Block-o-Matic: a Web Page Segmentation Tool and its Evaluation

Andrés Sanoja, Stéphane Gançarski

Block-o-Matic: a Web Page Segmentation Tool and its Evaluation[1]

Andrés Sanoja, Stéphane Gançarski

Laboratoire d'Informatique Paris 6 (LIP6)

1 Introduction

Nowadays, Web page segmentation is an extensively studied topic. Web page segmentation refers to the process of dividing a Web page into visually and semantically coherent segments called blocks. Web page segmentation has a variety of benefits and potential Web applications, in mobile web, voice browsers, web page phishing, duplicate detection, information retrieval, information extraction, user interest detection, evaluating visual quality (aesthetics), caching, archiving, publishing dynamic content, semantic annotation, web accessibility, web page classification and clustering. For instance remove noisy content for visualizing web pages in devices with small screens or blocking part of a web page with inappropriate content for some audiences instead of blocking the whole page.

There are different approaches to web page segmentation, most of them described in [5]: structure, layout, hybrid, image, fixed-length and text based approach. The structure based approach references the uses only of the HTML tags as they are in the web page source code or the DOM interface of the web page after rendering in a web browser. The layout based approach focuses on the visual properties of the page. The hybrid approach uses both the structure and the layout of the page to create a hierarchy of blocks through block extraction and recursive refinement. The image based approach takes an image of a web page and apply image processing techniques to detect blocks. The fixed-length based approach removes all the semantic information (tags) from the page and then uses fixed-length algorithms to segment the web page. The text based approach retrieve segments from web pages based on the properties of text such as passage similarity, clustering, among others.

We think that the hybrid approach allows having a good perception of the layout of the web page. Because allows to analyze the document from the structural perspective and also evaluates the visual and geometric aspects of the page. However, this aspects are not enough the have a good description of the page. Blocks should be related to some role into the layout, in order to give more information to applications that depends on web page segmentation, for instance the layout flow. To accomplish this, the logic structure of the web page should be taken into account, along with the content and the geometric structure as used in [1].

We propose a tool for web page segmentation: Block-o-Matic. It is based on an original model which combines two popular approaches for web page and document analysis: the Vision-based model and the Geometric Layout Model [1, 4]. A web page is processed to build three structures: content, geometric and logical structure, each structure giving different information about the page document. The content structure distinguishes the HTML elements used to hold content and which serve for organizing. The geometric structure allows representing the page objects based on the presentation. It allows also adjusting the block shape to the content geometry. The logical structure describes the connections between blocks based on the page layout, on blocks geometric properties and on content elements properties [2]. The outcome of the processing is a segmented web page, which is a consolidated view of the three structures above mentioned

In this context, one important challenge is the evaluation of the segmentation algorithm. In order to

1 This work was partially supported by the SCAPE Project. The SCAPE project is co-funded by the European Union under FP7 ICT-2009.4.1 (Grant Agreement number 270137)

compare one algorithm with another, it is crucial to have a initial evaluation, validated by human assessors, to check its correctness. Evaluators task is to indicate which parts of the web page represent semantically coherent segments, and at the same time to have a semantic partition of the page. This given information must be then translated in order to identify to which HTML elements correspond those segments. This process is tedious and error prone. Having a ground truth database, ready to use, can take several days or months depending on the collection size. To handle these challenges, we propose a new tool for the pagelyzer suite [6], Block-o-Manual which allows performing the web page segmentation manually. It is intended to be a tool to help assessors discover and segmenting web pages manually, in the more efficient way possible and reduce the intervention of analyst in the process. Altogether allow us to evaluate its automatic contra-part: block-o-matic segmentation tool (in the same suite).

Ruby 1.9.2 was used as programming language for implementing the Block-o-Matic tool, Nokogiri libraries are used for HTML/XML manipulation. Block-o-Manual is provided as a Firefox extension. It is implemented using the Mozilla Add-on Builder and pure Javascript and its backend in PHP5. There are several similar tools for annotating web pages; however they are oriented to text content annotation, notes, highlighting and text formatting. In our case we want in reveal those structural elements used to organize the content of a web page: blocks.

In the rest of this paper, we first present the Block-o-Matic segmentation model, the prototype architecture and finally, we introduce the demonstration scenario.

