A Rendering-based Method for Selecting the Main Data Region in Web Pages

Leandro Neiva Lopes Figueiredo
Departamento de Computação
Universidade Federal de Ouro Preto
Ouro Preto, MG, Brazil
leomcl@gmail.com

Anderson Almeida Ferreira
Departamento de Computação
Universidade Federal de Ouro Preto
Ouro Preto, MG, Brazil
ferreira@iceb.ufop.br

Guilherme Tavares de Assis
Departamento de Computação
Universidade Federal de Ouro Preto
Ouro Preto, MG, Brazil
gtassis@iceb.ufop.br

***Abstract*—Extracting data from web pages is an important task for several applications, such as comparison shopping and data mining. Much of that data is provided by search result pages, in which each result, called search result record, represents a record from a database. One of the most important steps for extracting such records is identifying, among different data regions from a page, one that contains the records to be extracted. An incorrect identification of this region may lead to an incorrect extraction of the search result records. In this paper, we propose a simple but efficient method that generates path expression to select the main data region from a given page, based on the rendering area information of its elements. The generated path expression may be used by wrappers for extracting the search result records and its data units, reducing its complexity and increasing its accuracy. Experimental results using web pages from several domains show that the method is highly effective.**

***Keywords*-rendering information; visual information; wrapper; main data region; path expression**

I. Introduction

Most of the data available in web pages comes from records stored in databases. In this context, if a website produces several web pages, such pages usually use the same template and are generated by the same script. Generally, such pages are accessed from a search in a website.

These pages, known as search result pages, may contain many data regions, i.e., groups of data in different locations on the screen, for instance, the region that contains the menu, ads, or search result records (SRRs). Each SRR represents an object from a database related with a user search and may contain many attributes. For example, in Fig. 1, the rectangles (a), (b) and (c) are data regions and (d) to (k) are SRRs. Each SRR has attributes, e.g., name, price and user reviews.

Several applications, such as comparison shopping, digital libraries, and data mining and integration, need data from these web pages. Thus, pattern discovery and extraction of data from such pages has required much attention from the scholarly community. Among the steps for extracting data from a search result page, one of the most important is identifying its main data region, i.e., the data region with the SRRs. For instance, on Fig. 1, rectangle (c) is the page's main data region. An incorrect identification may lead to the extraction of irrelevant data (i.e., noises) and, thus, to a decrease on the accuracy of the wrapper.

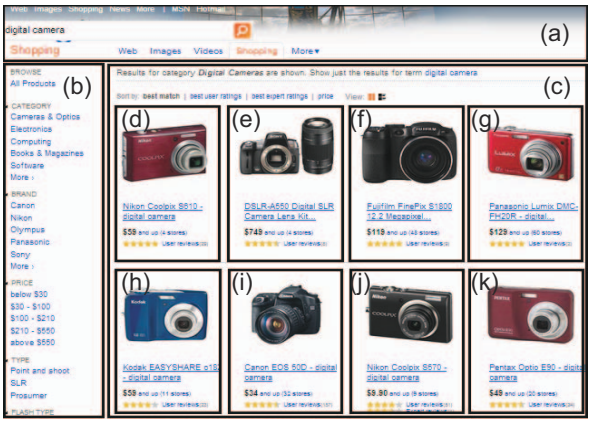

Figure 1. Example of data regions ((a),(b) and (c)), SRRs ((d) to (k)) and data units (e.g., product name, price, user reviews)

In this work, we propose a completely automatic method for identifying the main data region from a HTML web page, analyzing its DOM (Document Object Model) tree and the rendering information of its elements. As a result, the method generates a path expression (i.e., XPath expression) that may be used by algorithms that extract SRRs and its corresponding data units (i.e., attributes).

In web pages, the layout characteristics are known as *rendering area information* and are available when the browser loads the web page (e.g., X and Y coordinates, height and width of each element on the screen). Thus, this information cannot be obtained by analyzing solely the structure of the page.

Like [1] and [2], we assume that the intention of the arrangement of elements on the screen is to emphasize the information the user is looking for, i.e., the region with the SRRs. Thus, this region fills a large area on the screen. This spatial information is obtained when we combine the rendering information (e.g., height and width) of each element. Based on our assumption and the area permeated by the elements, we may decrease the runtime to process

978-1-4799-6953-1/14 $31.00 © 2014 IEEE
DOI 10.1109/LAWeb.2014.14

the DOM tree, focusing only on the most promising regions to contain the SRRs.

Thus, in summary, the main contributions of our work are: (1) completely automatic selection of the main data region, i.e., no human intervention is needed; (2) the method works on a single page, not requiring many page examples to process; (3) it is domain independent, i.e., its rules are not based on specific domains; and (4) the produced path expression may be used by wrappers to extract SRRs and data units.

The rest of this paper is organized as follows: the next section reviews previous related work. Section III describes our method. Section IV reports our experimental evaluation and discusses the results and Section V concludes the paper and discusses future work.

