PAPER

A Web Page Segmentation Approach Using Visual Semantics

Jun ZENG[†*a)], **Brendan FLANAGAN**[†], **Sachio HIROKAWA**[††], ***Nonmembers***, ***and*** **Eisuke ITO**[††], ***Member***

SUMMARY Web page segmentation has a variety of benefits and potential web applications. Early techniques of web page segmentation are mainly based on machine learning algorithms and rule-based heuristics, which cannot be used for large-scale page segmentation. In this paper, we propose a formulated page segmentation method using visual semantics. Instead of analyzing the visual cues of web pages, this method utilizes three measures to formulate the visual semantics: layout tree is used to recognize the visual similar blocks; seam degree is used to describe how neatly the blocks are arranged; content similarity is used to describe the content coherent degree between blocks. A comparison experiment was done using the VIPS algorithm as a baseline. Experiment results show that the proposed method can divide a Web page into appropriate semantic segments.

key words: *web page segmentation, visual semantics, seam degree, content similarity*

1. Introduction

On the internet, web pages are often considered the smallest indivisible units of information. For example, the major commercial search engines build an index from web pages. However web pages are not atomic units on the web. A typical web page consists of multiple segments with different functionalities, such as main content, navigation bar, menu list, advertisements. The process to divide a web page into visually and semantically cohesive segments is so called web page segmentation.

Web page segmentation has a variety of benefits and potential web applications, such as browsing web pages on mobile devices [1]–[3], detecting duplicate web pages [4], information extraction [5]–[7].

The early techniques of web page segmentation are mainly based on machine learning algorithms [1], [4], [8], [9] and rule-based heuristics [2], [3], [5]–[7], [10]–[12], [15]. Because of the small scale training data set, machine-learning-based methods can only be applied in some certain fields of web pages. The heuristics-based approaches involve simple rule-based heuristics either by interpreting the meaning of tag structures or visual analysis. While a heuristic approach might work well on small sets of pages, it isn't suitable for large-scale sets of pages.

In this paper, we assume that a web page is made up of finite blocks and the web pages can be segmented using the visual features of blocks. The visual features can be considered as the following three parts: (1) similar visual blocks have similar semantics, e.g. in a shopping site, the product records are arranged in a similar layout; (2) relevant blocks are always neatly arranged and put visually close together; (3) blocks with different functionalities contain different types of contents, e.g. in a news site, long text may be the main content; a link list may be the related news list; a big picture may be an advertisement, etc. Due to the different visual features, humans can easily identify each of the segments without any descriptions. We call these visual features visual semantics. However, these semantics are intuitive and human friendly. In other words, they are not machine friendly and therefore difficult to be understood by computers. This issue gives rise to the question: How can these visual semantics be formulated? We use three formulated measures to represent these visual semantics: layout tree [18] is used to recognize the similar visual blocks; seam degree is used to describe how neatly the blocks are arranged; content similarity is used to describe the content coherent degree between the blocks. Based on these three measures, we proposed a web page segmentation method. The experiment results show that the proposed method can divide a web page into appropriate semantic segments.

The rest of the paper is organized as follows: Related works are reviewed in Sect. 2. Visual block and preprocessing of web pages are introduced in Sect. 3. A web page segmentation method using visual semantics is proposed in Sect. 4. Experiment analysis and results are reported in Sect. 5. Finally, conclusion and future work are given in Sect. 6.

2. Related Work

In the past few years, there has been plenty of work on automatic web page segmentation.

Some of the early approaches are based on machine learning algorithms [1], [4], [8], [9]. These approaches segment pages by training the clues from DOM or simple vision cues. Machine-learning-based approaches can only be applied in some certain fields of web pages, because of the limitation of the training data set. Because these algorithms need to be trained, they can be regarded as semi-automatic approaches.

Manuscript received May 21, 2013.
Manuscript revised September 25, 2013.

[†]The authors are with the Graduate School of Information Science and Electrical Engineering, Kyushu University, Fukuoka-shi, 812–8581 Japan.

[††]The authors are with Research Institute for Information Technology, Kyushu University, Fukuoka-shi, 812–8581 Japan.

[*]Presently, with the School of Software Engineering, Chongqing University, Chongqing, 400044, China.

a) E-mail: zengjun@cqu.edu.cn

DOI: 10.1587/transinf.E97.D.223

In order to automatically segment web pages, heuristics-based approaches were proposed. Some of the heuristics-based approaches use HTML structure tags or DOM tree to segment a web page [3], [11], [12], [15]. These methods also have some limitations, for example: these methods may falsely separate closely related contents and combine unrelated contents together. Some other heuristics-based approaches rely on visual cues from browser renderings [2], [5]–[7], [10]. Most of them focus on the location, size or font cues of web pages. Hereinto, VIPS [10] is considered to be the most representative visual-cue-based algorithm. It has three steps: first, a web page is recursively divided into blocks by using a number of heuristics; second, horizontal and vertical separators are determined; third, the structure of the page is constructed. These approaches can make good use of the visual features of web pages. However, heuristics are often based on simple models that cannot be generalized. In other words, even through a heuristic approach might work well on small sets of pages it isn't suitable for large sets of pages.

