Visually Extracting Data Records from the Deep Web

Neil Anderson
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
nanderson423@qub.ac.uk

Jun Hong
School of Electronics, Electrical Engineering and Computer Science
Queen's University Belfast
Belfast BT7 1NN, UK
j.hong@qub.ac.uk

ABSTRACT

Web sites that rely on databases for their content are now ubiquitous. Query result pages are dynamically generated from these databases in response to user-submitted queries. Automatically extracting structured data from query result pages is a challenging problem, as the structure of the data is not explicitly represented. While humans have shown good intuition in visually understanding data records on a query result page as displayed by a web browser, no existing approach to data record extraction has made full use of this intuition. We propose a novel approach, in which we make use of the common sources of evidence that humans use to understand data records on a displayed query result page. These include structural regularity, and visual and content similarity between data records displayed on a query result page. Based on these observations we propose new techniques that can identify each data record individually, while ignoring noise items, such as navigation bars and adverts. We have implemented these techniques in a software prototype, *rExtractor*, and tested it using two datasets. Our experimental results show that our approach achieves significantly higher accuracy than previous approaches. Furthermore, it establishes the case for use of vision-based algorithms in the context of data extraction from web sites.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Clustering, Information filtering*

Keywords

Data record extraction, vision-based data extraction, deep web data integration

1. INTRODUCTION AND MOTIVATION

Web databases are now pervasive. Users retrieve information from these databases by submitting HTML query forms. Query results are displayed on a web page, but in a proprietary presentation format, dictated by the web site designer. We call these pages, *query result pages*. Figure 1 shows a typical query result page from Argos.co.uk. On this page each DVD is presented as a data record.

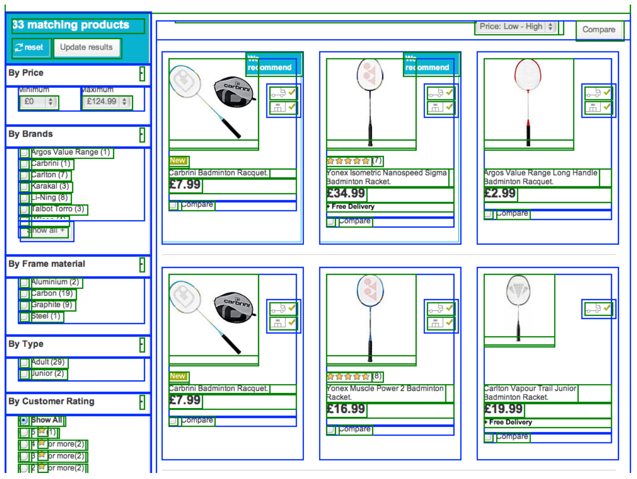
33 matching products
reset
Update results
By Price
By Brands
By Frame material
By Type
By Customer Rating
Price: Low - High
Compare
Carbrini Badminton Racquet.
£7.99
Yonex Isometric Nanospeed Sigma Badminton Racket.
£34.99
Argos Value Range Long Handle Badminton Racquet.
£2.99
Yonex Muscle Power 2 Badminton Racket.
£16.99
Carlton Vapour Trail Junior Badminton Racket.
£19.99

Figure 1: Query Result Page from Argos.co.uk

In turn, each data record contains a collection of data items; including the title, description, release date, availability, price and customer rating, etc. as well as an image of the product. Any item on the page that does not belong to a data record is called a *noise item*, for example, a menu button or an advert. A query result page is designed for a human to read rather than a computer to process, thus there is no standard way to automatically extract structured data from the page.

Automatic data extraction is the process of extracting automatically a set of data records and the data items that they contain from a query result page. Such structured data can then be integrated with data from other data sources and presented to the user in a single cohesive view in response to their query. For instance, there is great commercial demand for comparison shopping search engines. A user may wish to buy a DVD; a comparison shopping search engine can extract data from many different online stores, integrate the data and display it to the user. Other practical applications include flight and hotel booking sites, financial product comparisons, property sales and rentals.