II. Related Work

Information extraction from web pages has been studied in several previous works. Some of them analyze the structure of the pages as a tree, representing hierarchically their elements ([3], [4], [5], [6], [7], [8]). Others use machine learning techniques or probabilistic models to train the algorithm with labeled examples and then induce wrappers to extract data ([9], [10], [11], [12]). Other ones use examples provided by users in a GUI (Graphic User Interface) to find and extract data by analyzing similar objects ([13], [14]) and others use ontologies in the extraction task [15].

The use of structural analysis combined with visual information is found in some works. The VIPS method, proposed in [16], uses the position of the elements in the DOM tree and their visual information (e.g., font name, font color) in order to group similar objects from several segments. Next, it assigns a score to each segment and uses this score to improve search engines, which may use VIPS to detect navigation, decoration, interaction and contact information, and exclude those from the results.

In [17], the authors propose a method that uses positional information of the elements in the DOM along with their visual characteristics (e.g., font style, visibility and rendered area) to find the SRRs and generate candidate XPath expressions to be used in the extraction task. These candidates are ranked by the similarity of their resulting nodes.

Another method, called ViNTs[18], initially uses the visual content (without HTML elements) of the page to identify the regularities from content itself (e.g., text, images, anchors), and then combines those with the HTML tag structure to generate wrappers. The method requires at least five search result pages (from the same web site) with at least four SRRs to identify the regularity among the SRRs that is explored in the wrapper construction. It also requires a special result page called no-result page, with no SRRs, as a negative example.

In [1], the authors represent each page's element as a path expression obtained using its positional and visual information; this path expression is called visual signal. The method aims to identify occurrences of visual signals representing individual data records, parts of data records or sets of data records.

In [19], the proposed method identifies and extracts the main data region based on the visual clue (location of data region, data records and data items on the screen) of web pages. It assumes that the main data region fills the major central portion of the web page and is also a HTML "table" element. As we will demonstrate in Section III, this assertion is not true for the recent web pages, and, thus, other manners to identify the correct main data region are necessary.

The method proposed in [20] represents each element as a symbol and a group of symbols as an alphabet. It analyzes the sequence of symbols of the alphabets to recognize SRRs and, then, the main data region. Afterwards, it excludes all other regions from the source code and uses MDR [21] to read the new structure and extract SRRs and data units. Such method was applied on a collection with 23 search result pages and obtained an accuracy of 86.96% in the task of identifying the main data region.

Some methods assume that SRRs are child elements of the same element ([22], [23]). This assumption may lead to a lower accuracy since the SRRs may have different parent elements. In [24], the same assumption is used and the web pages with child elements of different elements were not considered in the experimental evaluation.

Our method differs from the previous ones since it does not consider the location of the main data region when a browser renders it, does not assume that the elements with SRRs are children of the same element and does not use specific symbols from the HTML language.

III. Proposed Method for Identifying the Main Data Region

With the goal of automatically identifying the main data region of a search result page, we propose a three-step method based on the page rendering information. Our method receives as input a search result page and produces as output an XPath expression to select the main data region. This XPath expression may be used by a wrapper, as initial phase, for extracting the SRRs. The three steps of our method are:

1) The first step generates candidate XPath expressions for selecting the main data region;
2) The second step obtains the visual data based on rendering information; and
3) The third step chooses among the candidate XPath expressions the one that is most likely to select the main data region.

Our method works on the HTML DOM[1] tree, i.e., it manipulates the web page as a tree of objects. In this

[1]http://www.w3.org/DOM/

tree representation, amidst the types of nodes, there are document, element, attribute and text. The *document* node is the root node and has an element node (the element root) as a child. The content of an *element* node, that corresponds to an HTML tag of an HTML page, is the part between the opening and ending tags in the HTML page. The *element* nodes may have elements, attributes and text as child nodes and their child elements are ordered, i.e., we have the first element, second element, and so on. Each *attribute* node has a name and a value attached to an element. And each *text* node has an attached text.

Next, we describe the steps of our method in detail.

A. Determining the candidate XPath expressions

As previously mentioned, the main data region of a search result page (HTML page) contains the SRRs of this page. The main data region may have many SRRs and all SRRs are descendants of the "body" element that is a child of the "html" element (the web page root element). Thus, any other subtree of the "html" element will not be considered by our method (e.g., "head" element).

As first step, our method produces candidate XPath expressions to select the main data region. We consider that all data region with at least a given number of records (i.e., all element with at least a given number of child elements in DOM format) are candidates to be the main data region. We used the $\#_{min}$ parameter to set the minimum number of child elements of a same type that a given parent element must have in order to be considered a candidate. The total number of child nodes of a node does not consider some types of HTML elements that are not used as record delimiters on pages (i.e., elements that indicate form fields, comments, javascript codes, table columns and so on). Thus, our method does not consider elements "input", "textarea", "select", "option", "link", "script", "style", "img", "!–", "td" and "noscript" in the counting process of the total number of child nodes.

After we've selected all nodes that contain at least $\#_{min}$ child elements, our method generates a positional XPath expression for each selected node and adds it to the set of candidate XPath expressions, S. For instance, if our method is executed in a page whose DOM format is shown in Fig. 2, (a) "/html[1]/body[1]/div[1]/ol[1]" and (b) "/html[1]/body[1]/div[2]/ol[1]" are added to S.