Because of the limitation of heuristics-based approaches, many non-heuristics-based approaches [13], [14], [16], [17] were proposed. X. Liu et al. [13] proposed a Gomory-Hu Tree based Web page segmentation algorithm. The algorithm firstly extracts vision and structure information from a web page to construct a weighted undirected graph, whose vertices are the leaf nodes of the DOM tree and the edges represent the visible position relationship between vertices. Then it partitions the graph with a Gomory-Hu tree based clustering algorithm. G. Hattori et al. J. Kong et al. [14] proposed Spatial Graph Grammar (SGG) to perform the semantic grouping and interpretation of segmented screen objects. Instead of analyzing HTML source codes, they applied an image processing technology to recognize atomic interface objects from the screenshot of an interface and produce a spatial graph, which records significant spatial relations among recognized objects. However, there are not enough quantified experiment results to indicate that SGG is effective to segment any kinds of web pages. J. Kang et al. [16] proposed repetition-based web page segmentation method. They consider the repetitive tag patterns to be key patterns in the DOM tree structure of a page. By detecting key patterns in a page and generating virtual nodes to correctly segment nested blocks, the method can segment pages into logical blocks. However, this method is only suitable for the pages that contain repetitive patterns. C. Kohlschtter et al. [17] utilized the notion of text-density as a measure to identify the individual text segments of a web page. Although, his method can reduce the problem to solving a 1D-partitioning task, it can be used for small-scale pages that have certain patterns.

Our work can be classified as a vision-based approach. Different from the visual-cue-based method, e.g. VIPS, our work formulates the visual feature as quantified and formulated measures. Based on these measures, the proposed approach can divide web pages into semantic segments.

3. Preliminaries

3.1 Visual Blocks

A web page is made up of finite blocks. We also call these blocks visual block or block for short. We consider a visual block as a visible rectangular region on a web page. The definition of a visual block is as follows:

Definition 3-1: Visual block $B = (Obj, Rect)$, where Obj is a DOM object, and $Rect$ represents the visible rectangular region where B is displayed in the web page.

According to W3C standard, a web page can be transformed into a DOM tree, and each DOM object has a corresponding element in the web page. If an element is visible, it will be displayed within a rectangular region in a web page. Therefore, a DOM object and its rectangular region (if it is visible) represent a block in a web page. Moreover, we define the child block and leaf block as follows:

Definition 3-2: For two given visual blocks $B_1 = (Obj_1, Rect_1)$ and $B_2 = (Obj_2, Rect_2)$, if Obj_1 is a child node of Obj_2, then B_1 is the child block of B_2.

Definition 3-3: If a visual block $B = (Obj, Rect)$ does not have any child blocks, then B is a leaf block.

3.2 Pre-Processing Web Pages

According to Definition 3-1, each visual block has a corresponding DOM object. Thus, we first obtain the DOM tree of the web page.

In a DOM tree, each node is a DOM object. The DOM objects can be divided into five types of objects: element, attribute, text, comment and document. We further classify the element objects into two categories: visible element objects and invisible element objects. The visible element objects whose *width* and *height* properties are not *zero* and the *display* property is not *none* can be seen through the browser. The invisible element objects contains objects whose tags are <head>, <script>, <meta>, etc, which do not have visual attributes. Moreover, we classify the visible element objects into two categories: inline objects and line-break objects. Inline objects affect the appearance of text and can be applied to a string of characters without a line break, including objects whose tags are <b>, <big>, <font>, etc. The other visible element objects are line-break nodes. Obviously, only the visible element objects and text objects can be displayed in Web pages. Thus we need to prune the DOM tree. The pruning rules are as follows:

Rule 1: The attribute nodes, comment nodes, and document nodes should be cut.

Rule 2: The invisible nodes whose tags are <head>, <script>, <meta>, etc should be cut.

Rule 3: The visible nodes whose *width* and *height* properties are *zero* and *display* property is *none* should be cut.

Rule 4: If a node contains only one node whose node name is <#text>, then the <#text> node should be cut.

Rule 5: If a node only contains <#text> nodes and inline nodes, and each inline node has only one <#text> node, then all the <#text> nodes and inline nodes should be cut.

As mentioned before, Rule 1, Rule 2, and Rule 3 aim to prune the nodes that cannot be displayed in web pages. As for Rule 4, the text that appears in a web page is within a tag in HTML, such as text within a <p> tag. However, in the DOM tree, the <p> node contains a child node whose node name is <#text>. The <#text> node does not have width and height properties, and its parent node <p> also contains the text information of the <#text> node. Even if the <#text> nodes are cut, its text information will not be lost. Rule 4 is used to cut such <#text> nodes. Similarly, Rule 5 aims to prune the inline nodes with nested <#text> nodes.

