A Web Page Segmentation Method by using Headlines to Web Contents as Separators and its Evaluations

Hiroyuki Sano†, Robin M. E. Swezey†, Shun Shiramatsu†, Tadachika Ozono†, and Toramatsu Shintani†

†Dept. of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology
Aichi, 466–8555 Japan

Summary
In this paper, we describe a Web page segmentation method based on title blocks and show its evaluation. Title blocks are minimum blocks that function as headlines for specific Web content. A typical Web page consists of multiple elements with different types of features, such as main content, navigation panels, copyright and privacy notices, and advertisements. Web page segmentation is the division of the page into visually and semantically cohesive pieces. Our segmentation method is comprised of three steps. First, it divides the page into minimum blocks. Second, it classifies the blocks into two classes, title blocks or non-title blocks. Third, it assembles groups of these blocks into Web content blocks. While the minimum blocks can play many roles, this study focused on blocks that are the titles of various Web content bits. A decision tree learning is used with nine features for each minimum block to extract title blocks from Web pages. Experimental results showed that our segmentation method could divide Web pages that are collected from the news site with 96.1 percent accuracy, independently of amount of content. The results also describes that the method can divide all Web pages that are used in the experiment in less than 1000 milliseconds.

Key words:
Web Page Segmentation, Semi-structured Data, Web Intelligence, Decision Tree Learning.

1. Introduction

A typical Web page consists of multiple elements with different functionalities, such as main content, navigation panels, copyright and privacy notices, and advertisements. Visitors to Web pages are only interested in the main content and have no use for the noisy content.

These days, Web page segmentation is an extensively studied topic. Web page segmentation is a process to divide a Web page into visually and semantically cohesive pieces. Web page segmentation clearly has a variety of benefits and potential Web applications, but if applications such as information retrieval or extraction treat all content on a Web page equally - e.g., with no differentiation between main and noisy content - there may be a decline in accuracy. It is necessary that such applications deal only with the main content of a Web page.

We proposed a new Web page segmentation method in the paper [1]. We focused that Web page designers appoint a headline to each Web content on a page, which we call "title blocks," to make for easy reading, and it is these blocks that the proposed algorithm focuses on. Title blocks can be used as separators when we segment a Web page. In the paper [1], we proposed the use of decision tree learning to extract the title blocks from Web pages. In this paper, we show the detail of the segmentation algorithm and its evaluations.

The methods proposed in the related works divide a Web page into very small pieces at first and then combine semantically cohesive pieces into Web content blocks. We propose a method for assembling the small pieces that focuses on pieces that function as headlines for specific bits of Web content.

2. Related Works

There are several different Web page segmentation algorithms already in place. The popular of them are DOM-based and layout-based.

2.1 DOM-based Method

In DOM-based segmentation, tag information is used to divide a Web page based on the Document Object Model (DOM). The DOM has a tree structure in which each node contains one of the components from an HTML tag.

Buyukkokten et al. [2] split a Web page using some relatively simple DOM nodes such as the <P>, <TABLE>, and <UL> nodes for further conversion or summarization.

Lin and Ho [3] only consider the <TABLE> node and its offspring to be a content block and use an entropy- based approach to determine the informative ones. Nodes such as <TABLE> and <P> are not only used for content organization but also for layout presentation. For example, consider the Web page layout in Fig.1. In this example, when the part marked "1" in Fig.1 (a) is a single content block, only "<TABLE>" needs to be extracted. On the other hand, when the part marked "2" is a single content block, we must extract and merge the "B" and "D" nodes of the tree. DOM-based segmentation cannot detect the "2" part as a single content block.

Manuscript received January 5, 2013
Manuscript revised January 20, 2013

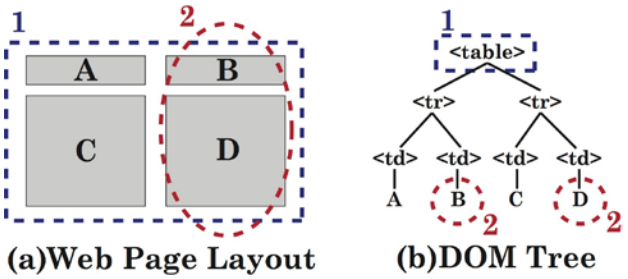


Fig. 1 Problem of DOM-based segmentation

2.2 Layout-based Method

The layout-based segmentation method uses layout information after rendering, assuming that similar content blocks are located close to each other and have similar shapes.

