COMPUTER
ENGINEERING
PROGRAM

Middle East Technical
University
Northern Cyprus
Campus

eMINE Technical Report Deliverable 2 (D2),

January 2012

Vision Based Page Segmentation: Extended and Improved Algorithm

Elgin Akpınar[1] **and Yeliz Yesilada**[2]
[1]Department of Computer Engineering
Middle East Technical University,
Ankara, Turkey

[1]Middle East Technical University,
Northern Cyprus Campus,
Kalkanlı, Güzelyurt, TRNC,
Mersin 10, TURKEY

Web pages consist of different segments, serving different purposes. Most common types of these segments are header, right or left columns and main content. Moreover, these parts may include several subparts, as a news web page may contain more than one article in a page. In order to detect different segments in a web page, we first need to construct its block structure, and using visual cues is a very useful practice in this process. Being one of the most popular algorithms for this purpose, Vision Based Segmentation Algorithm (VIPS Algorithm) needs some improvement in its most important part, visual block extraction. We defined some additional terms and detected visual cues for extending visual block extraction part. In this technical report, deficiencies of VIPS algorithm are explained, and new rules are defined. Moreover, our implementation of VIPS algorithm is introduced.

METU
NCC

eMINE

The World Wide Web (web) has moved from the Desktop and now is ubiquitous. It can be accessed by a small device while the user is mobile or it can be accessed in audio if the user cannot see the content, for instance visually disabled users who use screen readers. However, since web pages are mainly designed for visual interaction; it is almost impossible to access them in alternative forms. Our overarching goal is to improve the user experience in such constrained environments by using a novel application of eye tracking technology. In brief, by relating scanpaths to the underlying source code of web pages, we aim to transcode web pages such that they are easier to access in constrained environments.

Contents

Middle East Technical University,
Northern Cyprus Campus,
Kalkanlı, Güzelyurt, TRNC,
Mersin 10, TURKEY

Tel: +90 (392) 661 2000
http://www.ncc.metu.edu.tr/

Corresponding author:
Elgin Akpınar
Tel: +90 (555) 564 5427
elgin.akpinar@metu.edu.tr

1 Introduction

Vision Based Segmentation Algorithm (VIPS Algorithm) aims to extract the block structure by using some visual cues and tag properties of the nodes. Visual Block Extraction is the basic and the most important part of this algorithm. Therefore, this process should be handled in care and detail so that all visual cues for block extracting are used. [1] gives some definitions and rules to find blocks in a web page. However, there are some missing definitions and ambiguities in the algorithm [1].

In this report, we explain how we extended and improved the VIPS Algorithm. We particularly show how we added ssome definitions and visual cues to Visual Block Extraction step and redesigned this part of the algorithm.

2 VIPS: A VIsion based Page Segmentation Algorithm

VIPS Algorithm aims to find visual blocks in a web page in three main steps which are: (1) visual block extraction, (2) visual block separation and (3) content structure construction. The most important one of these three steps are visual block extraction, since the other two of the steps depend on the first step. If the block structure of a web page is constructed false, the separators and content structure are constructed false, too. Therefore, visual block separation needs more attention, since it is the root of the algorithm.

Based on WWW HTML Specification 4.01[1], [1] defines the following definitions with respect to the visual properties of different nodes.

Inline node the DOM node with inline text HTML tags, which effect the appearance of text and can be applied to a string of characters without introducing line break. Such tags include <B>, <BIG>, <EM>, <FONT>, <I>, <STRONG>, <U>, etc.

Line-break Node the node with tag other than inline text tags.

Valid node a node that can be seen through the browser. The width and height of the node are not equal to zero.

Text node the DOM node corresponding to free text, which does not have a HTML tag.

Virtual text node (recursive definition):

- Inline node, with only text node children, is a virtual text node.
- Inline node, with only text node and virtual text node children, is a virtual text node.

Using above definitions, [1] developed a recursive approach for block extraction by using both tag properties and visual properties of the nodes.

The visual cues used in [1] are tag, color, text and size cues. For tag and text cues, above definitions are used. For color cue, background color is the main criteria. Finally, for size cue, the size of the nodes are compared to a threshold, comparing the node size to the whole page or a part of the page. Note that, this threshold is undefined.