After the DOM tree has been pruned, only a part of visible nodes and text nodes remains in the DOM tree. It should be noted that both element nodes and text nodes have corresponding objects of the DOM. According to Definition 3-1, each visual block has a corresponding DOM Element object and a rectangular region. Thus, it is necessary to get the corresponding rectangular region of each Element object. The Element object does not contain the absolute coordinate of the corresponding HTML element, and it only contains a relative coordinate to the parent HTML element. Fortunately, some browsers provide APIs to get absolute coordinate easily. As for the text nodes, the width and height can be indirectly calculated by analyzing the width and height of parent node and sibling nodes. After rectangular regions are determined, the corresponding visual blocks are also determined.

Fig. 1 An example of similar visual blocks.

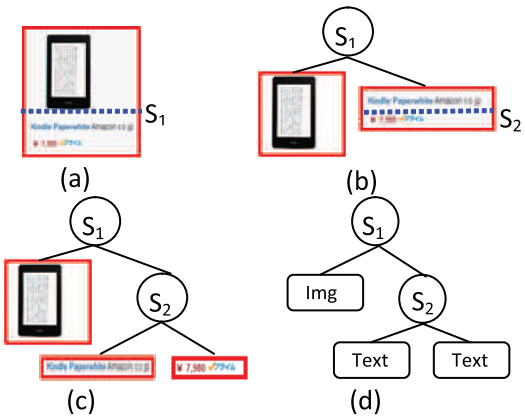

Fig. 2 Generation of layout tree.

4. Web Page Segmentation Using Visual Semantics

4.1 Recognizing Similar Visual Blocks Using Layout Tree

Some pages contain similar visual blocks. For example, Fig. 1 shows a page of a shopping site, and the red rectangles indicate the similar visual blocks. Each block is a product record, thus we consider that these blocks have independent semantic and they should not be divided into smaller segments. Therefore, we should recognize the similar visual blocks in advance.

In our early work, we proposed a layout tree based method to identify the similar visual blocks [18]. For a given block, if the block is not a leaf block, we can transform the block into a layout tree as shown in Fig. 2.

In the Fig. 2, there are two separators S_1 and S_2. Each separator can divide the block into two smaller parts. The separators can be considered as a root of a tree, and the two smaller parts can be considered as the left subtree and the right subtree. Generally, if the separator is horizontal, the upper part is the left subtree and lower part is the right subtree. If the separator is vertical, the left part is the left subtree and right part is the right subtree. Therefore, the given block can be transformed as a tree. We call the tree a "layout tree".

For two given blocks, first they are transformed into two layout trees respectively. Then the Tree Edit Distance (TED) algorithm [19] is used to calculate the similarity of two layout trees. If the similarity is less than the threshold, then the two blocks are visual similar. According to our early experiments, the optimal threshold is 0.4. Using this method all the similar visual blocks can be recognized. Due to paucity of space we only introduce the method roughly. See paper [18] for the detailed algorithm.

4.2 Calculating Seam Degree of Blocks

Because the blocks are visible rectangles in a web page, they are always arranged by certain rules. The relevant blocks are always neatly arranged and put visually close together. For any two given blocks, their arrangements can be classified into three types as shown in Fig. 3. In Fig. 3 (a), a_1 and a_2 are not adjacent, thus we consider them to be visually irrelevant. In Fig. 3 (b) (c), b_1 and b_2, c_1 and c_2 are adjacent blocks. Intuitively, we consider c_1 and c_2 are closer than b_1 and b_2. Suppose there is a minimum rectangle that can just cover the two blocks in Fig. 3 (b) and Fig. 3 (c). c_1 and c_2 can fully fill up the minimum rectangle, but b_1 and b_2 cannot fill up it. It is known that each segment has a corresponding rectangle appearing in the page. In other words, c_1 and c_2 are more likely to be a segment, but b_1 and b_2 cannot be considered as a segment. We utilize seam degree to describe how close the two blocks are arranged. For two given adjacent blocks

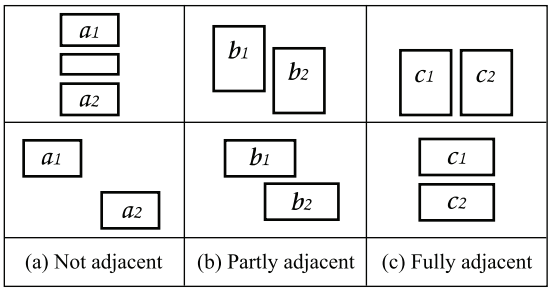

Fig. 3 Three arrangement types of two blocks.