In this paper, we focus on the problem of data record extraction, that is, to identify the groups of data items that make up each data record on a query result page.

Data records on a query result page display regularity in their content, structure and appearance. They exhibit structural and visual similarities: that is, they form visual patterns which are repeated on the page. This is because data records on the same site are often presented using the same template. The displayed data is in an underlying database, and as the data items for each record are

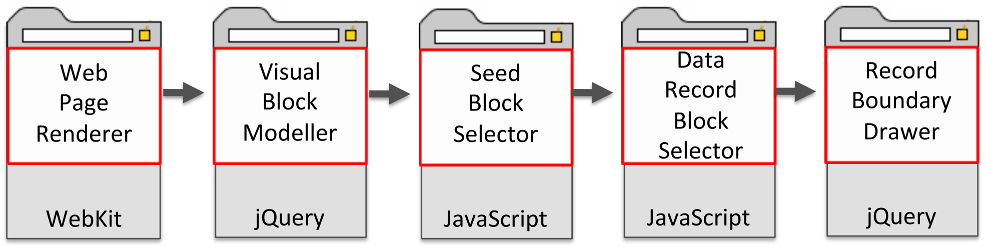

Figure 2: *rExtractor* System Architecture

retrieved from the database (in response to the user's query), the same template is used each time to present the record.

Much of the existing work that deals with the extraction of data records is based on the theme of identifying repeated patterns. The common premise is to find and use the repeated patterns of data records. The main differences between the existing approaches are where they look for these patterns and how they use them in data extraction.

Early approaches [2, 5] identify repeated patterns in the HTML source code of multiple training pages from a data source in order to infer a common structure or template, and use it to extract structured data from new pages from the same source. Other approaches [4, 17] identify repeated patterns in the source code of a query result page in order to directly extract data records and align data items in them. However, these approaches are all limited by the rapidly increasing complexity of source code. For instance, the widespread use of Javascript libraries, such as jQuery, can make source code structurally much more complex; thus these approaches start to fail.

Later approaches [11, 18, 20, 21] use the tag tree representation of a web page to identify repeated patterns in the subtrees of the tag tree. This representation is useful because it provides a hierarchal representation of the source code. However, the tag tree was designed for the browser to use when displaying the page, and unfortunately does not accurately resemble the structure of the data records on the displayed page. The use of scripts and other runtime features contribute further to the differences between the structure of the tag tree and the dynamically displayed web page.

The state-of-the-art approaches [13, 16, 19] identify visually-repeated patterns using additional information displayed on a rendered page. However, they all succumb to the same limitation: they still rely on direct access to either the source code or the tag tree.

The approach in [12] is the first that uses primarily visual features for data record extraction. However, this approach has several other limitations caused by the granularity of its input data and how it deals with noise items on a web page.

In addition, many of the current approaches [12, 16, 18, 19] start by identifying a section of the page, referred to as the *data-rich section*, which contains all of the data records. However, correctly identifying the data-rich section can be problematic. There is a risk that some data items may be omitted from, or unwanted noise items incorrectly included in, the data-rich section. Furthermore, it is possible to identify the wrong sub-section of the page entirely.

In this paper, we propose a novel visual approach to data record extraction, which shows a strong correlation with human intuition. Our approach is guided by how most humans expect query result pages to be visually presented. This also influences web site designers, who often present pages to meet human expectations.

A common convention is to place noise items, such as menus or adverts, around the periphery of the page, reserving the centre of the page for the data records. Furthermore, data items in each data record are displayed together, forming what is known as a locality constraint. As shown in Figure 1, query result pages follow these conventions.

To make it easy for a human reader to perceive data items in each data record as a group, designers display the data items repeatedly in the same relative position in each data record. Furthermore, designers also reserve the same horizontal space (or width) for each data record. Originally they did so because human readers disliked pages that required horizontal scrolling [9], therefore, the data records on the page must have a uniform width so they would not cause the page to exceed the screen width. Indeed this accepted design constraint contributes greatly to the regular visual appearance of data records. Technical constraints also have a part to play. Each data record on the page is displayed using the same template code, which, when rendered, reinforces a repeated visual appearance and therefore a similar composition for each data record.

