettes (1999)** - **DVD**, Letterbox, Sensormatic, [English] Director: Risa Bramon Garcia Starring: Christina Ricci, Kate Hudson, **Ben Affleck**, Martha Plimpton, Janeane Garofalo Also Starring: Casey Affleck, Dave Chappelle, Jay Mohr, Paul Rudd, Courtney Love, Guillermo Diaz, Angela Featherstone, Gaby Hoffmann, Brian McCardie, Nicole Parker Release Company: Paramount Home Entertainment	1999

Figure 4. An example from bestvideobuys.com

We also note that the work in Ref. [22] uses an agglomerative clustering algorithm to group data items of the same semantics based on four visual and content features. Initially, every data item itself forms a group. The distance is calculated between two groups based on the aggregated sum of the similarities on the five features. However, this method has two major drawbacks. First, if a data record contains multiple nested data records (e.g. a plate may have multiple sizes, colours and prices), the attribute values of the nested data records may be wrongly aligned. This is because the method defines the similarities between the data items from the same data record as zero, so the data items of those nested data records with the same semantics will not be grouped together. Second, this method merges two groups with the highest similarity, but it does not consider the case where two groups have a very high similarity but different semantics.

In order to solve these problems, we introduce an agglomerative nesting algorithm (AGNES)[31,32] that takes multiple features into account. Our approach iteratively clusters data item groups with the highest similarity which is measured in terms of the data contents, presentation styles, data types, tag strings and block sizes of the two groups. In addition, we also use the positional information of data items for clustering data items with the same semantic.

4.1 Visual and content features of data items

Each data item (i.e. text node) on the Visual Block tree is associated with a set of attributes, which contains visual and content information about the data item. For example, coordinates, size, height, width, data content and so on. The data items of the same semantics, usually have a number of similar visual and content features. Our approach uses five features:

1. *Data Content*. Data items having the same semantics usually contain some similar keywords. This observation can be further investigated in two cases. First, query terms that are used to make queries in query interfaces, usually re-occur in the corresponding data items. For example, in Fig. 5, the visual blocks that represent product titles in the two data records both contain “Accent Plates”, which is a query term used to make the query. Second, some static texts are usually placed in front of data items to indicate the meaning of the data items. For example, in Fig. 5, the data items that represent the prices of the plates are prefixed by “GBP”, which indicates the currency.

2. *Presentation Style*. The data items of the same semantics are usually presented in the same style. We describe the presentation styles of a data item based on the following features: font face, font size, font colour, font weight, font style, text decoration and background colour. For example, in Fig. 5, the product titles have the same font and are in italic.

3. *Data Type*. The data type of a data item can be defined in terms of its textual values. We define six data types: image, date, time, price, number and string. Values that are not one of the first five types are defined as strings. Note that, since each type except string follows some conventional formats, e.g. a date is usually formatted as “dd–mm–yy” or “dd/mm/yy”, so it can be recognized by their values. The data items of the same semantics usually have the same data type.

4. *Tag String*. The tag string of a data item is a string of tags used to encode the data item. The data items of the same semantics usually have very similar tag

strings, since query result pages of the same web database are generated using fixed templates.

5. *Block Size.* The visual blocks that represent the data items of the same semantics have similar sizes. This is because the data records in the query result pages of the same web database tend to have a uniform and repetitive layout, and the location and size of each block is relatively fixed. For example, in Fig. 5, the blocks for prices have almost the same sizes.

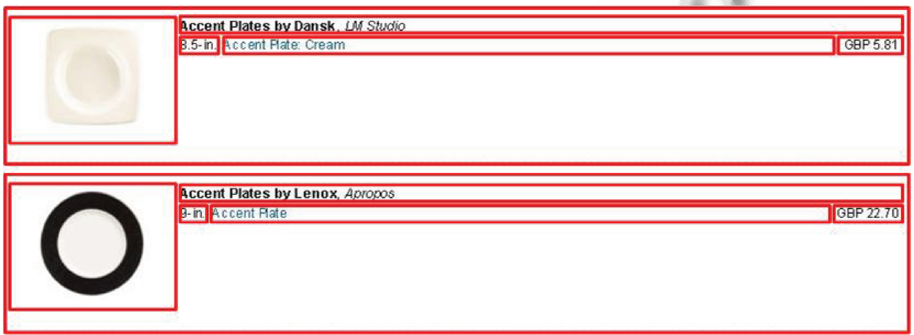

Accent Plates by Dansk, *LM Studio*
8.5-in.
Accent Plate: Cream
GBP 5.81
Accent Plates by Lenox, *Apropos*
9-in.
Accent Plate
GBP 22.70