Cai et al. [4, 5] use layout information such as "font," "color," and "size" to restructure a Web page in a content block tree. Baluja [6] considers the Web page segmentation problem from the perspective of a machine-learning framework: he re-examines the task through the lens of entropy reduction and decision tree learning. However, a consideration of the layout differences that exist in the various parts of Web pages seems to be lacking in this layout-based segmentation method. Web pages have various layouts for various parts of themselves (associated pages, "headers," "footers," etc.), so it is difficult to adapt the same rules or parameters for all Web pages and all parts of a Web page.

3. Proposed Approach

The proposed Web page segmentation method is comprised of three steps: (1) dividing a Web page into minimum blocks, (2) classifying the blocks into title blocks or non-title blocks, and (3) combining the blocks into Web content chunks based on title blocks.

3.1 Minimum Blocks Extraction

The methods proposed in the related works mentioned above divide a Web page into very small pieces at first and then assemble them into semantically cohesive pieces of Web content. Our method also divides pages into very small pieces and combines them. These small pieces, which we call "minimum blocks," are divided on the basis of block-level elements defined by World Wide Web Consortium (W3C). All elements in a web page are classified as either block-level elements or inline elements. When rendered visually, a block-level element secures a rectangle space and displays child nodes in the same area. The overall Web page layout is determined by these block-level elements, each of which fit inside another. We define a minimum block as any block-level element that does not contain other block-level elements within it. If a node is an inline element that is the sibling of a minimum block, we adopt it as a minimum block even though it is technically an inline element. Therefore, all nodes that are rendered in a Web page are eventually assigned to minimum blocks.

The algorithm showed in Fig.2 is used to determine whether or not a node *n* is a block-level element. The algorithm is based on simple if-then rules.

```
procedure isBlockLevelNode(n)
BT ← ['p', 'blockquote', 'pre', 'div', 'noscript',
      'hr', 'address', 'fieldset', 'legend',
      'h1', 'h2', 'h3', 'h4', 'h5', 'h6',
      'ul', 'ol', 'li', 'dl', 'dt', 'dd', 'table', 'caption',
      'thead', 'tbody', 'colgroup', 'col', 'tr', 'th', 'td'];
1: if isValidNode(n) is false then
2:     return false;
3: else if displayStyle(n) is 'block' then
4:     return true;
5: else if tagName(n) ∈ BT then
6:     return true;
7: else
8:     return false;
9: end if
```

Fig. 2 The algorithm to judge that a node is block-level node or not

When a Web browser renders a Web page, some of the nodes are not displayed. We call such nodes "invalid". If a DOM node has the four qualifications listed below, however, we consider it "valid".

$$\begin{cases} width(n) \geq 1 \wedge height \geq 1 & (1) \\ top(n) + height(n) > 0 \wedge left(n) + width(n) > 0 & (2) \\ displayStyle(n) \neq \text{'none'} & (3) \\ visibilityStyle \neq \text{'hidden'} & (4) \end{cases}$$

To go into more detail, with (1), the size of the node means the value after multiplying the pixel width of the node by the pixel height. With (2), the coordinate of the node is expressed by rectangular coordinates, which treat the Web page as a plane. The origin is then at the upper left of the Web page. The x-axis has values that increase from left to right, while the y-axis has values that increase from top to bottom. With (3), the display style property specifies the type of box an element should generate. Nodes that have

been given “none” for the property are not displayed on a Web browser. With (4), the visibility style property defines whether or not the boxes generated by an element are displayed. Nodes that have been given “hidden” for the property are not displayed on a Web browser. However, an invisible box is still secured, and it affects the page layout.
Web designers can specify whether an element is inline or block-level by giving a display style property. An element whose tag is listed in *BT* in Fig.2 can be an inline element when it has an “inline” value for the display style. We consider the display style in the second rule (line: 3 in Fig.2) before we consider the tag in the third rule (line: 5 in Fig.2).

3.2 Classify Minimum Blocks into Title Blocks or Not

Our method focuses on title blocks to organize minimum blocks into Web content blocks. The second step of the segmentation method is to extract title blocks from a Web page.
As stated earlier, title blocks are minimum blocks that function as headlines for specific Web content. Web page designers assign a title block to each web content on a page to make for easy reading, and these title blocks can be used as separators to segment the different parts of a Web page.
In this study, we focused on the four title block characteristics listed below. (1) A title block has few child nodes. (2) A title block has a short text length. (3) The width of a title block is greater than its height. (4) The size of a title block is smaller than that of the block underneath it. These four characteristics, as well as characteristics based on HTML tag names, are used to create the features for machine learning. We adopted the nine features F_1 to F_9 calculated by formulas and procedures listed below.