A human reader uses the regularity of the visual-repeated pattern and structure that appears when the data records are displayed on the page to infer semantic relationships between data items and can therefore group the data items into data records.

In summary, our main contributions in this paper are as follows: First, we remove the requirement to identify the data-rich section of a query result page. We propose a novel visual approach which identifies, as a seed block, a single data item, which is a basic content block in a data record. We then implement our observations on visual and structural regularity to group together only the data items in each record. Second, our visual approach directly accesses a rendering engine to retrieve positional information and visual features of each item on the page, avoiding the need to interpret increasingly complex HTML source code and tag trees.

The rest of the paper is organised as follows. The fundamentals behind our proposed vision-based approach are presented in Section 2. Solutions for identifying a record block for each data record on the query result page are described in Section 3 and Section 4. A summary of our demonstration system is given in Section 5. Experimental results are reported in Section 6 and finally, Section 7 concludes the paper.

2. VISUAL DATA RECORD EXTRACTION

In this section, we first introduce the *visual block model* (VBM) to represent a rendered web page, which allows us to access the positional information and visual properties of each item on the page. Next, we present our method to measure the visual similarity between the visual blocks in the model, followed by our definitions of the spatial relationships between these visual blocks. Finally, we present an overview of our approach.

2.1 Visual Block Model

The visual block model of a query result page is a product of the tag tree and the Cascading Style Sheet (CSS) of the page. A layout engine generates visual blocks for each node in the tag tree, according to the instructions contained in the CSS. This process, called rendering, draws a rectangular box around the minimum boundary of each visible node on the page. We refer to each box as a *visual block*. The position of each visual block is represented by its four borders in the four directions on the two-dimensional plane. The plane has its origin at the top-left of the page, with the x-axis running from left to right and y-axis running from top to bottom.

Some of the visual blocks in the VBM, such as those outlined in blue in Figure 1, represent the structural components of the query result page. These blocks, which we call *container blocks*, can be thought of as the scaffolding that holds the structure of the page together. Each of these visual blocks contains at least one other visual block, each of which we call a *child block*. The larger visual blocks shown in Figure 1, such as those that appear to surround each data record, contain many other blocks.

The remaining visual blocks in the VBM, such as those shown in green in Figure 1, which we call *basic blocks*, represent the individual labels and data items displayed on the result page. These blocks do not contain any other visual blocks. A container block for a data record typically contains a number of these basic blocks.

For each visual block, we obtain 160 visual properties that can be used to define the visual appearance of the block. We choose the properties that have historically enjoyed good cross-browser representation and are the most widely used by web site designers to define the visual appearance of query result pages, for example, *fontWeight*. Our approach uses these properties to determine the visual appearance of each block and is flexible; additional visual properties from the CSS specification can be incorporated easily. We use the WebKit layout engine [10] to render query result pages, however, our approach is independent of any specific engine.

In contrast to the VIPS Algorithm [3], used in other visual approaches, our VBM makes no prior assumptions regarding the organisational hierarchy of a query result page, it simply provides the visual properties for each displayed item. Furthermore, our VBM has finer granularity of visual blocks than the VIPS algorithm. For instance, it can represent two consecutive visual blocks containing two visually distinct texts whereas the VIPS cannot segment the two texts into two visual blocks. As shown in Figure 2, the Visual Block Modeller is a key component of our system architecture.

2.1.1 VBM vs. Tag Tree

While the VBM and the tag tree are related, they are not equivalent. The tag tree is a complex representation of the HTML code of the page, created for the browser to interpret and is only part of the information required to render the page as the designer intended. We choose to use the VBM in preference to the tag tree for data record extraction for a number of reasons. First, nodes that are close together on the tag tree may be spatially far apart on the displayed page and vice-versa. By considering only the visual blocks in the VBM, our approach can 'see' the result page in the same way that a human can. Crucially, this is how the page was designed: inferred relationships, such as a group of data items that form a data record, are much easier to identify in a visual context. Second, our approach is insulated from developments in coding practices and standards. Our approach relies on the rendering engine, accordingly we are at liberty to make use of the best engine without the need to adapt our VBM.