In this example, the first expression selects the data region with the page's menu and the second one selects the data region with the page's content containing the SRRs. The final decision regarding the correct XPath expression to select the main data region is obtained in the final of the next two steps.

B. Obtaining the area of each element

In this step, the visual information used by our method corresponds to the attributes (height and width) used to determine the area filled by each element whose expression are in S (i.e., each element selected by the expressions in S) when it is shown in a web browser. In order to obtain such area, our method requires the rendering of the search result page. We use MSHTML[2] to obtain the height and width of each selected data region and then obtain the area.

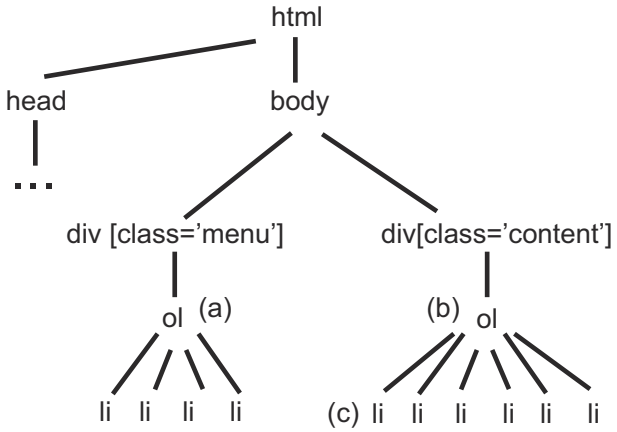

Figure 2. An example of HTML page in DOM format with two data regions (a) and (b), and six SRRs (c).

In detail, the height and width of each element and consequently the area are obtained as follows:

1) First, the web page is loaded by the MSHTML that displays it in the same manner it should be shown by the Internet Explorer browser. When a browser renders a page on the screen, a visual information is associated to each element, such as x and y coordinates and height and width of the element; and
2) Next, our method assigns the corresponding height and width to each element in the DOM tree.

C. Choosing the path expression to select the main data region

The third step of our method must choose, amidst the candidate expressions in S, the one that is most likely to select the main data region. As our candidate expressions are positional expressions, we only have one element selected by each candidate expression, and the previous step has associated to such elements their height and width.

To determine the main data region of a page, our method assumes that this region, which contains the most important information to the user, fills the larger portion of the screen when the browser renders the page.

As formerly stated, the main data region is inside the "body" element of a web page and a browser renders only descendant elements of the "body" element. Thus, to

[2]MSHTML is an dynamic-link (DLL) library that simulates the Internet Explorer browser and may be used by programs written in several visual programming languages.

determine the portion on the screen filled by each element, we calculate the fraction of the area associated to each element regarding the "body" element's area. To determine among the candidate expressions the one that selects the main data region, in this step, our method performs some stages to filter the correct expression in S. Next, we describe each stage.

Filtering the candidate expressions using area

The first stage's aim is to filter, among the candidate expressions, the ones that select an element whose area on the screen figures within the largest ones and, thus, is most likely to contain the SRRs. Formally, let $S = \{exp_1, exp_2, \cdots, exp_n\}$ be the set of n candidate expressions and $E = \{e_1, e_2, \cdots, e_n\}$ be the set of elements selected by each expression in S, where e_i is the element selected by the expression exp_i. Each element e_i and the "body" element have height and width defined in the second step. The area of each element, e_i and "body", is calculated by multiplying its corresponding height and width. We calculate the quota of each area associated to each element e_i ($area(e_i)$) to the area of the "body" element ($area(body)$) and if this quota is bigger than a given threshold $\%_{area}$, the corresponding exp_i is inserted into the set of filtered expressions S', i.e.,

$$S' = \{exp_i \in S | area(e_i)/area(body) \geq \%_{area}\} \quad (1)$$

Filtering the candidate expressions using height

There are elements showed by a browser that fill a large portion on the screen (i.e., a large area), but do not contain SRRs, for instance, the page's header and footer or other elements whose height is very low compared with the "body" element. To solve such problem, we remove from the set S' (i.e., set of filtered expressions) the expressions whose selected elements' heights are too low. If the ratio between the height of an element e_i ($height(e_i)$) and the height of the "body" element ($height(body)$) equals at least a given threshold, $\%_{height}$, its expression exp_i continues belonging to the set S'. Otherwise, the corresponding expression is removed from S', i.e.,

$$S' = S' - \{exp_i \in S' | height(e_i)/height(body) < \%_{height}\} \quad (2)$$