$$F_1(n) = length(text(n)) + \sum_{ch \in children(n)} length(text(ch))$$

$$F_2(n) = \frac{\sum_{ch \in children(n) \wedge isTextNode(ch)} size(ch)}{size(n)}$$

$$F_3(n) = \frac{\sum_{ch \in children(n) \wedge isImgNode(ch)} size(ch)}{size(n)}$$

$$F_4(n) = \frac{height(n)}{width(n)}$$

$$F_5(n) = \begin{cases} 1 & if\ size(n) < size(below(n)) \\ 0 & else \end{cases}$$

$$F_6(n) = \begin{cases} 1 & if\ tagName(n) \in ['H1','H2','H3','H4','H5','H6','DT'] \\ 0 & else \end{cases}$$

procedure $F_7(n)$

1: $i \Leftarrow 0$;
2: **while** $tagName(below(n))$ is $tagName(n)$ **do**
3: $i++$;
4: $n \Leftarrow below(n)$;
5: **end while**
6: **return** i;

procedure $F_8(n)$

1: $j \Leftarrow 0$;
2: **while** $tagName(above(n))$ is $tagName(n)$ **do**
3: $j++$;
4: $n \Leftarrow above(n)$;
5: **end while**
6: **return** j;

$$F_9(n) = |children(n)|$$

We used decision tree learning to classify minimum blocks because it has only two possible classifications— “title block” or “not”—and because the learning algorithm can avoid over-fitting thanks to branch cut.
In [1], we proposed the use of decision tree learning to extract the title blocks from Web pages. We specified three classifiers for determining whether minimum blocks are title or non-title blocks, and measured the performance of the classifiers. These classifiers were constructed by decision tree learning using a J4.8 algorithm, decision tree learning using a random tree algorithm, and a support vector machine. We measured the accuracy of the title block extraction by ten cross-validations when training the classifiers.
The decision tree created by the J4.8 algorithm attained an 83.7% P_{tb} and a 92.9% R_{ntb}. These were the best results of the three algorithms, which indicate the J4.8 algorithm’s suitability for extracting title blocks for Web page segmentation.

3.3 Creating Combined Web Content Blocks based on Title Blocks

Fig.3 shows the algorithm to assemble minimum blocks into Web content blocks based on title blocks. The algorithm assembles an initial title block followed by consecutive non-title blocks below it.

procedure $makeContentBlocks(MB)$
$MB = \{mb_1, mb_2, \cdots, mb_n\}$
(minimum blocks in the Web page)

```
1: CB ⇐ {};
2: for all mb ∈ MB do
3:   if isTitleBlock(mb) then
4:     Container = {mb};
5:     left ⇐ ∞; top ⇐ top(mb);
6:     x ⇐ −1; y ⇐ −1;
7:     Q ⇐ {mb};
8:     while Q.length > 0 do
9:       b ⇐ shift(Q);
10:      if left > left(b) then
11:        left ⇐ left(b);
12:      end if
13:      if x ≥ left(b) + width(b) then
14:        x ⇐ left(b) + width(b);
15:      end if
16:      if y < top(b) + height(b) then
17:        y ⇐ top(b) + height(b);
18:      end if
19:      Belows ⇐ belowBlocks(b);
20:      Q ⇐ Q ∪ Belows;
21:      flag ⇐ false;
22:      for all b2 ∈ Belows do
23:        if isTitleBlock(b2) is true then
24:          flag ⇐ true;
25:          break;
26:        end if
27:      end for
28:      if flag is false then
29:        Container ⇐ Container ∪ Belows;
30:      else
31:        for all added ∈ Container do
32:          delete added from MB;
33:        end for
34:        for all mb2 ∈ MB do
35:          if inRect(left, top, x − left,
                 y − height, mb2) then
36:            Container ⇐ Container ∪ mb2;
37:            delete mb2 from MB;
38:          end if
39:        end for
40:        CB ⇐ CB ∪ {Container};
41:        break;
42:      end if
43:    end while
44:  end if
45: end for
46: return CB;
```

Fig. 3 The algorithm to assemble minimum blocks into Web content blocks by using title blocks

The argument for the procedure *makeContentBlocks* is an array. The array contains all minimum blocks in the Web page for segmentation.
The procedure *makeContentBlocks* returns an array that contains Web content blocks in the web page, and finishes all its processing.

4. Experimental Results

4.1 Segmentation Results by the Proposed Method

The four examples of the method executions are showed in Fig.4. (a) is a news page in Yahoo! News in Japan. (b) is a popular blog site (Ameba Blog) in Japan. (c) is the top page of amazon.co.jp. (d) is the top page of our laboratory Web site. Solid red lines show content blocks that were generated by the segmentation method. The segmentation was very successful on (a) and (b), successful on (c), failed on (d).