2.2 Spatial Relationship between Visual Blocks

We now define a spatial relationship between the visual blocks. The structural regularity of data records on a query result page can be recognised by identifying this spatial relationship of data items in data records.

DEFINITION 1. ***Contains:*** *A visual block contains another if all of the borders of the later are inside those of the former.*

For example, as shown in Figure 3, block F contains blocks A, B, C, D and E.

2.3 Similarity between Visual Blocks

Our approach decides if two visual blocks have visual, width or content similarity.

DEFINITION 2. ***Visual Similarity:*** *Two visual blocks, A and B, are visually similar if all of the visual properties of both blocks are the same. Let $P_A = \{P_{a1}, P_{a2}, ..., P_{an}\}$ be a set of visual properties of A, and $P_B = \{P_{b1}, P_{b2}, ..., P_{bn}\}$ be a set of visual properties of B. The visual similarity between A and B, $Sim(A, B)$, is defined as follows:*

$$Sim(A, B) = \begin{cases} 1 & if P_{ai} = P_{bi}, i = 1, 2, ..., n; \\ 0 & Otherwise. \end{cases}$$

For example, as shown in green in Figure 1, the visual blocks that contain the product name in each record on the query result page have the same visual proprieties and are, therefore, visually similar.

DEFINITION 3. ***Width Similarity:*** *Two visual blocks have similar widths if the width properties for both blocks are within a threshold of 5 pixels of each other.*

For example, as shown in blue in Figure 1, the visual block that contains all of the data items of the first record, is the same width as the visual blocks that contain all of the data items for the remaining records. We say these blocks have similar widths.

Our approach also needs to decide if two blocks have block content similarity, that is, they have similar sets of child blocks. Our observation is that two record blocks have a high degree of block content similarity. This is because both record blocks are created from the same template, repeated for each record on the page. For example, the two record blocks, as shown in Figure 1, contain a large number of visually similar blocks. In contrast, our observation is that there is little block content similarity between a record block and a noise block. For example, the container block for the 'By Price' option in the left sidebar, which is the same width as, the record blocks in Figure 1, does not share any visually similar child blocks with the record blocks.

We use a variant of the Jaccard index [15] to measure block content similarity between two visual container blocks. The index ranges between 0 and 1, where 1 means that the two blocks are identical, and 0 means they have nothing in common. In our approach, we consider that each container block contains a set of child blocks. We can then measure block content similarity between two container blocks by a similarity index between two corresponding sets of visual child blocks.

DEFINITION 4. ***Block Content Similarity:*** *Two visual container blocks have similar block contents if they have a similarity index above a preset threshold.*

For example, as shown in blue in Figure 1, the visual block that contains all of the data items of the first record has a large number of child blocks that have the same visual appearance as those child blocks in the visual blocks that contains all of the data items of the remaining record. We say these blocks have block content similarity. In Section 4.2, we formalise our usage of the similarity index.

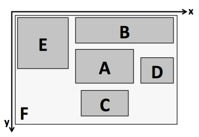

Figure 3: Examples of Spatially Related Visual Blocks.

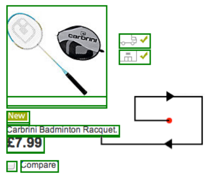

Figure 4: An Ulam Spiral Encountering a Basic Block.