2 Block-o-Matic Segmentation Model

We extend the concepts of [4], a model for segmenting digitized document in the optical character recognition domain, and adapt it to web pages. The content structure is the DOM interface after rendering a page in a browser. The geometric structure describes a page with respect to the geometric properties of content elements. The logical structure describes the connections of blocks at a layout level. For example if a logical block is found at top of page and the next in the bottom, that suggests a vertical layout flow. As shown in Figure 1, the segmentation process of a page (W) is divided into three phases: page analysis, page understanding and page reconstruction. The extraction of the geometric structure (W_{geom}) from the content structure (W_{cont}) is called web page analysis (c2g function). Mapping the geometric structure (W_{geom}) into a logical structure (W_{log}) is called document understanding (g2l function). Rebuilding the web page from the logic (W_{log}), geometric (W_{geo}) and content structure (W_{cont}) is called web page reconstruction (Rec function) where, $W' = Rec(W_{cont}, c2g(W_{cont}), g2l(c2g(W_{cont})))$. The c2g function constructs the geometric structure from the content structure. It organizes the content in function of the tag as: content, container, content-container or default elements. Additionally, it checks that there are not overlapping between siblings blocks. The g2l function constructs the logic structure from geometric structure. It assigns labels to geometrics blocks depending on their position in the web page, the relative area they occupy and the visual cues of their content elements. The labels are: left, right, top, bottom, middle or in other case standard. The Rec function builds the page (W') which is the original document (W) enriched with all structure, into an linear document in a XML-like style.

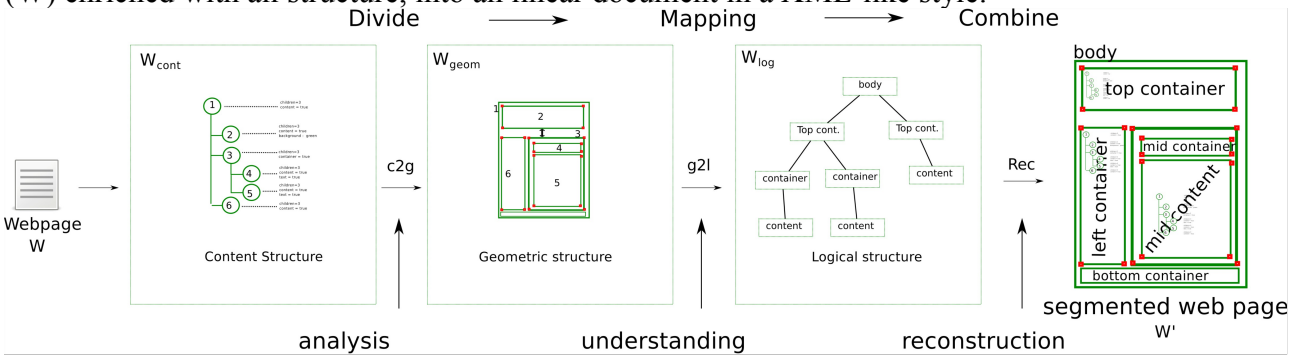


Figure 1. Web page segmentation process

3 Prototype Architecture

The prototype in question comprise the Block-o-matic component and the Block-o-manual component directly in conjunction with the others components in the pagelyzer suite. At a glance this suite is conformed by three main components: *capture*, *analyzer, comparator* and *changedetection*. The *capture* component handles the task of having the rendered HTML copy of a web page integrated with the visual cues and a screenshot. The *analyzer* component implements the block-o-matic segmentation algorithm. The comparator component compares two segmentation results and gives a score. The *changedetection* component compares two web pages versions and decides if they are similar or not. This decision is based on a combination of structural and visual comparison methods embedded in a statistical discriminative model, a visual similarity measure designed for Web pages that improve change detection and a supervised feature selection method adapted to Web archiving.

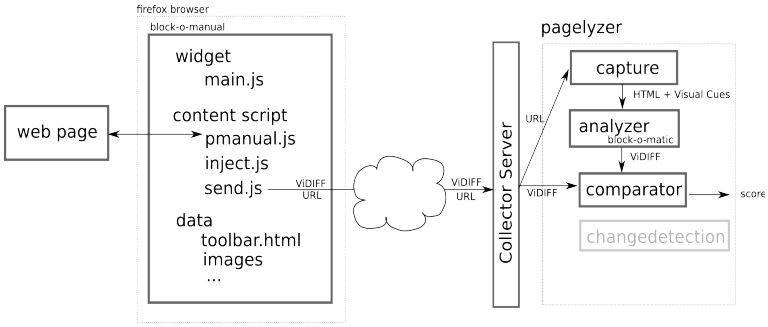


Figure 1: Prototype architecture

Block-o-manual has been developed on top of the Mozilla Add-On SDK. It relies on XUL and Javascript technologies. Using Add-on SDK, the development of this browser extension is done through the design and implementation of a set of components: widgets, content scripts and general data. Widgets components represent functional elements integrated to browser environment (menus, toolbars, etc). Content scripts are javascript code that interacts with the current web page document. General data components are images, web documents or other needed files. Figure 1 shows the prototype architecture.