Fig. 4 Segmentation results by the proposed method

The proposed method cannot generate a content block if no title block exists. There is no title block in the upper right on (c), so the method failed to divide this region. In the upper right region on (d), there are four title blocks, but

the method does not succeed. This is because those four title blocks are populated with images. The training data that was used for the decision tree learning contains few title blocks with images. To get higher accuracy for title blocks with images, we require training data that features many title blocks with images.

4.2 Segmentation Results independent of amount of content

Segmentation methods proposed in previous research divide plural Web pages in a same Web site into different results. This is because those methods assemble minimum blocks into Web content by using amount of content. While amount of content in the blocks is less than threshold, they assemble the blocks. Thus, the segmentation results depend on the amount of content. Web pages in the same Web site present the same layout. They should be divided into equivalent results.

To tackle the problem, we proposed a method based on title blocks. The experiment was performed to show that the method divides Web pages of a same Web site into similar results.

We gathered 140 news web pages that were most viewed in Yahoo! News in Japan at 2 am, on 24th April 2012. We focused on the news article sections of the web pages. The news article sections are composed of six parts showed in Fig.5 (a) to (f): (a) is the title of the news. (b) is the time that the news was delivered. (c) is the body of the news. (d) is the list of related news. (e) is the last modified date. (f) is the logo of the news source.

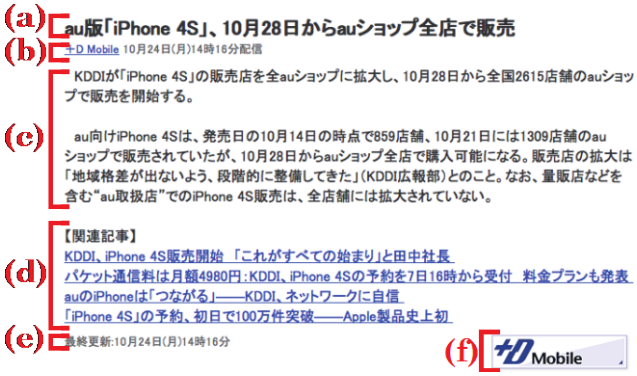


Fig. 5 The structure of news articles delivered in Yahoo! News in Japan.

We observed that our algorithm for assembling minimum blocks (showed in Fig.3) assembles these six parts into one content block, or not. The experiment is successful if all six parts are assembled into one content block. In the experiment, the following two patterns are equivalent to a failure: these six parts are assembled into more than two content blocks, or more parts in addition to these six are assembled into one content block.

The experiment shows that segmentation succeeds in 135 pages and fails in 5. The accuracy is 96.4%, thus showing practical usability.

4.3 Segmentation Processing Time

We conducted another experiment to show the processing speed of the proposed method. The segmentation processing time by the method were measured. The experiment was performed on the iMac (Mid 2010) that has Core i3 3.2GHz CPU and 8GB DDR3 SDRAM (PC3-8500). The Operating System running on the machine was Mac OS X 10.6.8 and the Web browser was Safari 5.1.5. One hundred Web pages were divided on the experiment environment.

We measured the change in the processing time in relation to the increase of the DOM nodes in Web pages. More specifically, we evaluated amounts of DOM nodes in the range of about 1,000 to 1,850. The Web pages that were used in the experiment were newest 100 news pages that were delivered on Yahoo! News in Japan at 11 am, on 23rd April 2012. Each Web page was divided 10 times by the method, and we measured the average processing time.

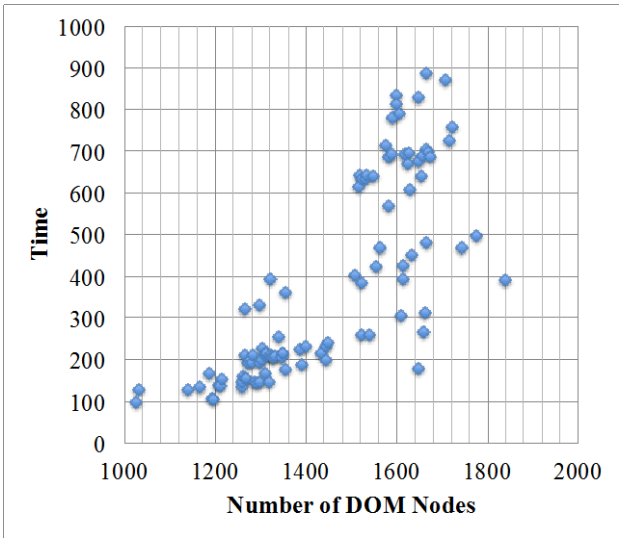


Fig. 6 Processing time of the proposed method

Fig.6 shows a scatter plot of the processing time for segmentation. The horizontal axis shows the number of DOM nodes in the web pages, while the vertical axis shows the processing time (milliseconds). All Web pages that were used in the experiment can be divided less than 1,000 milliseconds by the method. The longest processing time was 885 milliseconds when the number of DOM nodes in the case was 1,667.