Figure 5. Two data records in the data section with their leaf nodes visually blocked

4.2 Data item types

The data items in a data record can be divided into two types – *atomic data items* and *composite data items*. An atomic data item is a text node that contains the value of a single attribute. For example, every text node that represents the price in Fig. 5 is an atomic data item. A composite data item is a text node that contains the values of multiple attributes. For example, every product title in Fig. 5 is a composite data item, since it can be further divided into plate name ("Accent Plates by Dansk" and "Accent Plates by Lenox") and plate pattern ("LM Studio" and "Apropos") by a comma.

We observe that for a specific web database the values of a fixed set of attributes are usually encoded into a composite data item in every data record. These composite data items have the same string pattern, and there are usually some symbols in the string that visually distinguish the attributes encoded in the string, and can be used to break the composite data item into multiple atomic data items.

4.3 Align data items

To utilize the five features described in section 4.1 we define five similarity measures respectively.

1. *Data Content Similarity (simDC)*: It is defined as Cosine similarity[25] between the contents of two data items. Each data item d_i is represented by a binary term vector $TERM_{i1}, TERM_{i2}, ..., TERM_{it}$, where $TERM_{ij}$ is an unique word extracted from the texts of the two data items. A given data item collection containing two data items may then be represented as a matrix of terms where each row represents a data item, and each column represents an assignment of

a specific term to the two data items. The Cosine similarity is defined as:

$$COSINE(d_i, d_j) = \frac{\sum_{k=1}^{t}(TERM_{ik} * TERM_{jk})}{\sqrt[2]{\sum_{k=1}^{t}(TERM_{ik})^2 * \sum_{k=1}^{t}(TERM_{jk})^2}} \tag{1}$$

where $TERM_{ij}$ is assigned to one whenever term k occurs in data item i, and zero otherwise.

2. *Presentation Styles Similarity (simPS)*: It is defined as the number of common style features divided by the total number of unique style features of two data items based on the seven style features. If the two data items have the same presentation styles, *simPS* is one, otherwise it is ranging from one to zero.

3. *Data Type Similarity (simDT)*: If two data items have the same data type, then it is defined as one, otherwise zero.

4. *Tag String Similarity (simTS)*: We use edit distance EDT[30] between the tag strings (t_1 and t_2) of two data items (d_1 and d_2) to represent the number of operations (insertions, deletions and substitutions) needed to transform t_1 into t_2. The similarity between t_1 and t_2 can be defined as follows:

$$simTS(d_1, d_2) = 1 - EDT(t_1, t_2)/(LEN(t_1) + LEN(t_2)) \tag{2}$$

where $LEN(t)$ denotes the length of the tag string in tags.

5. *Block Size Similarity (simBS)*: It is defined as the ratio between the block sizes of two data items. If the two data items have the same block size, *simBS* is one, otherwise it is ranging from one to zero.

The similarity between data items d_1 and d_2 is defined as follows:

$$\begin{aligned} Sim(d_1, d_2) = \; & w_1 * simDC(d_1, d_2) + w_2 * simPS(d_1, d_2) \\ & + w_3 * simDT(d_1, d_2) + w_4 * simTS(d_1, d_2) \\ & + w_5 * simBS(d_1, d_2) \end{aligned} \tag{3}$$

where w_1, w_2, w_3, w_4 and w_5 are non-negative real numbers so that $w_1 + w_2 + w_3 + w_4 + w_5 = 1$, and $0 \leqslant Sim(d_1, d_2) \leqslant 1$.