Filtering the candidate expressions using width

Similarly to elements with large areas but low heights, there are also elements whose areas are large but whose widths are short, and do not contain SRRs, for instance, page's side menu. Thus, we also remove from S' the expressions whose corresponding element's width is too short. In order to check whether the width of an element e_i is too short, we divide its width ($width(e_i)$) by the width of the "body" element ($width(body)$) and compare the result with a given threshold $\%_{width}$. If the result is lower than the threshold, we remove its expression from S', i.e.,

$$S' = S' - \{exp_i \in S' | width(e_i)/width(body) < \%_{width}\} \quad (3)$$

Filtering candidate expressions that select ancestral elements

Frequently, a candidate element (i.e., an element selected by a candidate expression) to a main data region contains other candidate elements as children. This situation occurs, for instance, when before a "div"[3] element, that contains the SRRs, there is another "div" element, with a text informing the current shown page (e.g, showing 1 of 10) or yet when after a "div" element there is another "div" element with a panel for changing the page (e.g., "previous page" and "next page"). Thus, both the "div" element containing the SRRs and its parent element may have candidate expressions in S' after the previous filters. It is noticeable that the element with SRRs fills a large area of its parent's area.

In order to remove from S' a candidate expression exp_i that is a prefix of other expression exp_j where the element e_j (selected by exp_j) contains the SRRs, we check whether the ratio between e_j's area and e_i's area is bigger than a given threshold $\%_{parent}$. If the ratio is bigger than $\%_{parent}$, the expression exp_i (the expression that selects the parent element) is removed from S' (the set of candidate expressions), i.e.,

$$S' = S' - \{exp_i \in S' | exp_j \in S' \wedge parent :: e_j = e_i \wedge area(e_j)/area(e_i) > \%_{parent}\} \quad (4)$$

Defining the expression to select the main data region

Among the candidate expressions in S', the one with the largest area will be used to select the main data region, but, this expression may not be able to select the element with all SRRs. This situation occurs because, sometimes, after a browser renders a web page, the SRRs are shown in the same portion on the screen, but, in DOM, the elements that have the SRRs may have sibling elements without SRRs or may have distinct parents, i.e., we cannot consider that elements with SRRs have the same parent element in DOM. An example can be seen in DOM format in Fig. 3. If we apply the previous filters on this DOM, we should have two candidate expressions in S', selecting the "ol" elements in "div" elements (b) and (d). The "ol" element in the "div" element (b) would be considered the main data region since its corresponding area is the largest one. However, this result is incorrect because it does not include all SRRs. As we are using positional expressions, the same

[3]The "div" element defines a division or section in a HTML document.

expression is not able to select both elements. Thus, as an option, we combine the corresponding expressions in only one expression that selects a common ancestral. In Figure 3, the resulting expression would select the "div" element (a).

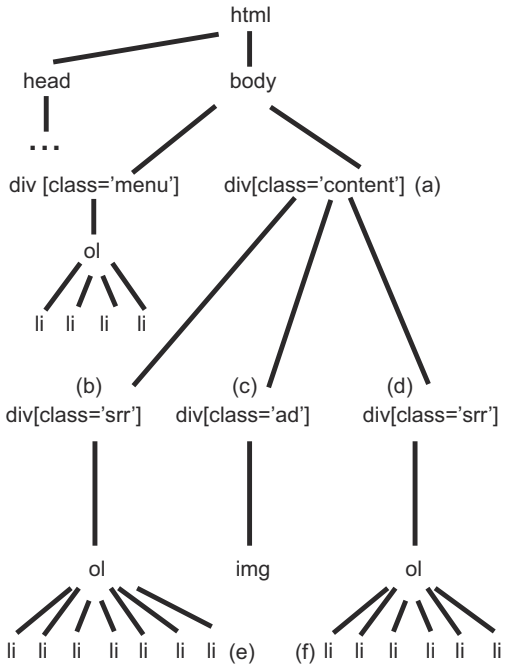

Figure 3. An example of a HTML page in DOM format with two groups of SRRs (e) and (f) in different subtrees (b) and (d). (c) is a region containing advertising and (a) is the main data region.

To solve such situation, our method performs the following stages after the expression exp, with the largest area, and its selected element e, have already been chosen:

1) Let p be the parent of e;
2) If there are descendants of p similar to children of e, p is considered the element with the main data region and a positional expression is generated for it and returned as the expression to select the main data region.
3) If the stage 2 is false, p receives its parent and the stage 2 is performed again.

The method returns to stage 2 at most three times (empirically evaluated). We use the Robust Algorithm for the Tree Edit Distance (RTED) [25] to calculate the similarity between elements (subtrees of DOM). RTED receives as input two elements (subtrees) and returns as output a value related with the number of operations to transform a tree in the other one. For very similar elements, the value is closer to 0. For two identical trees, the value is 0. Our method makes some considerations for using RTED, that are described next.

- For each operation of rename (exchange) of elements, we use weight 3 and, for other operations, we maintain weight at 1.
- To compare two elements (subtrees), we do not use the whole subtree, avoiding the returning of a big value from RTED when a subtree of a SRR with many attributes is compared with another subtree of a SRR with few attributes. We limited the number of levels in the comparison of subtrees to only two levels, i.e., we compare the subtrees considering only the root nodes and the children of the root nodes. Furthermore, as the elements of SRRs are not "br", "h1", "h2", "h3", "h4", "h5", "h6" and "a" elements, that are used to separate visually the SRRs on the screen or show any information related to them, these are ignored.
- As for the final distance between two subtrees that may contain the SRRs, we use the average distance between the possible SRRs in both subtrees. Thus, two subtrees are considered similar if this average distance is, at most, equal to a given threshold δ_{max}.