Thus, the experiment shows that the proposed method can divide Web pages very quickly, and that the method has potential for practical use.

5. Conclusion

We proposed a Web page segmentation method is comprised of there steps: (1) dividing a Web page into minimum blocks, (2) classifying the blocks into title blocks or non-title blocks, and (3) combining the blocks into Web content bits based on title blocks.
A decision tree generated by the J4.8 algorithm was used for title blocks extraction in the research. Experimental results showed that the proposed method could divide Web pages that are collected from the news site with 96.1 percent accuracy, independently of the amount of content. The results also shows that the method can divide all Web pages that are used in the experiment in less than 1000 milliseconds.

Acknowledgments

This work was supported by KAKENHI (22500128) and the Hori Sciences & Arts Foundation in Japan.

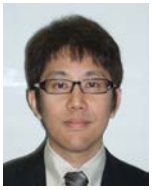
References

[1] H. Sano, S. Shiramatsu, T. Ozono and T. Shintani: A Web Page Segmentation Method based on Page Layouts and Title Blocks, International Journal of Computer Science and Network Security, Vol.11, No.10 pp.84-90 Oct. 2011.
[2] O. Buyukkokten, H. Garcia-Molina, and A. Paepcke: Accordion Summarization for End-game Browsing on PDAs and Cellular Phones, Proceedings of the Conference on Human Factors in Computing Systems, pp.213–220, 2001.
[3] S.-H. Lin and J.-M. Ho: Discovering Informative Content Blocks from Web Documents, Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp.588–593, 2002.
[4] D. Cai, S. Yu, J.-R. Wen and W.-Y. Ma: VIPS:AVision-based Page Segmentation Algorithm, 2003.
[5] D. Cai, S. Yu, J.-R. Wen and Wei-Ying Ma: Extracting Content Structure for Web Pages based on Visual Representation, The 5th Asia-Pacific Web Conference on Web Technologies and Applications, pp.406–417, 2003.
[6] S. Baluja: Browsing on Small Screens: Recasting Web-page Segmentation into an Efficient Machine Learning Framework. In Proceedings of the 15th international conference on World Wide Web, WWW '06, pp.33–42, 2006.
[7] P. Baudisch, X. Xie, C. Wang, and W.-Y. Ma.: Collapse-to- zoom: viewing web pages on small screen devices by interactively removing irrelevant content. In Proceedings of the 17th annual ACM symposium on User interface software and technology, UIST '04, pp.91–94, New York, NY, USA, 2004. ACM.
[8] Y. Chen, W.-Y. Ma and H.-J. Zhang: Detecting Web Page Structure for Adaptive Viewing on Small Form Factor Devices, The 12th international conference on World Wide Web, pp.225–233, 2003.

Hiroyuki Sano is currently a PhD student at Nagoya Institute of Technology, Japan. He received his BS in computer science from Nagoya Institute of Technology, Japan, in 2008 and his MS in 2010. His focus is on intelligent Web technologies.

Robin M. E. Swezey is currently a PhD student at Nagoya Institute of Technology, Japan. He received his Engineering and MS degree in computer science from the French School of Electronics and Computer Science (EFREI) in 2009. Previously, from 2008 to 2009, he did research on Multi-Agent Systems at Nagoya Institute of Technology as a special research student. His focus is on Data Mining, Machine Learning, Natural Language Processing and the Semantic Web applied to real-world problems.

Shun Shiramatsu received his PhD in information science from Kyoto University, Japan, in 2009, and his MS from Tokyo University of Science in 2003. He is currently (2012) an Assistant Professor of Computer Science. His research interests include discussion support and conversation modeling.



Tadachika Ozono received his MS and PhD in computer science from Nagoya Institute of Technology of Nagoya City, Japan, in 1998 and 2000, respectively, and is currently an Associate Professor of Computer Science there. His main research interest (2012) is Web intelligence.

Toramatsu Shintani received his MS in industrial engineering and his PhD in computer science from Tokyo University of Science in 1982 and 1993, respectively. He was a research staff member in Fujitsu Limited from 1982 to 1993. He is currently (2012) a Professor of Computer Science at Nagoya Institute of Technology of Nagoya City, Japan.
His current research interests include decision support systems and Web intelligence.