B_1 and B_2, if they are upper-lower adjacent, the seam degree $SD(B_1, B_2)$ can be calculated as in formula (1):

$$SD(B_1, B_2) = \frac{SeamLength(B_1, B_2)^2}{Width(B_1) \times Width(B_2)} \quad (1)$$

where $SeamLength(B_1, B_2)$ represents the seam length of B_1 and B_2, and $Width(B_i)$ represents the width of B_i. Similarly, if B_1 and B_2 are left-right adjacent, the seam degree $SD(B_1, B_2)$ can be calculated as in formula (2):

$$SD(B_1, B_2) = \frac{SeamLength(B_1, B_2)^2}{Height(B_1) \times Height(B_2)} \quad (2)$$

where $Height(B_i)$ represents the height of B_i.

$SD(B_1, B_2)$ is between 0 and 1. Since the seam degree is based on the visual information of blocks, it can indicate the visual coherent degree of adjacent blocks.

If a block has child blocks, the average seam degree of adjacent child blocks can indicate the visual coherent degree of the child blocks in the block. For a given visual block B, the set of child blocks in B is $Child(B) = \{b_1, b_2, \cdots, b_n\}$. If two child blocks b_i and b_j are adjacent, we count 1 pair. Let us assume that there are m pairs of adjacent child blocks. The averaging seam degree $AvgSD(B)$ can be calculated as in formula (3):

$$AvgSD(B) = \frac{\sum SD(b_i, b_j)}{m} \quad (3)$$

where b_i and b_j ($i \neq j$) are two adjacent child blocks. $AvgSD(B)$ degree is also between 0 and 1. If it is closer to 0, the visual coherent degree of child blocks is lower. If it is closer to 1, the visual coherent degree of child blocks is higher.

4.3 Calculating Content Similarity of Blocks

Blocks with different semantics always have different types of contents. For example, a navigation bar has a list of short link text; an advertisement has a big picture; a user registration form has some text boxes, pull-down menus, buttons, etc. These different contents have different visual features. If the contents of two blocks are similar, the two blocks have a high content coherent degree. We introduce the Content Similarity to describe the content coherent degree. We roughly classify the contents into four categories:

1. **Text Contents (TC):** all the text falls into this category, except the text that contains a hyper link.
2. **Link Text Contents (LTC):** the text that contains a hyper link can be classified into this category.
3. **Image Contents (IMC):** this category contains pictures, photos, icons, etc.
4. **Input Contents (INC):** this category includes elements that can accept user input, such as: text box, radio button, pull-down menus, etc.

For a given block B, the content set of B is $C = \{c_1, c_2, \cdots, c_n\}$. First, the contents are classified into the four categories mentioned above. Then four types of content sets can be obtained, denoted TC, LTC, IMC, and INC. Obviously, TC, LTC, IMC and INC are the subsets of C. If one of the content subsets is ϕ, it means that B does not contain the corresponding type of the contents. The contents are also the elements of web page, thus each of them has a corresponding block. We use $Area(c_i)$ to represent the area of the corresponding block of content c_i. If c_i is a text content or link text content, we approximately calculate the area as in formula (4):

$$Area(c_i) = Length(c_i) \times FontSize(c_i)^2 \ (c_i \in TC \cup LTC) \quad (4)$$

where $Length(c_i)$ represents the character byte size of text or link text, $FontSize(c_i)$ represents the font size of text or link text.

For a given content subset (TC, LTC, IMC or INC), according to the area of contents, the content of the given subset can be sorted from large to small area. By utilizing the sorted content subsets, four content area vectors can be obtained, denoted V_{tc}, V_{ltc}, V_{imc} and V_{inc}. The values of elements in the four vectors are the areas of corresponding contents. After the content vectors are determined, the content similarity of two blocks can be calculated.

If the content area vectors of two given blocks are determined, the similarity of each content area vector (V_{tc}, V_{ltc}, V_{imc} and V_{inc}) can be calculated. There are many algorithms to calculate the similarity of two vectors, of which the cosine similarity is a simple and efficient algorithm [20]. Here we take the vector of the text content as an example to explain the calculation of cosine similarity. For two given blocks B_1 and B_2, their text content area vectors are $V_{tc\_1} = (u_1, u_2, \cdots, u_m)$ and $V_{tc\_2} = (v_1, v_2, \cdots, v_n)$. Let us assume that $V_{tc\_1} \neq \phi$, $V_{tc\_1} \neq \phi$, and $n > m$. Because the cosine similarity requires that the two vectors must have the same number of elements, we need to add $(n - m)$ elements whose value are 0 into V_{tc_1}, denoted $V'_{tc\_1} = (u_1, u_2, \cdots, u_m, u_{m+1}, \cdots, u_n)$. The cosine similarity of V'_{tc_1} and V_{tc_2} can be calculated as in formula (5):

$$Cos(V'_{tc\_1}, V_{tc\_2}) = \frac{\sum_{i=1}^{n} u_i \times v_i}{\sqrt{\sum_{i=1}^{n} (u_i)^2} \times \sqrt{\sum_{i=1}^{n} (v_i)^2}} \quad (5)$$

If both V'_{tc_1} and V_{tc_2} are ϕ, $Cos(V'_{tc\_1}, V_{tc\_2})$ is ill-formed. In this case, we define the $Cos(V'_{tc\_1}, V_{tc\_2})$ to be

zero. Similarly, the cosine similarity of other content area vectors (including V_{ltc}, V_{imc} and V_{inc}) can also be determined.