We define the similarity between two data item groups C_1 and C_2 to be the average of similarities between each data item d_i in C_1 and each data item d_j in C_2.

$$Sim(C_1, C_2) = \frac{\sum_{d_i \in C_1} \sum_{d_j \in C_2} Sim(d_i, d_j)}{n_1 * n_2} \tag{4}$$

where n_i represents the number of data items in C_i.

An agglomerative clustering algorithm shown in Algorithm 1 is used to align data items of the same semantics. Initially each data item forms itself a data group. The intuition of the algorithm is that when two data item groups are highly similar based on the similarity measures described above, they are very likely to contain data items with the same semantic, and we cluster them into one group. We iterate this process until all data items with the same semantics are clustered into the same group. Each group corresponds to a column in the relational table. The algorithm takes as input a set of data records (denoted as R), and outputs a set of data item clusters, each of which contains data items with the same semantics. Initially, each data item in every data record forms its own group (lines 3-6). We then iteratively merge two groups with the highest similarity until no two groups have similarity greater than a threshold (denoted as T) (lines 7-15). In particular, if one group has the highest similarity with another group, we will also consult the left positions (left coordinates) of the data item(s) in the two groups (lines 11-12). If two groups have a very proximate or the same left positions, they are merged into one group, since data items of the same semantics usually have the same left position.

Algorithm 1 Clustering Data Items

```
Require: a set of data records $R$
Ensure: a set of data item clusters $C$
 1: Set $C$, a set of data items $N$, a set of data item groups $G$, all to $\{\}$
 2: Add every data item in $R$ to $N$
 3: for every data item $n_i \in N$ do
 4:     Set a data item group $g$ to $\{n_i\}$
 5:     Add $g$ to $G$
 6: repeat
 7:     Remove a data item group $g$ from $G$
 8:     repeat
 9:         if $\{g'_1, ..., g'_i, ..., g'_n\} \in G$ for $i = 1, 2, ..., n$, and $g'_i \neq g$ have the highest
            similarity $Sim(g'_i, g)$ with $g$ and $Sim(g'_i, g) > T$ and $g'_i$ is aligned with $g$
            then
10:             Set $g$ to $g \cup g'_i$
11:     until $Sim(g', g) < T$ for all $g' \in G$
12:     Add $g$ to $C$
13: until $G = \{\}$
14: Return $C$
```

We need to further identify composite data items and divide them into atomic data items. If the data items of a specific column satisfy the following three conditions, they are identified as composite data items. First, the texts of the column are all different. If the column have the same text, it is very likely that the text is a label used to annotate the data in the following column. Second, the data type of the column is string. If the data type is a non-string type, then each text of the column is itself an atomic data item. For example, if the data type of a column is "currency", then each text is an atomic data item representing "price". Third, we cannot find any separator. A separator is a non-digit, non-letter, non-space character that appears in every composite data item of the same column that visually divides the composite data item into atomic attributes. We scan all the texts of the composite data items to identify candidate separators. For each separator, we record its occurrences, and we select the separator that has the biggest occurrence as the right separator to break all the composite data items into atomic data items.

The data items in the identified data item clusters need to be aligned to construct a relational table. We observe that the majority of data items in data records are mandatory. If a data record does not have an optional data item, a blank will be filled in the position for the optional data item. If a data record has multiple nested data items, it may need multiple rows for the nested data items.