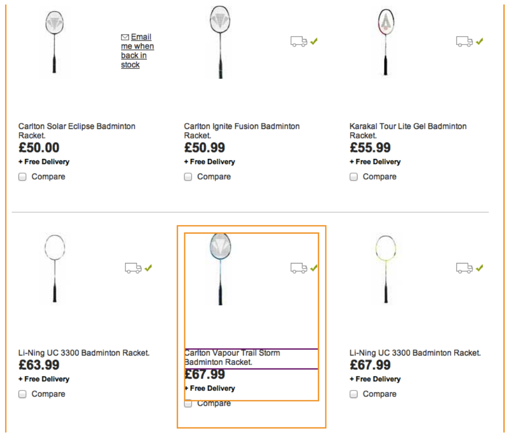
Email me when back in stock
Carlton Solar Eclipse Badminton Racket.
£50.00
+ Free Delivery
Compare
Carlton Ignite Fusion Badminton Racket.
£50.99
+ Free Delivery
Compare
Karakal Tour Lite Gel Badminton Racket.
£55.99
+ Free Delivery
Compare
Li-Ning UC 3300 Badminton Racket.
£63.99
+ Free Delivery
Compare
Carlton Vapour Trail Storm Badminton Racket.
£67.99
+ Free Delivery
Compare
Li-Ning UC 3300 Badminton Racket.
£67.99
+ Free Delivery
Compare

Figure 5: Seed Block and Candidate Record Blocks Highlighted

2.4 Data Record Extraction

Each data record on the page is represented by a visual block, which contains all of the contents of the data record and nothing else. We have completed a survey of 600 query result pages from our data sets and found that in over 98% of cases there is a single visual block that exactly contains each data record. The goal of our approach is to identify this block for each data record on a query result page. We call such visual blocks, *record blocks*.

Our approach starts by identifying a single basic visual block that is very likely to be one of the basic blocks of a data record, we call this the *seed block*. The seed block is contained in a set of larger blocks, which we call the *candidate record blocks*, as only one of them is the record block for the data record that the seed block is in. Our approach must select the correct candidate record block as the record block.

3. SEED BLOCK SELECTION

The goal of seed block selection is to identify a single basic visual block from the VBM which is part of a single data record. Let us look at the organisation of a query result page. Since the Western reading order is from top to bottom, and from left to right, it follows that the data records should start in the top left of the page. However, most web pages have common navigation menu and header structures that appear around the edges of the page. Thus, the starting point of the data records can be shifted down and to the right. As human readers expect this convention, they start looking for data records in this area of the page. A study on web usability by eye tracking [14] confirms that the highest priority area for content is between the centre and the top left of the page.

Our approach starts at the centre of the page, furthest from the noise blocks at the edges and closest to the highest priority area for data records. We trace a clockwise Ulam Spiral [6], as shown

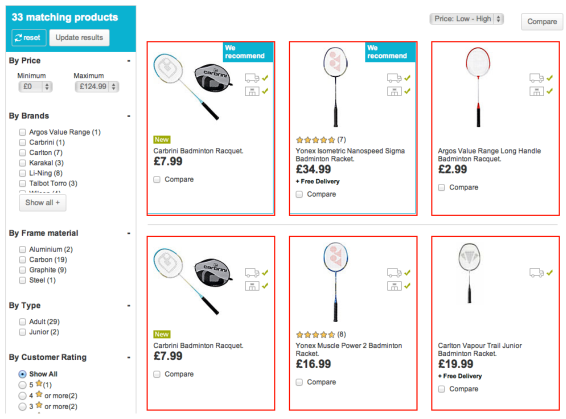

Figure 6: Data Record Boundaries Highlighted

in Figure 4, which naturally grows from the centre towards the top left of the page. This is the area of the page most likely to contain data records.

The Ulam Spiral was specially selected as it covers the largest possible proportion of the highest priority area, before it reaches the edges of the page. A simple plane between the centre of the page and the top left corner of the page has, on the other hand, the potential to miss the basic blocks belonging to sparsely populated data records. Instead it could quickly reach the edge of the page and select as the seed a basic block belonging to a noisy feature such as a left menu.