Additionally, the four types of contents may have different weight in B_1 and B_2. Also, we take the text content as an example to explain the calculation of weight. For two given blocks B_1 and B_2, their text content area vectors are $V_{tc\_1} = (u_1, u_2, \cdots, u_m)$ and $V_{tc_2} = (v_1, v_2, \cdots, v_n)$. The weight of text content can be calculated as in formula (6):

$$Weight(Tc) = \frac{\sum_{i=1}^{m} u_i + \sum_{j=1}^{n} v_j}{Area(B_1) + Area(B_2)} \quad (6)$$

where the $Area(B_i)$ represents the total area of all contents in B_i. It means that the greater the area of the corresponding type of contents is, the higher its weight will be.

After the cosine similarity and weight of each content area vector are determined, the content similarity $CS(B_1, B_2)$ of B_1 and B_2 can be calculated as in formula (7):

$$CS(B_1, B_2) = \sum Weight_i \times Cos_i \quad (7)$$

where $Weight_i$ represents the weight of four types of contents (TC, LTC, IMC and INC), and Cos_i represents the cosine similarity of the corresponding content area vector. $CS(B_1, B_2)$ is between 0 and 1. Since the content similarity is based on the content information of blocks, it can indicate the content coherent degree of blocks.

If a block has child blocks, the average content similarity of adjacent child blocks can indicate the content coherent degree of the child blocks in the block. It should be noted that only the content similarity of adjacent child blocks is considered. For a given visual block B, the set of child blocks in B is $Child(B) = \{b_1, b_2, \cdots, b_n\}$. If two child blocks b_i and b_j are adjacent, we count 1 pair. Let us assume that there are m pairs of adjacent child blocks. The average content similarity $AvgCS(B)$ can be calculated as in formula (8):

$$AvgCS(B) = \frac{\sum CS(b_i, b_j)}{m} \quad (8)$$

where b_i and b_j ($i \neq j$) are adjacent child blocks. $AvgCS(B)$ is also between 0 and 1. If it is closer to 0, the content coherent degree of child blocks is lower. If it is closer to 1, the content coherent degree of child blocks is higher.

4.4 Segment Web Pages Using Visual Semantics

After the pruned DOM tree is obtained, the similar visual blocks can be recognized and the seam degree and content similarity of each block can be determined in advance. Here, we introduce the threshold α of $AvgSD(B)$ and the threshold β of $AvgCS(B)$. Empirically, we set α to be 0.9 and β to be 0.8. Our web page segmentation algorithm is a top-down method. It begins from the root node of the DOM tree and which is set to be the current node. The corresponding block of the current node will be judged according to the steps shown in Table 1. If the current node should be divided, then its child blocks will be judged as well. If the current node should not be divided, then it will be pushed into an array of segments and its child blocks will not be judged anymore. The detailed algorithm is shown in Fig. 4.

Table 1 Steps for judging a block.

Step 1	If the current block is a leaf block, then do not divide it. Otherwise go to Step 2.
Step 2	If the current block contains recurrent blocks, then divide it. Otherwise go to Step 3
Step 3	If the current block is one of recurrent blocks, then do not divide it. Otherwise go to Step 4
Step 4	If the current block contains only one child block, then divide it. Otherwise go to Step 5
Step 5	If the $AvgSD(B)$ of the current block is less than α, then divide it. Otherwise go to Step 6
Step 6	If the $AvgCS(B)$ of the current block is less than β, then divide it. Otherwise go to Step 7.
Step 7	If the area of the current block is greater than the half of client area, then divide it. Otherwise go to Step 8.
Step 8	If the current block does not satisfy all of the above conditions, then do not divide it.

```
Input: a DOM tree of a web page T
Output: a array of segments SegmentSet

Begin
1  SegmentSet = Φ
2  CurrentNode = T → Root
3  Segment(CurrentNode){
4     if CurrentNode is NOT divisible then
5        push CurrentNode into SegmentSet
6     end if
7     if CurrentNode is divisible then
8        ChildList = CurrentNode → Children
9        for each Child_i ∈ ChildList
10          Segment(Child_i)
11       end for
12    end if
13 }
14 return SegmentSet
End
```

Fig. 4 Algorithm for web page segmentation.