Algorithm 2 Construct relational tables

```
Require: a set of data item clusters $C$
Ensure: a relational table with all data items aligned
 1: Set tempItemSet and currentRowItemSet to $\{\}$
 2: for every cluster $c_i \in C$ for $i = 1, 2, ..., n$ do
 3:     Sort the data items in $c_i$ from top to bottom
 4: Sort the clusters in $C$ from left to right
 5: for every cluster $c \in C$ do
 6:     Add $Item_i^1 \in c$ to tempItemSet for $i = 1, 2, ..., n$
 7: repeat
 8:     Separate data items in tempItemSet into groups $\{g_1, g_2, ..., g_m\}$, based on their record numbers
 9:     if $g_i$ has the majority of data items in tempItemSet then
10:         Move data items in $g_i$ to currentRowItemSet
11:     else
12:         if the first data item in $g_i$ has the left-most position among the first data items in the remaining groups then
13:             Move data items in $g_i$ to currentRowItemSet
14:     Fill blanks into the positions left by the data items in tempItemSet for currentRowItemSet
15:     if there is a row above currentRowItemSet then
16:         for every remaining data item $i \in$ tempItemSet and $j \in$ currentRowItemSet do
17:             if $i$ matches any previous data item based on record number at its corresponding column then
18:                 Add a new row under the least matched data item and put $i$ in its corresponding column
19:             else
20:                 if $i$ matches any previous data item based on record number at that row then
21:                     Put $i$ in its corresponding column of the row
22:             if $j$ matches any previous data item based on record number at that row then
23:                 Put $j$ in its corresponding column of the row
24:     for every cluster $c \in C$ do
25:         if $Item_i^j$ exists in currentRowItemSet or tempItemSet $= \{\}$ then
26:             Add $Item_i^{j+1}$ to tempItemSet
27: until tempItemSet $= \{\}$
```

Our algorithm for constructing relational tables is shown in Algorithm 2. The algorithm takes as input a set of data item clusters, and produces a relational table as output. The clusters and the data items in each cluster are sorted first (lines

2-4) from left to right and from top to bottom respectively. Then the first data item of each cluster is put into a pool (denoted as *tempItemSet*) (lines 5-6). Next the data items in the pool are processed iteratively until the pool is empty. In every iteration, the data items in the same record are grouped together. The group contains the majority of data items in the pool is chosen as *currentRowItemSet*. The data items in *currentRowItemSet* (lines 8-10) form a row and blanks are filled into those positions for the data items that are still in *tempItemSet* (line 14). In particular, if no group has the majority, we choose one of the groups that has the left-most position for the first data item in the group (lines 12-13). The reason for doing this is that the data items that are located at the left-most position of the data records are tend to be mandatory. The algorithm then tries to align nested data items, that if a data record is a nested data record, the record number of every forth coming nested data item from its corresponding cluster will be matched with the one of data item in the previous row to see if they belong to the same data record (lines 15-23). *tempItemSet* is then replenished with the first unaligned data item from each cluster (line 24-26).

To further explain the algorithm, let's walk through an example as shown in Fig. 6. Each data item is represented by two numbers. The first indicates which cluster it belongs to, the second indicates which record it belongs to. The first cluster represents mandatory data items, the second one represents optional data items and the third represents nested data items. Initially, 1_1, 2_2 and 3_1 are put in *tempItemSet* (Fig. 6a), since 1_1 and 3_1 are from the same record and are the majority in *tempItemSet*, they are moved into *currentRowItemSet* and a blank is filled in position left by 2_2 (Fig. 6b). The *tempItemSet* is replenished with 1_2 and 3_1. As 1_2 and 2_2 are from record two and are the majority in *tempItemSet*, they are moved into *currentRowItemSet* and a blank is filled in position left by 3_1. Since there is a row ahead of *currentRowItemSet* and 3_1 matches its previous item in the same column, thus 3_1 is filled into a new row under the matched one (Fig. 6c). The rest of data items are processed in the same way. Note that when the algorithm processes 1_4 and 3_3, and chooses 1_4 as the leading data item of the row, since 1_4 is ahead of 3_3.

5 Experimental Results

We have implemented our proposed approach in a prototype using Visual C++. Each query result page is first parsed by the VIPS into a Visual Block tree which the prototype takes as input. We have conducted experiments on a data set of 200 query result pages that are returned from 20 web databases in the UIUC Web Integration Repository[26]. These web databases are from 5 domains - Books, Jobs, Movies, Music and Hotels. 15 of these pages contain a single data record. For each web database, 10 result pages are collected after manually submitting 10 different queries via its query interface.

5.1 Performance evaluations

We use two common measures, *recall* and *precision*, to evaluate the performance of our approach. For data extraction, the recall is the percentage of the number of data records that have been correctly extracted by our approach over the total number of data records on a result page. The precision is the percentage of the number of data records that have been correctly extracted over the total number of data records that have been extracted. For data alignment, the recall is the percentage of correctly aligned data items by our approach over all the data items manually aligned by human expert. The precision is the percentage of correctly aligned data items over all the data items aligned by our approach.