The exponential growth of the spiral combined with its direction of travel (clockwise) ensures that it shows bias to the area between the centre and the top left of the page thereby covering more of the highest priority area than is possible for a simple plane cover. As shown in Figure 4, the spiral terminates when it first encounters a basic block. This block is taken as the seed block. For our running example, the seed block is shown in Figure 5, highlighted in purple.

4. DATA RECORD SELECTION

The goal of data record selection is to identify a set of container blocks from the VBM, one block for each of the data records on the query result page.

The seed block is contained inside a number of container blocks, each of which provides a structure on the page. Examples of these container blocks are shown in Figure 1, highlighted in blue. By isolating only the container blocks in which the seed block is actually contained, our approach identifies the set of candidate record blocks, as shown in Figure 5, highlighted in orange. We observe that one of these candidate record blocks is the record block for the data record and furthermore, this visual block has the similar width to the record block for each of the records on the same query result page.

4.1 Getting Candidate Record Blocks and Clustering Container Blocks

The seed block is a basic block, which is contained inside one or more of the container blocks. As shown in Figure 5, the seed block, which was selected in the previous step, is contained inside a number of container blocks, highlighted in orange. As one of these blocks is the record block, all four are taken as the candidate record blocks. Next, our approach filters the set of all container blocks on the page, discarding any block that is not the same width as one of the candidate record blocks. Our approach uses a one-pass algorithm to cluster the filtered container blocks into a strict partition based on block width. This step creates a number of clusters, one of which contains the record blocks. In our example, the algorithm would create four clusters, one for each of the candidate record blocks.

4.2 Measuring Block Content Similarity

Our approach uses a similarity measure to determine if two container blocks have similar block contents. Assume that block A contains a set of child blocks $\{a_1, a_2, ..., a_m\}$ and block B contains a set of child blocks $\{b_1, b_2, ..., b_n\}$.

It is reasonable to expect that a container block may contain more than one child block with the same visual properties. For example, a data record on a car sales web site may contain an individual child block for each feature of the car, such as the engine size, number of doors and fuel type. These child blocks could share the same visual properties. Accordingly, our approach uses a multi-set representation for each container block. This generalisation of the notion of a set, in which members may appear more than once, allows our approach to represent the child blocks of a container block.

Our approach uses a one-pass algorithm to cluster each of the child blocks contained in a set into a strict partition based on their visual similarity. The function Sim, defined in Definition 1, is used for this purpose. Two child blocks, a and a', are clustered together if $Sim(a, a') = 1$. So a set of child blocks, A, is clustered into a strict partition $A_c = \{A_1, A_2, ..., A_m\}$.

For instance, assume that container block A contains four child blocks, a_1, a_2, a_3 and a_4. Two child blocks, a_1 and a_2, are visually similar, while a_3 and a_4 are visually distinct. The corresponding multi-set, A, is clustered into a set of subsets, A_c, where each subset in A_c represents a single cluster:

$$A_c = \{\{a_1, a_2\}, \{a_3\}, \{a_4\}\} \tag{1}$$

Select one child block from each cluster in A_c as its representative, so we have a set of representative child blocks, A_x, for A_c:

$$A_x = \{x_1, x_2, ..., x_m\} \tag{2}$$

Given two sets of visual blocks, A and B, we use our similarity measure to find the block content similarity between A and B, defined as follows:

$$SimBlockContent(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{3}$$

We define an indicator function for A_x as follows:

$$1_{A_x}(x_i) = |A_i| \; x_i \in A_x \wedge A_i \in A_c, for \; i = 1, 2, ..., m \tag{4}$$

Assume that B is clustered into B_c and $B_y = \{y_1, y_2, ..., y_n\}$ is a set of representative child blocks for B_c. We define an indicator function for B_y as follows:

$$1_{B_y}(y_j) = |B_j| \; y_j \in B_y \wedge B_j \in B_c, for \; j = 1, 2, ..., n \tag{5}$$