5. Experiments

5.1 Data Set

We randomly selected 50 query keywords from the chiebukuro category[†] of Yahoo Japan. We submitted the 50 queries to Yahoo Japan. For each of the 50 queries, we randomly collected 10 pages from the top-100 search results. As a result we collected 500 web pages. Since 81 pages are invalid or cannot be correctly displayed, we chose the remaining 419 pages as experiment data, in which 348 (83.0%) pages are written in HTML 5 and 71 (17.0%) pages are written in HTML 4. Moreover, 207 (49.4%) pages use javascript to generate HTML elements automatically. The

[†]http://chiebukuro.yahoo.co.jp/

(a) (b)

Fig. 5 (a) is an original page, (b) is visual segment result using our evaluation program.

419 pages are from 117 different web sites, thus the diversity of the data set can be guaranteed. The diversity can be used to test the robustness of the proposed method. In the 419 web pages there are 261 pages that contain similar visual blocks.

5.2 Evaluation Method

Different from other automatic evaluation experiments, the evaluation of web page segmentation is a human-involved task. A lot of previous work manually labeled the segments of web pages in advance and compared the labeled segments with the segment results using their method. However, labeling the segments is a time-consuming process. Therefore, we developed an evaluation program by using the APIs of Chrome Extension[†], which is a small piece of software program that can modify and enhance the functionality of the Chrome browser. The Chrome Extension can help the proposed method to obtain the visual information (e.g width, height, and coordinate) of blocks after the browser transforms an HTML file into a web page. Therefore, even if some HTML elements are generated by javascript, the information of these elements can also be obtained. This evaluation program can visualize the segmentation results as shown in Fig. 5.

The colored overlay rectangles represent the segment results. When a rectangle is clicked, the area and number of the clicked rectangle will be recorded by our program automatically.

Direct comparison of the proposed method with all of the related work describe in Sect. 2 is difficult, since the data set and evaluation program used are not open to the public. Therefore we implemented the VIPS [10] algorithm as the comparison baseline. VIPS is a popular page segment method that is often taken as a comparison baseline by other work. Our evaluation experiment contains two steps.

In the first step, we utilized our evaluation program to segment the 419 pages using VIPS and the proposed method respectively. The badly divided segments (bad segments) were then manually checked. For each page, the program recorded the area and number of both the checked segments and all segments. Given a page P, $S = \{s_1, s_2, \cdots, s_m\}$ is the set of segment results of P, and $S' = \{s'_1, s'_2, \cdots, s'_n\}$ is the set of bad segments of P. We used the area rate of bad segments (Bad Area Rate, BAR) and number rate of bad segments (Bad Number Rate, BNR) to evaluate the segment results of a page P. $BAR(P)$ and $BNR(P)$ can be calculated as in formula (9):

$$BAR(P) = \frac{\sum Area(s'_i)}{\sum Area(s_j)}$$
$$BNR(P) = \frac{n}{m} \quad (9)$$

where $Area(s_i)$ represents the area of segment s_i. The $BAR(P)$ and $BNR(P)$ are the less the better.

In the second step, we manually classified the segment results into 5 categories according to the results of the first step. The 5 categories and their descriptions are shown in Table 2. When we were classifying the segment results we considered not only the number of bad segments, but also the effect of bad segments. The effects are associated with the area of segments and the place where the segments are displayed. Image that, there are two bad segments, if one is at an inconspicuous place and the other one is at an important place, then their effects on the segment results are different.

[†]http://www.chromeextensions.org/

Table 2 Five categories of segment results.

Categories	Descriptions
Perfect	There are no bad segments.
Good	There are few bad segments, and these bad segments have little effect on segment results.
Fair	There are some bad segments, and these bad segments have an effect on segment results.
Bad	There are a lot bad segments, and the segment results are not acceptable.
Too Bad	Almost all the segments are bad segments.

Table 3 Average BAR and BNR of 419 pages and 261 pages.

	419 pages		261 pages	
	BAR	BNR	BAR	BNR
VIPS	34.2%	21.9%	32.2%	20.4%
Proposed method	1.65%	3.55%	0.990%	2.23%

5.3 Experiment Results

Our experiments are done on CORE i5 2.6 GHz, 4 GB, 12.1 inch, 1280*800 PCs. Table 3 presents the average BAR and BNR. In Table 3, the "419 pages" represents the experiment results of the total 419 pages, and the "261 pages" represents the experiment results of the 261 pages that contain similar visual blocks. From Table 3, the 21.9% segments of VIPS are bad segments, and the area of the bad segments accounts for 34.2% of the total area of all segments. Contrarily, only 3.55% segments of our method are bad segments, and their area accounts for only 1.65%. Moreover, in the results of VIPS, the BAR is greater BNR. It infers that the bad segments of VIPS have a large area. Contrarily, the bad segments of the proposed method have a small area. As mentioned above, we consider the smaller area that the bad segments have, the less effect they will have. Therefore, it is obvious that our method performances better than VIPS.