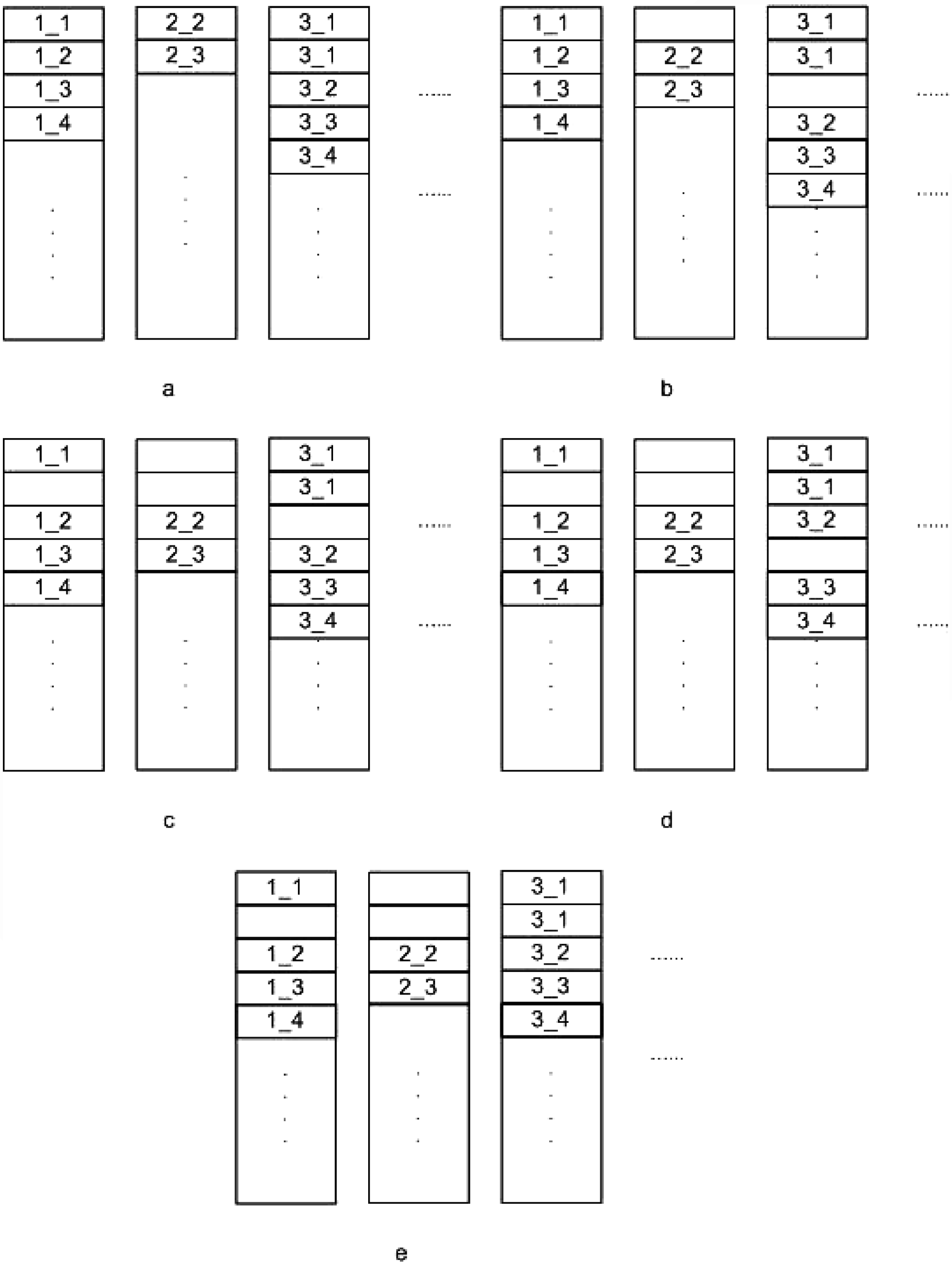


Figure 6. An example of constructing a relational table

5.2 Experimental results on data record extraction

We compare our approach with MDR[3], which is a well known data record extraction system based on HTML DOM tree and available for download online. We set the similarity threshold for MDR at its recommended value (60%). Table 1 shows the experimental results of both our approach and MDR. As we can see from Table 1, our approach has much better experimental results than MDR in total, and in almost every domain our approach significantly outperforms MDR. The precision and recall of our approach are both high across all domains, approaching 100%. Our approach can also extract query result pages with single data records, but MDR cannot. Table 1 shows that our approach has slightly higher precision than recall. The main reasons for missing data records are as follows. First, sometimes some data records do not contain any query terms so our approach cannot identify the appropriate starting leaf nodes. Second, sometimes the VIPS divides a data section into multiple sections, and our approach only identifies the largest section as the data section. The main reasons for extracting data records incorrectly are as follows. First, some noisy blocks have not been removed from the data section because they may contain query terms. Second, sometimes the VIPS parses result pages incorrectly so that some data items are missing on the Visual Block tree. Third, sometimes the VIPS fails to give correct block positions, which leads to data units missing from some data records. The performance of MDR is inversely proportional to the complexity of the result pages, and it performs relatively well on extracting data records from tables.

Table 1 Comparison results between our approach and MDR

	Our Approach		MDR	
Domain	Precision	Recall	Precision	Recall
Books	97.86%	96.76%	40.38%	82.01%
Hotel	99.20%	98.30%	18.21%	32.68%
Jobs	99.48%	98.37%	99.62%	67.60%
Movies&Music	100%	98.54%	28.05%	72.46%
Single Record Page	100%	100%	0%	0%
Total	99.26%	98.11%	38.68%	74.86%

5.3 Experimental results on data alignment

For data alignment, we evaluate the performance by comparing the data items clusters aligned by our approach with the actual data item clusters manually aligned. A correct alignment means that all data items of an attribute are clustered into one group. An incorrect alignment can be categorized into two cases. First, the data items of an attribute are put into several groups. Second, the data items of different attributes are put into the same group. When an incorrect alignment happens, we only consider the