We then have:

$$|A \cap B| = \sum_{i=1}^{m}\sum_{j=1}^{n} min\{1_A(x_i), 1_B(y_j)\}|Sim(x_i, y_j) \tag{6}$$

$$\begin{aligned} |A \cup B| &= \sum_{i=1}^{m}\sum_{j=1}^{n} max\{1_A(x_i), 1_B(y_j)\}|Sim(x_i, y_j) \\ &= 1 + |A - B| + |B - A| \end{aligned} \tag{7}$$

where

$$|A - B| = \sum_{i=1}^{m}\sum_{j=1}^{n} 1_A(x_i)|Sim(x_i, y_j) = 0$$

and

$$|B - A| = \sum_{i=1}^{m}\sum_{j=1}^{n} 1_B(y_j)|Sim(x_i, y_j) = 0$$

If $SimBlockContent(A, B)$ is above a preset threshold, A and B are considered to have similar block contents.

4.3 Selecting Record Blocks

Only one of the candidate record blocks represents the container blocks that provide the structure for each data record on the page. The other candidate record blocks represent container blocks that are used to provide structure to other areas of the page. By selecting the candidate record block, which has content similarity to the maximum number of container blocks, our approach identifies the blocks for each data record.

5. DEMONSTRATION

rExtractor [1] is implemented as a Java application, built in Eclipse, with an integrated WebKit browser that allows us to visualise the results of our data record extraction algorithms. In the demonstration we will exhibit the five modes of operation of *rExtractor*, each of which has been designed to demonstrate a different aspect of our vision-based approach. Each mode can be independently selected from the Java application. They are:

- **Render Only Mode:** *rExtractor* implements the WebKit browser engine to render an input query result page. No visual blocks are highlighted in this mode.

- **Debug Mode:** *rExtractor* highlights each visual block on the page by overlaying a colour-coded line onto the boundary of the visual block. For example, the output from this mode is displayed similarly to the example shown in Figure 1.

- **Seed Block Mode:** *rExtractor* highlights only the seed block on the query result page. The seed block is highlighted in purple, this is displayed similarly to the example seed block shown in Figure 3. As the algorithm implemented a Ulam Spiral to locate the seed block, the path of the Spiral along with its intersection with the seed block is also displayed.

- **Candidate Data Record Block Mode:** *rExtractor* highlights the candidate data record blocks that contain the seed block. The candidate data record blocks are highlighted in orange, and are displayed similarly to the example candidate data record blocks shown in Figure 5.

- **Data Record Block Mode:** *rExtractor* highlights the boundaries of data record blocks for each data record on the query result page. The boundaries of data record blocks are highlighted in red, and are displayed similarly to the example shown in Figure 6. The demonstration also includes an option to visualise the block contents of each data record block, the boundaries of these visual blocks are highlighted in light orange.

A sample of the *rExtractor* functionality is available as a screencast online at: `http://goo.gl/807st`

In the screencast, we use the *rExtractor* to extract the data records from a number of example query result pages. These include modern query result pages in which the data records are arranged in 1) a grid structure and 2) a single column structure. We show that *rExtractor* is able to extract data records from pages containing either arrangement of data records. For each query result page, we demonstrate each mode of operation available in *rExtractor* in sequence.

6. EVALUATION

We compared the performance of *rExtractor* with that of ViNTs [19], a state of the art extraction system available on the web, which is based on both visual content features and HTML tag structures.

In our experiments, we used two data sets, *DS1* and *DS2*, derived from third party sources [7, 19]. DS1 contains 752 data records and was used to compare the performance of *rExtractor* with that of ViNTs. DS2 contains a further 11,458 data records and was used to evaluate the performance of *rExtractor* on a large dataset. In total our data sets contained 12,210 data records from the domains of Books, Music, Movies, Shopping, Properties, Jobs, Automobiles and Hotels.