Because there are 261 pages containing recurrent blocks, we analyzed the segment results of these 261 pages

in much more detail. Similar to the results of 419 pages, our method has less bad segments and a smaller area of bad segments. It also should be noted that the average BAR and BNR changed little when the 261 pages were segmented by VIPS. Contrarily, the average BAR and BNR of our method decreased considerably. This is because VIPS doesn't consider the cases of similar visual blocks. It shows that our method can better segment the web pages that contain similar visual blocks.

Besides the analysis of bad segments, we also analyzed the distribution of all 419 web pages in 5 categories. As mentioned in the previous section, the 5 categories are perfect, good, fair, bad, and too bad. Their descriptions are shown in Table 2. Within the 5 categories, we consider that perfect is the most significant category. Let us see an example. Suppose there are two page segment approaches (A1 and A2), and they use the same data set (10 web pages). A1 can divide each page into ten segments, but every page has one bad segment. So, the average rate of bad segments is 10%. A2 can also divide each page into ten segments. Nine pages of them have no bad segments and the tenth page has ten bad segments. Obviously, the average rate of bad segments is also 10%. Based on the average rate of bad segments, A1 and A2 have the same performance. But in real applications, A2 may be the better choice. To make the rate of bad segments 0%, A1 has to be manually tuned for all the ten pages, while A2 just needs to be manually tuned for only one page. In other words, though A1 and A2 have the same rate of bad segments, A1 cannot perfectly segment any pages while A2 can perfectly segment nine pages. In this case, we consider A2 is better than A1. Table 4 presents the results of five categories. The perfectly segmented pages of VIPS account for only 29.8% while the perfectly segmented pages of our method account for 76.6%. It indicates that our method needs less manual intervention in order to make 100% perfectly segmented pages. If we consider both the perfect and good results are acceptable results, then VIPS has only 63.5% acceptable results while our method has 93.1% acceptable results.

We analyzed the 93.1% web pages and noticed that these are typical web pages which have different HTML tags (the tags of HTML 4 and HTML 5 are not totally the same), patterns, colors, fonts, and even different languages. Therefore the heuristic-based methods (e.g. VIPS) cannot divide all the web pages into appropriate segments. There are two reasons why the proposed method is suitable for these pages. The proposed method does not use the tags to divide a web page, therefore, even if a web page is written in HTML 5 or HTML 4, the segment result will not change. And, these pages still have some common visual features. First, a segment arranges its blocks in the same way and different segments have different arrangement. Second, a segment has the same type of contents and different segments have different types of contents. By formulating these visual features, the proposed method can be suitable for various web pages.

Table 4 Results of five categories.

	VIPS		Proposed method	
	Number	**Rate**	**Number**	**Rate**
Perfect	125	29.8%	321	76.6%
Good	141	33.7%	69	16.5%
Fair	88	21.0%	27	6.42%
Bad	45	10.7%	2	0.48%
Too bad	20	4.77%	0	0%

We also analyzed the "Bad" results and found they were due to "Text Only" web pages. In these pages, there are only text contents. In this case, the content similarity would be ineffective and the whole web page is just one segment. In many web applications these segment results are "Bad" segmentation. However, in some specific applications, these results can also be regarded to be acceptable. For example, the web segmentation is used for informative content extraction. This is because there is no uninformative content in the page. In this case, the whole web page can be considered as an informative content and this segment result will not reduce the accuracy of informative content extraction.

6. Conclusion and Future Work

Web page segmentation has a variety of benefits and potential web applications. However, early techniques of web page segmentation are mainly based on machine learning algorithms and rule-based heuristics, which cannot be used for large-scale page segmentation. In order to overcome these limitations, in this paper, we proposed a formulated page segmentation method using visual semantics. Instead of analyzing the visual cues of web pages, this method utilized the following three measures to formulate the visual semantics: layout tree was used to recognize the visual similar blocks; seam degree was used to describe how neatly the blocks are arranged; content similarity was used to describe the distance between the blocks. Web pages are first transformed into a DOM tree. After pruning the invisible nodes and meaningless text nodes, the proposed method judges the DOM tree nodes top-down based on the three measures. Compared with VIPS, the experiment results show that the proposed method can divide a web page into appropriate semantic segments with few bad segments. The perfectly segmented pages of the proposed method are also more than that of VIPS. Finally, we can conclude that the proposed method can effectively divide various web pages into appropriate segments.

In the future work, we will investigate how to improve the current measures of visual semantics and discover other meaningful visual semantics to segment web pages effectively and accurately.