group with the most data items of some attribute as the correct alignment group for the attribute, and take the other aligned data item group(s) for the attribute and mixed in data item(s) of a different attribute as mis-aligned. Only correctly extracted data records are used as input for this step. The alignment results are shown in Table 2. "ACT" indicates the number of actual data items which reside in the Web pages; "COR" shows the number of correctly aligned data items; "WRG" means the number of incorrectly aligned data items. Our approach achieves very high accuracy in terms of both precision and recall in all domains. Especially in the hotels domain, precision and recall are approaching 100%. The results prove that the observations that we made and applied on query result pages are very robust. The incorrect alignments mainly come from the situation where the data contents of the data items with the same semantics share no common words, so that the data items clustering are only based on visual features and tag strings. The situation can be worse when some websites use the same or similar visual styles and tag strings to encode data items of different semantics. For example, "Books" and "Movie&Music" domains suffer from this problem, since the query results of books, movies and music can vary in terms of their title, starring, author, artists and so on. Also the use of agglomerative clustering algorithm can sometimes generate unsatisfied results. The reasons are twofold. First, the agglomerative clustering algorithm merges data item groups with highest similarity. Second, once two data item groups are clustered, it cannot be undone, and the new fusion will operate on the previously merged groups. It sometimes happens that the data items of an attribute clustered into several groups, but some items of these groups may have high similarities one another and these items can never be clustered again.

Table 2 Data alignment performance of our approach

Domain	ACT	COR	WRG	Recall	Precision
Books	6992	6636	356	94.91%	94.91%
Hotel	6906	6779	127	98.16%	98.16%
Jobs	6570	6295	275	95.81%	95.81%
Movies&Music	7268	6714	554	92.38%	92.38%
Total	27736	26424	1312	95.27%	95.27%

6 Related Work

Automatic extraction of web query results has attracted a lot of attention over the recent years. Several automatic extraction systems have been developed. Earlier works mainly focus on finding repetitive patterns and templates in result pages, e.g., IEPAD[14], RoadRunner[13], DeLa[15] and EXALG[16]. Recent techniques have focused on exploiting tag structures and visual features, e.g., MDR[3], DEPTA[4, 5], MSE[7], ViNTs[6], ViPER[8], ViDE[18] and [9].