Both manual and automatic segmentation should take the same kind of decisions for similar situations. The main idea is that when an assessor selects an element in the page give him/her the same options as the automatic segmentation would do. For that matter we use the same rules in both tools. Block-o-matic traverse the DOM tree from the root node to the leaves, making decisions in the way, however Block-o-manual depends on user input and it is evaluated one element at a time, as a response for that input. To be more precise in the kind of decisions both tools should make, let us describe the most relevant rules that should be followed to assure parity in the comparison.

Rule 1: Only Line-break elements can become blocks.
Inspired by the HTML specification[2] the block-o-matic segmentation model proposes to organize HTML elements as: inline and line-break elements:

- Inline elements: elements that use only the space of its content. For example tags <*a*>, <*b*> or <*i*>
- Line-break elements: elements that organize content and reserve a rectangular space of a web page for rendered content. Only elements that are valid (non terminal and visible) are included. An element is visible if its visibility style property value is either “block” or “inline” and its area is not zero. The line break elements are classified into the three following categories:
  - Container elements. Some elements are used generally for organizing content. That is the case, for example, of <*div*> and <*p*> tags.
  - Content elements are the elements used to hold enriched text. For example tags such as <*li*> or <*h1*>.

2 http://www.w3.org/TR/html4/

- Content-Container elements are elements that, depending on their-self and their children configuration belong to both categories. For example a <*p*> tag with text and inline elements as children.

Rule 2: If the current element has only one line break child, this latter replaces its ancestor
Rule 3: If an ancestor in the current element is in any of the line-break categories, it is proposed as a block.
Rule 4: If the current element is selected as a block and there are sibling elements which are not, those latter are automatically added as block to complete the partition.

Other rules are added to cover the entire segmentation model. Nevertheless, these are suggestions for the assessor and (s)he may decide to use them or not.

The next step is that the user is asked to indicate the semantic order of blocks within the page (flow of the layout). With block and layout flow done, a ViXML [3] document is created and sent to the collector server in conjunction with the web page URL. Thereupon, the *capture* component receives the URL and gets the rendered source code, the visual cues and the screenshot of the web page. Next, the *analyzer* component is requested to segment the page, producing another ViXML document. The comparator component is invoked, having as arguments the two documents produced (manual and automatic), and it gives a score and two images for visual comparison, one with the result of manual segmentation and the other with the automatic one.

4 Demonstration Scenario

In this section, we describe our demonstration scenario by giving the inputs and outputs for the evaluation of the segmentation process. The given scenario is to evaluate the segmentation of current web pages versions, at a coarse level of granularity, for example: the header, the footer, a main navigation or the main content. It is presented in two steps:

Step 1. Manual segmentation
We ask four assessors to one category each of a set of 100 web pages from the StumbleUpon[3] collaborative filtering services. They are distributed in sets of 25 pages per category. The categories are: Art/History, Computers, Sports and Sci/Tech. These categories are already defined and pages are associated already to them. Figure 2 shows an example of a web page in the Art/History category. In figure 3 can be seen that two blocks are being selected. In red the block clicked and in green the sibling element in order to have a complete partition. Figure 4 shows the final segmentation, composed of three blocks: one at the header and two parallel content blocks. Figure 5 depict the layout flow indicated. In this case the user makes click over all blocks; the click sequence is taken as the orientation of the flow of the layout.
Step 2. Automatic Segmentation and Comparison
After the manual segmentation is done the automatic segmentation starts the capture of the web page. The rendered HTML source codes in addition to the visual cues are then passed to the *analyzer* component to perform the segmentation. The outcome is another ViXML. Both documents are then passed to the *comparator* component as parameter and it returns a score and the two images, above mentioned. For instance Figure 6 shows that the manual and automatic segmentation are the similar but the automatic segmentation did not select the expected, the one that represents the footer of the page and also missed one block