There are situations in which the set of candidate expressions is empty after the previous filters. In such case, we consider the "body" element as the main data region and our method returns the positional expression "/html[1]/body[1]" as result.

IV. Experimental Evaluation

In this section we present experimental results that demonstrate the effectiveness of our method. We first describe the collections, the baseline and the evaluation metric. Then, we discuss the effectiveness of our method in comparison with the baseline.

A. Collections

In order to evaluate the effectiveness of our method, we adopted collections of web pages used in [17], hereafter referred to as WEBT, and in [20], hereafter referred to as WEBV. The collections are described below.

WEBT collection

The WEBT collection of search result pages was assembled by Trieschnigg et al. [17], combining two collections, Web1 e Web2, and contains 220 web pages. Web1 contains the search result pages obtained from the top 500 US websites listed by Alexa[4], removing websites that require a user account (LinkedIn, Facebook), contain pornographic material and torrent downloads. The authors used the search function on the main page of the webpage for obtaining the search result pages on each website. The complete result page was downloaded in Mozilla Archive Format[5], which includes all images and CSS (cascading style sheet) files and removes javascript. Web2 contains the search result pages obtained from the top UK websites listed by Alexa in a

[4]http://www.alexa.com/topsites
[5]http://maf.mozdev.org/

Table I
DISTRIBUTION OF THE SEARCH RESULT PAGES IN THE APPLICATION DOMAINS.

	WEBV		WEBT	
Domain	# of pages	% of pages	# of pages	% of pages
Academic	3	13.04	3	1.36
Vehicle & Real Estate	1	4.35	4	1.82
Bank	2	8.70	5	2.27
Books	0	0.00	1	0.45
Community	0	0.00	11	5.00
Company	0	0.00	19	8.64
e-Commerce	9	39.13	39	17.73
Game	0	0.00	1	0.45
Government	0	0.00	3	1.36
Images	0	0.00	19	8.64
Job	0	0.00	6	2.73
Music	0	0.00	3	1.36
News	5	21.74	50	22.73
Recipes	0	0.00	1	0.45
Search Engine	2	8.70	26	11.82
Technical Articles	0	0.00	13	5.91
Video	1	4.35	16	7.27
All Domains	23		220	

manner similar to that used for Web1. Websites already in Web1 were removed.

WEBV collection

The WEBV collection was assembled by Velloso and Dorneles [20] and contains 23 search result pages. The search result pages of this collection contains only the source code of the HTML pages, without images and CSS. Thus, its pages are wrongly rendered and present with a messy layout, differently from when we access it on the Web. In order to solve such problem, we accessed the websites again, searched for the same terms as in [20] and saved the whole search result pages on disc, including images and CSS.

Table I shows the popularity of each application domain in both collection.

B. Baseline

We used as baselines the works proposed in [20] and [17], from which we got the collections.

In [20], as described in Section II, the authors attempt to identify the main data region checking sequences of symbols that represent tags in HTML document. Thus, all other regions are removed from the DOM tree and the result is saved as a new page. Next, this new page is used as input for MDR [21] to improve its accuracy, since the number of analyzed nodes has decreased. We check the new pages generated by the method in order to calculate its accuracy. The main data region is correctly identified from a web page, if the new page contains only the main data region from the original page.

In [17], the authors propose a method to generate XPath expressions to extract the SRRs from the pages. In order to use it as baseline, we perform it on each page from the collections and check whether the generated XPath expressions select SRRs from only the main data region. If at least a selected SRR cames from other regions (e.g., menu), we consider the identification of the main data region as incorrect. This method has three parameters: *minSimilarityThreshold*, *avgSimilarityThreshold* and *minimumNodeCount*.

C. Evaluation metric

In order to evaluate the effectiveness our method, we use $accuracy$ metric, calculated as follows:

$$accuracy = \frac{\#correct}{\#total} \quad (5)$$

$\#correct$ is the total number of main data regions from the pages correctly identified by the expression provided by the method in the collection and $\#total$ is the total number of pages in the collection.

D. Experimental setup

Experiments were conducted in each collection. We perform each method on each web page and manually check whether the main data region was correctly selected.

To perform our method, we set $\#_{min}$=3 (the minimum number of child elements), $\%_{area}$=0.1 (the percentage of the area filled by a given element), $\%_{height}$=0.2 (the percentage of the height of a given element), $\%_{width}$=0.3 (the percentage of the width of a given element), δ_{max}=2 (the average distance between subtrees) and $\%_{parent}$=0.2 (the ratio between an element and its parent, when both are candidates). These values are the best parameter settings on WEBT. We make a sensitive analysis to the parameter values in Section IV-E.