References

[1] S. Baluja, "Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework," Proc. 15th Int'l Conf. World Wide Web, pp.33–42, Edinburgh, Scotland, May 2006.
[2] H. Ahmadi and J. Kong, "Efficient web browsing on small screens," Proc. Working Conf. Advanced Visual Interfaces, pp.23–30, Napoli,

Italy, May 2008.
[3] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Seeing the whole in parts: Text summarization for web browsing on handheld devices," Proc. 10th Int'l Conf. World Wide Web, pp.652–662, Hong Kong, May 2001.
[4] D. Chakrabarti, R. Kumar, and K. Punera, "A graph-theoretic approach to webpage segmentation," Proc. 17th Int'l Conf. World Wide Web, pp.377–386, Beijing, China, April 2008.
[5] R. Burget, "Layout based information extraction from html documents," Proc. 9th Int'l Conf. Document Analysis and Recognition, vol.2, pp.624–628, Curitiba, Parana, Brazil, Sept. 2007.
[6] R. Burget, "Automatic document structure detection for data integration," Business Information Systems, vol.4439, pp.391–397, April 2007.
[7] R. Burget and I. Rudolfova, "Web page element classification based visual features," Proc. 1st Asian Conf. on Intelligent Information and Database Systems, pp.67–72, Dong Hoi, Quang Binh, Vietnam, April 2009.
[8] Y. Borodin, J. Mahmud, I.V. Ramakrishnan, and A. Stent, "The hearsay non-visual web browser," Proc. 2007 Int'l Cross-Disciplinary Conf. Web Accessibility, pp.128–129, Banff, Canada, Aug. 2007.
[9] J.U. Mahmud, Y. Borodin, and I.V. Ramakrishnan, "Csurf: a context-driven non-visual web-browser," Proc. 16th Int'l Conf. World Wide Web, pp.31–40, Banff, Canada, May 2007.
[10] D. Cai, S. Yu, J. Wen, and W.G. Ma, "VIPS: a vision based page-segmentation algorithm," Technical Report MSR-TR-2003-79, Microsoft Research, 2003.
[11] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke, "Accordion summarization for end-game browsing on PDAs and cellular phones," Proc. SIGCHI Conf. on Human Factors in Computing Systems, pp.213–220, Washington, USA, March 2001.
[12] E. Kaasinen, M. Aaltonene, J. Kolari, S. Melakoski, and T. Laakko, "Two approaches to bringing internet services to WAP devices," Int. J. Comput. Telecommun. Netw., vol.33, pp.231–246, 2000.
[13] X. Liu, H. Lin, and Y. Tian, "Segmenting webpage with Gomory-Hu tree based clustering," J. Software, vol.6, no.12, pp.2421–2425, 2011.
[14] J. Kong, O. Barkol, R. Bergman, A. Pnueli, S. Schein, K. Zhang, and C. Zhao, "Web interface interpretation using graph grammars," IEEE Trans. Syst. Man. Cybern. C, vol.42, no.4, pp.590–602, July 2012.
[15] H. Sano, R.M.E. Swezey, S. Shiramatsu, T. Ozono, and T. Shintani, "A web page segmentation method by using headlines to web contents as separators and its evaluations," IJCSNS, vol.13, no.1, pp.1–6, Jan. 2013.
[16] J. Kang, J. Yang, and J. Choi, "Repetition-based web page segmentation by detecting tag patterns for small-screen devices," IEEE Trans. Consum. Electron., vol.56, no.2, pp.980–986, May 2010.
[17] C. Kohlschtter and W. Nejdl, "A densitometric approach to web page segmentation," Proc. 17th ACM Conf. Information and knowledge management, pp.341–344, Napa Valley, California, USA, Oct. 2008.
[18] J. Zeng, B. Flanagan, and S. Hirokawa, "Layout tree-base approach for identifying visual similar blocks from web pages," Proc. Int'l Symposium Web Engineering and Applications, pp.65–70, Niigata, Japan, June 2013.
[19] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," SIAM J. Comput, vol.18, no.6, pp.1245–1262, 1989.
[20] S. Amit, "Modern information retrieval: a brief overview," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, vol.24, no.4, pp.35–43, 2001.

Jun Zeng received the BS and MS degree in software engineering from Chongqing University in 2007 and 2010. He received Ph.D. degree in advanced information technology of Kyushu University in 2013. Since 2013, he has been a lecturer in the school of software engineering of Chongqing University. His research interest includes Web data extraction, data mining, and search engine.

Brendan J. Flanagan received a BS in Information Technology (Computing Studies) from RMIT University in 2010. Since 2013, has been a graduate student in the advanced information technology of Kyushu University. Research interests include: text mining, search engines, and e-learning.

Sachio Hirokawa received the BS and MS degree in Mathematics and PhD degree in Interdisciplinary Graduate School of Engineering Sciences from Kyushu University in 1977, 1979, and 1992. Since 1997, he has been a professor in research institute for information technology of Kyushu University. His research interest includes search engine, text mining, and computational logic.

Eisuke Ito received BS and MS degree in Engineering and PhD degree in Information Science from Kyushu University in 1992, 1994, and 1997. Since 2000, he has been an associate professor in research institute for information technology of Kyushu University. His research interest includes contents search, data mining, and information services.