[1] `http://www.w3.org/TR/html4/`

The aim is to reach to a predefined pDoC value, by comparing it to the DoC value obtained in each turn in the algorithm. If it finds a DoC value greater than pDoC value, it stops executing the rules defined. These rules are listed as follow:

1. If the DOM node is not a text node and it has no valid children, then this node cannot be divided and will be cut.
2. If the DOM node has only one valid child and the child is not a text node, then divide this node.
3. If the DOM node is the root node of the sub-DOM tree (corresponding to the block), and there is only one sub DOM tree corresponding to this block, divide this node.
4. If all of the child nodes of the DOM node are text nodes or virtual text nodes, do not divide the node. If the font size and font weight of all these child nodes are same, set the DoC of the extracted block to 10. Otherwise, set the DoC of this extracted block to 9.
5. If one of the child nodes of the DOM node is line-break node, then divide this DOM node.
6. If one of the child nodes of the DOM node has HTML tag <HR>, then divide this DOM node
7. If the background color of this node is different from one of its children's, divide this node and at the same time, the child node with different background color will not be divided in this round. Set the DoC value (6-8) for the child node based on the html tag of the child node and the size of the child node.
8. If the node has at least one text node child or at least one virtual text node child, and the node's relative size is smaller than a threshold, then the node cannot be divided Set the DoC value (from 5-8) based on the html tag of the node
9. If the child of the node with maximum size is small than a threshold (relative size), do not divide this node. Set the DoC based on the html tag and size of this node.
10. If previous sibling node has not been divided, do not divide this node
11. Divide this node.
12. Do not divide this node. Set the DoC value based on the html tag and size of this node.

After these rules, [1] lists some tag types and defines which rules to apply for which tag types. The list contains six tag types, which are inline text nodes, TABLE, TR, TD, P and other tags. However, this list is so narrow that, it does not cover neither all HTML tags nor any of HTML5[2] tags. Other tags than TABLE, TR, TD and P have different properties and behaviours in a web page so that they should not be used in the same category.

[2] `http://dev.w3.org/html5/spec/`

Another missing information in [1] is the precedence of this rules. The idea which accepts that the precedence of the rules are related with the order of the rules conflict with the example given at the end of section 4.2 in [1]. Moreover, in rules 9th and 10th, relative size of the node is compared to a threshold. This threshold is undefined and it is not possible to find a cue about how to calculate this threshold, although it is stated that this is a predefined value.

In conclusion, the algorithm in [1] is very limited and should be clarified. Moreover, it does not use many properties which can be used to state different topics in a web page.

3 Extended Visual Block Extraction

Although [1] divides tags into three as invalid nodes, inline nodes and line break nodes, we need more detailed specification for this purpose, since some tags in the same category in [1] might have different behaviors in a web page. For example, a P tag would have spaces around more than a DIV tag. Moreover, if we use both of these tags without any child, P tag will display empty spaces while DIV tag will not appear in the page in any format. Therefore, P tag might be used as a separator between two blocks while DIV tag is useless for this purpose. Due to this fact, we need to define different categories for such tags. Note that, many popular tags like DIV are not mentioned in [1].

Inline Node Nodes, which does not cause to a new line in a page, are inline nodes. A, STRONG, BIG, EM, etc. are inline nodes. These nodes are used only for changing the appearance of the text.

Line-break Nodes Line-break nodes cause to a new line in a page. DIV, P, TABLE, TR, UL, etc. are line-break nodes.

Invalid Nodes Invalid nodes are those does not appear visually in the page, such as PARAM, SCRIPT, STYLE, TITLE, <!–...–>, !DOCTYPE, etc. These tags effect the other elements of the page instead of appearing in the page layout.

Partially Valid Nodes These nodes effect the page layout only if they have a text node or any other visible children. Partially Valid Nodes consist of inline nodes and some of the line-break nodes such as FORM, HEAD, TABLE, DD, DT, etc. Main idea is that if a node is invisible without any child, then this node is partially valid node.

Valid Nodes Valid nodes are those appear in the page layout. They consist of partially valid nodes with visible children and the remaining of the line-break nodes that does not included in partially valid nodes. These line-break nodes appear in the page layout, even if they do not have any valid child. Nodes like UL, LI, DL, BLOCKQUOTE, etc. appear in the page as empty spaces if they do not have children. These nodes, moreover, have higher height than partially valid line-break nodes and can be used for separating different topics.