To perform the method proposed in [17], we used the same parameter values defined by the authors: *minSimilarityThreshold* = 0.55, *avgSimilarityThreshold* = 0.65 and *minimumNodeCount* = 3. Regarding the method proposed in [20], the only parameter is the minimum size percentage of a region compared with the rest of the sequence, so that a given region can be considered a main data region. This percentage is 0.2.

E. Results

Effectiveness of our method

We have show in Table II the number of main data regions incorrectly selected by our method in WEBV and WEBT collections in each domain. The last line of the table shows the accuracy of our method in each collection.

In the WEBV collection, only 2 main data regions were incorrectly selected by our method. We analyze each case and notice that the first search result page ("bradesco.com.br"), whose main data region was incorrectly selected, uses the "iframe" element to encompass the SRRs. This type of element inserts a page inside another one. Thus, in such a situation, our method is not able to provide an expression to select the main data region, since the "iframe" element appears empty.

Another main data region incorrectly selected by our method is the one in the search result page of the web site

Table II
EFFECTIVENESS OF OUR METHOD IN EACH COLLECTION.

Domain	# of main data regions incorrectly selected WEBV	WEBT
Academic	0	0
Vehicle & Real Estate	0	0
Bank	1	1
Books	0	0
Community	0	0
Company	0	1
e-Commerce	0	3
Game	0	0
Government	0	0
Images	0	2
Job	0	1
Music	0	1
News	1	8
Recipes	0	0
Search Engine	0	1
Technical Articles	0	0
Accuracy	91.30	91.82

"reuters.com". This page contains some elements that are not SRRs, but whose structures are similar to its SRRs. These similar elements induce our method in the last strategy of the third step (defining the expression to select the main data region) to incorrectly construct the XPath expression.

In the WEBT collection, amidst 220 search result pages, we have 18 main data regions incorrectly selected by our method. We checked the corresponding 18 pages and noticed that:

- In 10 search result pages, the failure occurs because our method, in the last strategy of the third step, when trying to find similar subtrees with the subtrees of the SRRs already selected, finds similar subtrees that do not contain SRRs or, due to similarity threshold δ_{min}, subtrees with SRRs are not considered similar;
- In 7 search result pages, the failure occurs due to the fact that elements without SRRs have bigger areas than the element with SRRs, passing through all steps; and
- A search result page, when loaded by the browser, is automatically redirected to another page, and, as the page is different from the page in the collection, we consider incorrect the selection of the main data region.

Effectiveness of our method compared with baselines

Table III shows the comparison of our method with the baselines in both collections, in term of accuracy to select the main data region. In the WEBV collection, the baselines obtained the same accuracy (86.96%). Our method has a gain of around 5% when compared to both baselines. In the WEBT collection, the gains of our method are around 100% and 2.5% compared with the methods proposed in [20] and [17], respectively.

Table III
COMPARISON THE EFFECTIVENESS (ACCURACY) OF OUR METHOD WITH THE BASELINES

Methods	Collections WEBV	WEBT
Our method	91.30	91.81
Proposed method in [20]	86.96	45.45
Proposed method in [17]	86.96	89.55

Table IV
EFFECTIVENESS OF OUR METHOD WITHOUT CONSIDERING A FILTER OR STRATEGY OF THE THIRD STEP.

Filter/Strategy	WEBV	WEBT
Area	86.95	91.81
Height	86.95	87.72
Width	86.95	90.45
Area of parent	0.00	4.55
SRRs in other subtrees	78.26	90.0

Influence of each filter in the final performance

In order to evaluate the influence of each filter/strategy of the third step in the performance of our method, we performed our method without one of the filter/strategies in the last step. Table IV shows the effectiveness (accuracy) of our method in such situations.

We can notice that, in the WEBV collection, when we remove the comparison of the area, height and width of an element selected by a candidate expression with the values of the "body" element, the accuracy of our method drops around 4.8%. When we do not search for SRRs in other subtrees (i.e., subtrees that are not selected by the expression obtained up to such moment) the performance drops around 14.3%. In this collection, the filter that compares the area of a region with the area of its parent leads to the worst performance (our method does not select correctly any main data region).

In the WEBT collection, the filter that compares the area of an element with the "body" element does not alter the final result. When we compare height and width of the elements, as well as the search for similar subtree (SRRs in other subtrees), there is a drop of performance of our method around only 4.5%, 1.5% and 2.0%, respectively. But, as happened in WEBV, if we do not compare the area of an element with its parent's area, our method has a poor performance.