Table 1: Results for DS1 and DS2

DataSet	Algorithm	Avg. Precision	Avg. Recall
1	*rExtractor*	95.5%	96.3%
1	ViNTs	46.3%	48.9%
2	*rExtractor*	97.0%	95.5%

On DS1, *rExtractor* achieved an average Recall of 96.3% and an average Precision of 95.5%. This is compared to ViNTs which achieved an average Recall of 48.9% and an average Precision of 46.3% across the same dataset. Detailed analysis of the results for ViNTs found that they identified a large number of false positive data records (427 in total). For example, they often selected a page navigation menu as the data-rich section, and then extracted each menu item as a false positive data record. In these cases, it was then impossible for ViNT to extract the correct data records. Consequently both their Recall and Precision values were negatively impacted.

Conversely, the seed block selection technique implemented in *rExtractor* selected correctly a basic visual block contained in a data record block in all of the test cases and *rExtractor* identified very few false positive data records. This demonstrates that our approach works effectively without having to explicitly identify a data-rich section. *rExtractor* also preformed very well on DS2, achieving an average Recall of 96.3% and an average Precision of 95.5%.

7. CONCLUSIONS & FUTURE WORK

This paper presents a novel approach to the automatic extraction of data records from query result pages. Our approach first identifies a single visual seed block in a data record, and then discovers the candidate record blocks that contain the seed. Next the container blocks on the page are clustered and a similarity measure is used to identify which of these blocks have similar contents to each candidate record block. Finally, our approach selects the cluster of visual blocks that correspond to the data records.

In future work, we plan to extend our approach so that 1) the similarity threshold, used to determine if two container blocks have similar block contents is set automatically by a machine learning technique and 2) the data items contained in each data record are semantically labelled using the vocabulary provided by Schema.org[8].

8. REFERENCES

[1] N. Anderson and J. Hong. Visually extracting data records from query result pages. In *APWeb Conference*, 2013.

[2] A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD Conference*, pages 337–348, New York, NY, USA, 2003.

[3] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma. Extracting content structure for web pages based on visual representation. In *APWeb Conference*, pages 406–417, 2003.

[4] C.-H. Chang and S.-C. Lui. Iepad: information extraction based on pattern discovery. In *WWW Conference*, pages 681–688, New York, NY, USA, 2001.

[5] V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB Conference*, pages 109–118, San Francisco, CA, USA, 2001.

[6] http://mathworld.wolfram.com/PrimeSpiral.html. Prime spiral, 2012.

[7] http://metaquerier.cs.uiuc.edu/repository/datasets/tel 8/. Tel-8 query interfaces, 2004.

[8] http://schema.org/. Schema.org, 2013.

[9] http://www.useit.com/alertbox/20021223.html. Jakob nielsen - usable i.t., 2002.

[10] http://www.webkit.org/. Webkit - layout engine.

[11] B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *SIGKDD conference*, pages 601–606, New York, NY, USA, 2003.

[12] W. Liu, X. Meng, and W. Meng. Vide: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering*, 22:447–460, 2010.

[13] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser. Extracting data records from the web using tag path clustering. In *WWW Conference*, pages 981–990, 2008.

[14] J. Nielsen and K. Pernice. *Eyetracking Web Usability*, pages 97–110. New Riders, first edition, 2010.

[15] R. Real and J. M. Vargas. The probabilistic basis of jaccard's index of similarity. *Systematic Biology*, 45:380–385, 1996.

[16] K. Simon and G. Lausen. Viper: augmenting automatic information extraction with visual perceptions. In *CIKM Conference*, pages 381–388, New York, NY, USA, 2005.

[17] J. Wang and F. H. Lochovsky. Data extraction and label assignment for web databases. In *WWW Conference*, pages 187–196, New York, NY, USA, 2003.

[18] Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW Conference*, pages 76–85, New York, NY, USA, 2005.

[19] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *WWW Conference*, pages 66–75, New York, NY, USA, 2005.

[20] H. Zhao, W. Meng, and C. Yu. Automatic extraction of dynamic record sections from search engine result pages. In *VLDB Conference*, pages 989–1000, 2006.

[21] H. Zhao, W. Meng, and C. Yu. Mining templates from search result records of search engines. In *SIGKDD Conference*, pages 884–893, New York, NY, USA, 2007.