3 http://www.stumbleupon.com

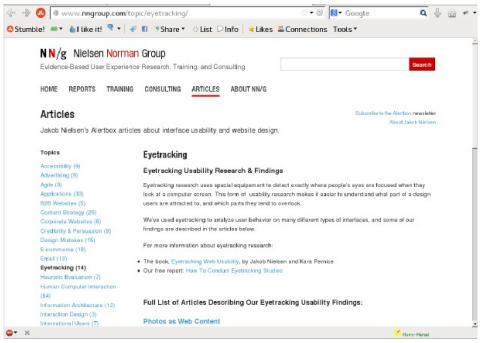


Figure 2. Computers category example page

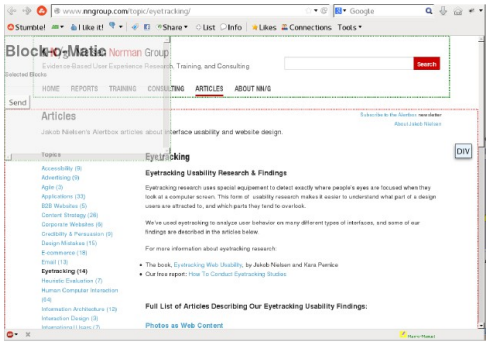


Figure 3. block selection example

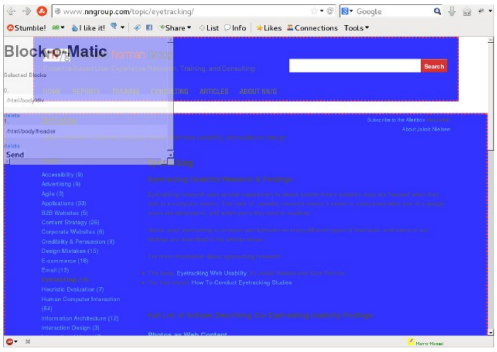


Figure 4. sample web page final segmentation

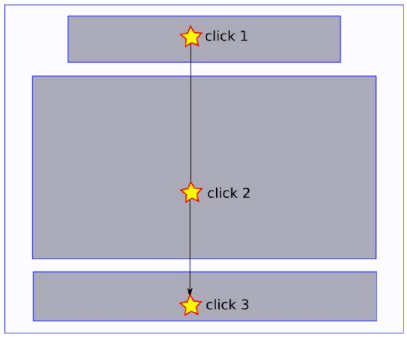
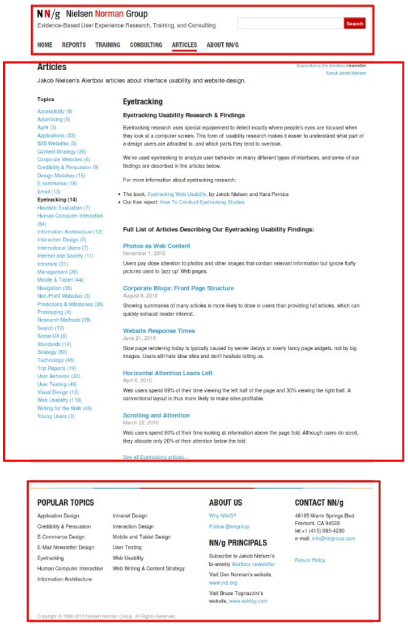
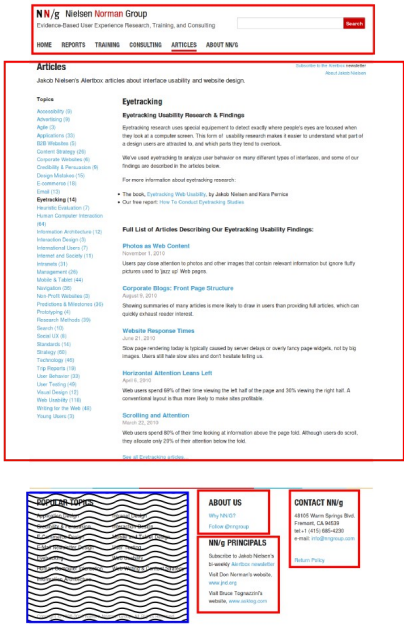


Figure 5. sample web page layout flow



a) Manual segmentation



b) automatic segmentation

Figure 6. manual and automatic segmentation comparison

4 References

[1] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Extracting content structure for web pages based on visual representation. In Fifth Asia Pacific Web Conference (APWeb'03), 2003.

[2] Jisheng Liang, Ihsin T. Phillips, and Robert M. Haralick. Performance evaluation of document structure extraction algorithms. Computer Vision and Image Understanding, 84(1):144-159, 2001.

[3] Zeynep Pehlivan, Myriam Ben-Saad, and Stéphane Gançarski. Vi-diff: understanding web pages changes. In Proceedings of the 21st international conference on Database and expert systems applications: Part I, DEXA'10, pages 1-15, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Y.Y. Tang, C. Y. Suen, C. D. Yan, and M Cheriet. Document analysis and understanding: a brief survey. First International Conference on Document Analysis and Recognition, pages 17-31, 1991.

[5] Yeliz Yesilada. Web page segmentation: A review. Technical report, Middle East Technical University Northern Cyprus Campus, March 2011.

[6] https://github.com/openplanets/pagelyzer