Sensitive to the parameters

We investigated the sensitive of our method to the parameters $\#_{min}$, δ_{max}, $\%_{area}$, $\%_{height}$, $\%_{width}$ and $\%_{parent}$. In each run, we vary the values of only one parameter. Figure 4 shows the accuracy on the WEBT collection when values of $\#_{min}$ and δ_{max} range from 1 to 10. Notice that, there exists a low variation of accuracy when we vary the $\#_{min}$ values. The best accuracies are obtained when we vary the $\#_{min}$ values from 2 to 4. About δ_{max}, the best accuracies are

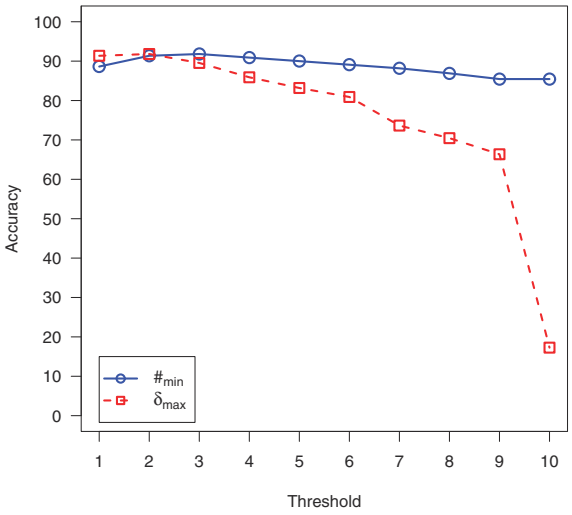

Figure 4. Sensitive to $\#_{min}$ and δ_{max}

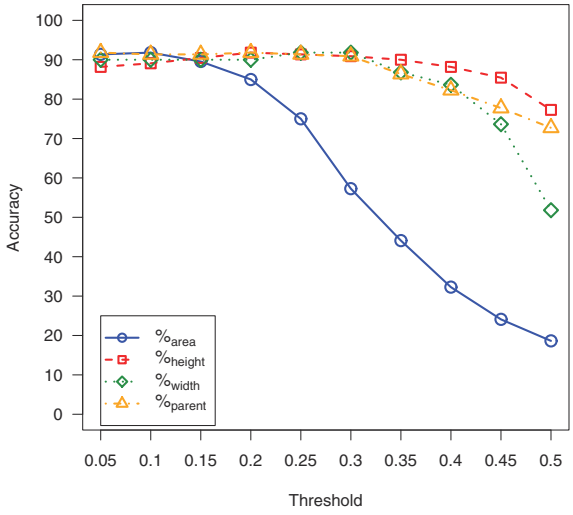

Figure 5. Sensitive to $\%_{area}$, $\%_{height}$, $\%_{width}$ and $\%_{parent}$

obtained when δ_{max}=1 and δ_{max}=2. There exists a slight decrease of accuracy from δ_{max}=3 to δ_{max}=9.

Figure 5 shows the accuracy on the WEBT collection when we vary the values of $\%_{area}$, $\%_{height}$, $\%_{width}$ and $\%_{parent}$ from 0.05 to 0.50. About $\%_{height}$, $\%_{width}$ and $\%_{parent}$, for the values lower than 0.30, our method has the best performances. For $\%_{area} \leq 0.15$, the accuracies are around 90%.

F. Discussion

Although our experiments showed that our method is effective, it has some limitations, which we discuss here.

As any method that works with DOM trees of HTML pages, ours supposes that the source codes of such pages are well structured and have no errors. Although there are many HTML parsers that correct most of these errors, they do not ensure total accuracy. Such characteristic may lead our method to erroneously obtain the height and width of the elements.

Another weakness of our method is due to the incorrect assignment of the rendering information by the browser or MSHTML. For instance, there are situations in which an element filled a large area on the screen but its height and width values are zero. This occurs when we use MSHTML, as well as other web browsers (e.g., Mozilla).

V. Conclusion

In this paper, we propose a method that combines structure and rendering information of HTML pages to generate a positional XPath expression to extract their main data region. Our method uses the height, width and, consequently, the area of each element when a browser renders the page, along with the number of child elements in DOM tree to decide whether an element encompasses the main data region, i.e., the region with the search result records. Our method is unsupervised (i.e., it does not need any examples), domain independent and performs in one page (i.e., it does not need several pages from the same web site to infer how to select the main data region).

We evaluate our method in two collections of search result pages from several domains and compare it with two baselines. The accuracy of our method is higher than 91% and gains are up to 100% when in comparison with the baselines.

As future work, we intend to generate XPath expressions containing predicates to better handle changes on the structure of the pages. Thus, after the method performs on a page from a web site and generates a expression to select the main data region, it will be able to use the same expression to select the main data regions from other pages of the same web site. In this work, when we compare two elements, we compare only their structure (i.e., their subtrees). To improve the method performance, we intend to compare their contents in order to check their similarity. Moreover, we also intend to extend our method for extracting the search result records.

Acknowledgments

This research is partially funded by FAPEMIG (Foundation for Research Support of the State of Minas Gerais).

Moreover, this research was carried out on the Laboratory of Management and Intelligent Analysis of Data of the Federal University of Ouro Preto (GAID/UFOP).

REFERENCES

[1] G. Miao, J. Tatemura, W.-P. Hsiung, A. Sawires, and L. E. Moser, "Extracting data records from the web using tag path clustering," in *Proceedings of the 18th International Conference on World Wide Web*, ser. WWW '09. Madrid, Spain: ACM, 2009, pp. 981–990.

[2] R. Song, H. Liu, J.-R. Wen, and W.-Y. Ma, "Learning block importance models for web pages," in *Proceedings of the 13th International Conference on World Wide Web*, ser. WWW '04. New York, NY, USA: ACM, 2004, pp. 203–211.