Font Size & Font Weight Although H1 to H6 tags are used to visualize the headings, some developers may use other tags with different font size and font weight. For this reason, by obtaining different font size and font weight, we can find where a new topic starts.

Margin Nonzero margin might be used to separate different blocks by putting some space between them.

Float Different float values can be used to group nodes into the same block.

Images Images are useful when finding headers since headers may include a logo.

By using the above definitions and visual cues, we defined the following rules for visual block extraction.

1. Remove all the invalid nodes and partially valid nodes which does not have any child. Note that, valid nodes without any child must be kept, since, although they do not have a valid content, they appear in the page layout and might be used as a separator between two different topics.
2. If a node has only text node or invalid children, no block extracted.
3. If node has only one child,
   (a) If child is a text node or virtual text node, then no block extracted.
   (b) If child is line-break node, then same rules are applied to the child node.
4. If all the children are virtual text nodes of a node, node is put into block pool and we do not divide this rule in next turns.
5. If a node contains a child whose tag is HR, BR or any of valid nodes which has no children, then the node is divided into two as the nodes before the separator and after the separator. For each side of the separator, two new blocks are created and children nodes are put under these blocks. Note that, separator does not extract a block under the main block, it just serves to extract two blocks which other nodes are put into.
6. If node is a table and some of its columns have different background color than the others, divide the table into the number saparate columns and construct a block for each piece.
7. If one of the child node has bigger font size than its previous sibling, divide node into two blocks. Put the nodes before the child node with bigger font size into the first block, and put the remaining nodes to the second block.
8. If the first child of the node has bigger font size than the remaining children, extract two blocks, one of which is the first child with bigger font size, and the other contains remaining children.
9. If a node has some valid nodes and some partially valid nodes in its children, divide this node. Each valid node will be a separate block, and each partially valid node next to each other create a block as a whole.
10. If node has at least one child with float value ”left” or ”right”, create three blocks. For each children,
    (a) If child is left float, put it into the first block.

(b) If child is right float, put it into the second block.

(c) If child is not both left and right float, put it into the third block. If first block or second block have children, create new blocks for them. Also, create a new block for the child without float.

11. If a node has a child, whose at least one of margin-top and margin-bottom values are nonzero, divide this node into two blocks. Put the sibling nodes before the node with nonzero margin into the first block and put the siblings after the node with nonzero margin into the second block.

    (a) If child has only nonzero margin-top, put the child into second block.

    (b) If child has only nonzero margin-bottom, put the child into first block.

    (c) If child has both nonzero margin-top and nonzero margin-bottom, create a third block and put it between two blocks.

12. If a node has a line-break child containing an image, divide node into two blocks. Put the nodes before the child with image into the first block, and put the remaining children into the second block.

13. If the first child of the node contains an image, put this child as a block, and create a new block for the remaining children.

4 Implementation

Extended VIPS Algorithm has been developed by using segmentation technologies on ACTF[3]. ACTF is a framework and extensible infrastructure, which enables developers to build a variety of utilities for the purpose of improving the accessibility of applications and content for people with disabilities.

Access to the properties of the document elements in a web page is provided by IWebBrowser interface, which extends ImodelService. Moreover, these interfaces enables to control the web browser, i.e., such as changing textual content of an element. After DOM structure is constructed and a variety of properties, including style and attributes of the elements, are set for each element, visual block structure is extracted according to the rules which elements match. At the end of the process, each visual block is listed in an hierarchical structure.

5 Summary

Web pages consist of different segments serving for different purposes. In order to automatically identify these segments, visual cues provide significant amount of information that can be used. Different algorithms have been proposed [3, 2]. One of the most popular algorithms among these algorithms, is Vision Based Page Segmentation (VIPS) algorithm. However, VIPS Algorithm includes some deficiencies in its most important part, visual block extraction. We have defined more clear and extended rules for visual block

[3] http://www.eclipse.org/actf/

extraction. In these rules, new definitions and terminologies are used. Finally, we had an implementation, which is introduced in this technical report.

References

[1] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. Vips: a vision based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft Research, 2003.

[2] Yeliz Yesilada. Heuristics for visual elements of web pages. Technical report, University of Manchester and Middle East Technical University Northern Cyprus Campus, 2011.

[3] Yeliz Yesilada. Web page segmentation: A review. Technical report, University of Manchester and Middle East Technical University Northern Cyprus Campus, 2011.