The works that use visual features include ViPER[8], ViNTs[6], MSE[7] and ViDE[9]. ViDE is the most related to our approach. It is the first work that is primarily based on visual features. There are several main differences between ViDE and our approach. ViDE first clusters data items of the same semantics based on similarity between their appearances, and then groups appropriate data items from each of the clusters into data records. Our approach directly group data items that are visually adjacent to each other in the same data records. ViDE may cluster data items with different semantics because sometimes neighboring data items in the same data record may not have distinguishable appearances, resulting in them being clustered together and then grouped into different data records. Second, ViDE uses the positions and sizes of visual blocks to determine if a block is a data section. If multiple blocks are identified as candidate data sections, it chooses the one with smallest size as the data section. Our approach counts the occurrences of query terms in candidate blocks to select the real data section that makes our approach more robust.

Third, ViDE identifies noisy blocks by deciding whether the blocks are aligned to the left of a data section but it may not remove all the noisy blocks. Our approach evaluates the importance of blocks within the section based on content and visual features which improve the effect of removing noisy blocks. Fourth, ViDE assumes that data items having the same font and position belong to the same column, but visually similar data items may have different semantics. We align the data items by considering both visual and content features that makes data item alignment more accurate. ViDE treats nested-structured data (e.g. data records containing multiple sub data records) flat, so that the aligned data expand horizontally and occupy more columns unnecessarily. In contrast, our approach align the data items uniformly by clustering all the data items with the same semantics together, and will not have such problem.

Our algorithm for grouping data items of a data record is inspired by the work of Gatterbauer and Bohunsky[1, 2] on extracting web tables. Our approach instead extracts data records from query result pages that have more complex content structures. Though our approach also uses the alignment and adjacency techniques, our alignment definition is much simpler than the one in Refs. [1, 2]. Our approach also uses query terms in the process of grouping data items.

7 Conclusions

In this paper, we have presented an automatic approach for extracting data from query result pages. Our approach first uses the sizes of visual blocks and the occurrences of query terms in visual blocks to identify the data section. It then groups data items in the data section, which are adjacent to each other, into data records. It also uses content and visual features of visual blocks to evaluate their importance and to filter out noisy blocks, and align data items. Our work can be part of a web data integration system which interacts with multiple web databases, e.g. e-commerce web sites. Our experimental results show that our proposed approach is highly effective. In future work, we will develop algorithms for annotating data items in the extracted data records so that data items of the same attribute can be labeled to indicate its meaning in the database table.

References

[1] Gatterbauer W, Bohunsky P, Herzog M, Krupl B, Pollak B. Towards domain-independent information extraction from web tables. 16th International Conference on World Wide Web. ACM, New York. 2007. 71–80.

[2] Gatterbauer W, Bohunsky P. Table extraction using spatial reasoning on the CSS2 visual box model. 21st AAAI. AAAI Press. 2006. 1313–1318.

[3] Liu B, Grossman R, Zhai Y. Mining data records in web pages. 9th ACM SIGKDD. ACM, New York. 2003. 601–606.

[4] Zhai Y, Liu B. Web data extraction based on partial tree alignment. 14th International Conference on World Wide Web. ACM, New York. 2005. 76–85.

[5] Zhai Y, Liu B. Structured data extraction from the web based on partial tree alignment. IEEE Trans. on Knowledge and Data Eng., Dec. 2006, 18(12): 1614–1628.

[6] Zhao H, Meng W, Wu Z, Raghavan V, Yu C. Fully automatic wrapper generation for search engines. 14th International Conference on World Wide Web. ACM, New York. 2005. 66–75.

[7] Zhao H, Meng W, Yu C. Automatic extraction of dynamic record sections from search engine result pages. 32nd International Conference on Very Large Data Bases. VLDB Endowment.

2006. 989–1000.