[3] A. Sahuguet and F. Azavant, "Wysiwyg web wrapper factory (w4f)," in *Proceedings of the 8th International Conference on World Wide Web*, Toronto, Canada, 1999, pp. 1–22.

[4] L. Liu, C. Pu, and W. Han, "Xwrap: An xml-enabled wrapper construction system for web information sources," *16th International Conference on Data Engineering (ICDE'00)*, vol. 0, p. 611, 2000.

[5] V. Crescenzi, G. Mecca, and P. Merialdo, "Roadrunner: Towards automatic data extraction from large web sites," in *Proceedings of the 27th International Conference on Very Large Data Bases*, ser. VLDB '01. Roma, Italy: Morgan Kaufmann Publishers Inc., 2001, pp. 109–118.

[6] A. Arasu and H. Garcia-Molina, "Extracting structured data from web pages," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '03. San Diego, California: ACM, 2003, pp. 337–348.

[7] J. Wang and F. H. Lochovsky, "Data extraction and label assignment for web databases," in *Proceedings of the 12th International Conference on World Wide Web*, ser. WWW '03. Budapest, Hungary: ACM, 2003, pp. 187–196.

[8] Y. Zhai and B. Liu, "Web data extraction based on partial tree alignment," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. Chiba, Japan: ACM, 2005, pp. 76–85.

[9] N. Kushmerick, "Wrapper induction for information extraction," Ph.D. dissertation, University of Washington, 1997, aAI9819266.

[10] I. Muslea, S. Minton, and C. Knoblock, "Stalker: Learning extraction rules for semistructured," in *In American Association for Artificial Intelligence (AAAI): Workshop on AI and Information Integration*, Madison, Wisconsin, USA, 1998.

[11] C.-N. Hsu and M.-T. Dung, "Generating finite-state transducers for semi-structured data extraction from the web," *Information Systems*, vol. 23, no. 9, pp. 521–538, 1998.

[12] C.-H. Chang and S.-C. Lui, "Iepad: Information extraction based on pattern discovery," in *Proceedings of the 10th International Conference on World Wide Web*, ser. WWW '01. Hong Kong, Hong Kong: ACM, 2001, pp. 681–688.

[13] B. Adelberg, "Nodose: A tool for semi-automatically extracting structured and semistructured data from text documents," *Special Interest Group on Management of Data - SIGMOD Rec.*, vol. 27, no. 2, pp. 283–294, 1998.

[14] A. H. F. Laender, B. Ribeiro-Neto, and A. S. da Silva, "Debye - data extraction by example," *Data and Knowledge Engineering*, vol. 40, no. 2, pp. 121–154, 2002.

[15] D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, and R. D. Smith, "Conceptual-model-based data extraction from multiple-record web pages," *Data and Knowledge Engineering*, vol. 31, no. 3, pp. 227–251, 1999.

[16] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "Extracting content structure for web pages based on visual representation," in *Proceedings of the 5th Asia-Pacific Web Conference on Web Technologies and Applications. Xian, China.*, ser. APWeb'03. Xian, China: Springer-Verlag, 2003, pp. 406–417.

[17] R. B. Trieschnigg, K. T. T. E. Tjin-Kam-Jet, and D. Hiemstra, "Ranking xpaths for extracting search result records," Centre for Telematics and Information Technology, University of Twente, Enschede, Technical Report TR-CTIT-12-08, 2012.

[18] H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu, "Fully automatic wrapper generation for search engines," in *Proceedings of the 14th International Conference on World Wide Web*, ser. WWW '05. Chiba, Japan: ACM, 2005, pp. 66–75.

[19] P. S. Hiremath and S. P. Algur, "Extraction of data from web pages: A vision based approach," *International Journal on Computer Science and Engineering*, vol. 1, no. 3, pp. 50–59, 2009.

[20] R. P. Velloso and C. F. Dorneles, "Automatic web page segmentation and noise removal for structured extraction using tag path sequences." *Journal of Information and Data Management*, vol. 4, no. 3, pp. 173–187, 2013.

[21] B. Liu, R. Grossman, and Y. Zhai, "Mining data records in web pages," in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '03. Washington, D.C.: ACM, 2003, pp. 601–606.

[22] I. V. Jabour, "Impacto de atributos estruturais na identificação de tabelas e listas em documentos html," Mestrado, Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática, 2010.

[23] T. Grigalis, "Towards web-scale structured web data extraction," in *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, ser. WSDM '13. Rome, Italy: ACM, 2013, pp. 753–758.

[24] M. Álvarez, A. Pan, J. Raposo, F. Bellas, and F. Cacheda, "Extracting lists of data records from semi-structured web pages," *Data and Knowledge Engineering*, vol. 64, no. 2, pp. 491–509, 2008.

[25] M. Pawlik and N. Augsten, "Rted: A robust algorithm for the tree edit distance," *Very Large Data Bases (VLDB) Endowment*, vol. 5, no. 4, pp. 334–345, 2011.