[8] Simon K, Lausen G. ViPER: augmenting automatic information extraction with visual perceptions. 14th ACM International Conference on Information and Knowledge Management. ACM, New York. 2005. 381–388.

[9] Miao G, Tatemura J, Hsiung W, Sawires A, Moser LE. Extracting data records from the web using tag path clustering. 18th International Conference on World Wide Web. ACM, New York. 2009. 981–990.

[10] Liu B, Zhai Y. NET - A System for Extracting Web Data from Flat and Nested Data Records. 6th International Conference on Web Information Systems Engineering. Springer, 2005. 487–495.

[11] Zhu J, Nie Z, Wen J, Zhang B, Ma W. Simultaneous record detection and attribute labeling in web data extraction. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York. 2006. 494–503.

[12] Chang KC, He B, Li C, Patel M, Zhang Z. Structured databases on the web: observations and implications. SIGMOD Rec. ACM, 2004, 33(3): 61–70.

[13] Crescenzi V, Mecca G, Merialdo P. RoadRunner: towards automatic data extraction from large web sites. 27th International Conference on Very Large Data Bases. Morgan Kaufmann Publishers, San Francisco, CA. 2001. 109–118.

[14] Chang CH, Lui SC. IEPAD: information extraction based on pattern discovery. 10th International Conference on World Wide Web. ACM, New York. 2001. 681–688.

[15] Wang J, Lochovsky FH. Data extraction and label assignment for web databases. 12th International Conference on World Wide Web. ACM, New York. 2003. 187–196.

[16] Arasu A, Garcia-Molina H. Extracting structured data from web pages. ACM SIGMOD International Conference on Management of Data. ACM, New York. 2003. 337–348.

[17] Chang C, Kayed M, Girgis MR, Shaalan KF. A survey of web information extraction systems. IEEE Trans. on Knowledge and Data Eng., 2006, 18(10), 1411–1428.

[18] Liu W, Meng XF, Meng WY. ViDE: a vision-based approach for deep web data extraction. IEEE Trans. on Knowledge and Data Eng., 2010, 22(3), 447–460.

[19] Cai D, Yu S, Wen J, Ma W. Extracting content structure for web pages based on visual representation. 5th Asia Pacific Web Conference. Springer. 2003. 406–417.

[20] Li J, Ezeife CI. Cleaning web pages for effective web content mining. 17th International Conference on Database and Expert Systems Applications. Springer, 2006. 560–571.

[21] Wang J, Wen J, Lochovsky F, Ma W. Instance-based schema matching for web databases by domain-specific query probing. Thirtieth international Conference on Very Large Data Bases. VLDB Endowment. 2004. 408–419.

[22] Lu Y, He H, Zhao H, Meng W, Yu C. Annotating structured data of the deep web. 23rd IEEE International Conference on Data Engineering. IEEE Computer. 2007. 376–385.

[23] Song R, Liu H, Wen J, Ma W. Learning block importance models for web pages. 13th International Conference on World Wide Web. ACM, New York. 2004. 203-211.

[24] Debnath S, Mitra P, Pal N, Giles CL. Automatic Identification of Informative Sections of Web Pages. IEEE Trans. on Knowledge and Data Eng., 2005, 17(9): 1233–1246.

[25] Salton G, McGill MJ. 1986 Introduction to Modern Information Retrieval. McGraw-Hill, Inc.

[26] The UIUC Web Integration Repository, `http://metaquerier.cs.uiuc.edu/repository/`.

[27] Madhavan J, Jeffery S, Cohen S, Dong X, Ko D, Yu C, Halevy A. Google Inc: Web-scale Data Integration: You Can Only Afford to Pay As You Go. Proc. of CIDR-07. 2007.

[28] Weng D, Hong J, Bell D. Extracting data records from query result pages based on visual features. 28th British National Conference on Databases. Springer, 2011.

[29] Jurafsky D, Martin J. Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. Prentice Hall PTR. 2000.

[30] Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, 1997.

[31] Han J. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers Inc. 2005.

[32] Kaufman L, Rousseeuw P. Finding Groups in Data: An Introduction to Cluster Analysis. 1990.