Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Box clustering segmentation: A new method for vision-based web page preprocessing

Jan Zeleny[*], Radek Burget, Jaroslav Zendulka

Brno University of Technology, Faculty of Information Technology, Centre of Excellence IT4Innovations, Bozetechova 2, 61266 Brno, Czech Republic

ARTICLE INFO

Article history:
Received 5 May 2016
Revised 1 December 2016
Accepted 2 February 2017
Available online 16 February 2017

Keywords:
Clustering
Segmentation
Vision-based page segmentation
VIPS

ABSTRACT

This paper presents a novel approach to web page segmentation, which is one of substantial preprocessing steps when mining data from web documents. Most of the current segmentation methods are based on algorithms that work on a tree representation of web pages (DOM tree or a hierarchical rendering model) and produce another tree structure as an output.

In contrast, our method uses a rendering engine to get an image of the web page, takes the smallest rendered elements of that image, performs clustering using a custom algorithm and produces a flat set of segments of a given granularity. For the clustering metrics, we use purely visual properties only: the distance of elements and their visual similarity.

We experimentally evaluate the properties of our algorithm by processing 2400 web pages. On this set of web pages, we prove that our algorithm is almost 90% faster than the reference algorithm. We also show that our algorithm accuracy is between 47% and 133% of the reference algorithm accuracy with indirect correlation of our algorithm's accuracy to the depth of inspected page structure. In our experiments, we also demonstrate the advantages of producing a flat segmentation structure instead of an hierarchy.

1. Introduction

Web page segmentation presents one of substantial preprocessing steps for data mining from web documents. There has been a lot of development in the area of web page partitioning. While some of the designed methods are targeted at specific problems like cleaning the noise from the web page, others, including page segmentation, are more generic in terms of possible utilization of their results. The problem with most of the segmentation algorithms is that they are quite slow, they depend on implementation details of the documents they process and they produce a hierarchical output that is difficult to process.

The objective this paper pursues is the development of a new web page segmentation method that is purely vision-based, independent of any HTML-related heuristics and implementation details of the processed documents. This requirement is present to make out method resilient to potential future changes in technologies used on the web. Moreover, the method should produce a flat model of the segmented page consisting of a list of visual segments with a consistent granularity level.

[*] Corresponding author.
E-mail addresses: izeleny@fit.vutbr.cz (J. Zeleny), burgetr@fit.vutbr.cz (R. Burget), zendulka@fit.vutbr.cz (J. Zendulka).

http://dx.doi.org/10.1016/j.ipm.2017.02.002
0306-4573/© 2017 Elsevier Ltd. All rights reserved.

And finally, the method must be unsupervised. The quality criteria against which this new method is evaluated include both speed and precision of the algorithm.

1.1. Background

Even though from the technological point of view, the web pages are considered as atomic carriers of information in the World Wide Web, in some research areas, it has been clear for some time (Cai, Yu, Wen, & Ma, 2003) that this granularity is too coarse for processing the contained information. Most web pages are logically split in smaller pieces. From the data mining point of view, some of these pieces can be thrown away as their informational value is negligible. Others can be then used for various purposes by different data mining techniques.

Page segmentation usually presents a preprocessing step in a more complex document processing task. From this point of view, we may find several application domains of page segmentation. Information retrieval and content classification techniques use page segmentation to improve both precision and performance by eliminating those parts of web pages that don't contain useful content Win and Thwin (2014). Separation of multiple topics in one web page is used for example in content classification. This important process can also use segmentation to gain precision Yu, Cai, Wen, and Ma (2003). In the adaptive view transformation (Aguado, 2015; Coondu, Chattopadhyay, Chattopadhyay, & Chowdhury, 2014), segmentation is used to identify coherent parts of the web page that should be kept undivided. Finally, in the information extraction area, page segmentation may be used for the identification of the data-intensive document sections Weng, Hong, and Bell (2011) or even the individual data fields Milička and Burget (2015).

Depending on the target application, different segmentation granularity may be required. The granularity corresponds to visual consistency of segments identified in the page. For the typical applications mentioned above, the following granularity levels may be considered:

- Informative content blocks level – for the page cleaning tasks in the information retrieval and document cleaning areas, the page segmentation is required to discover the basic blocks in the page such as the main content area, header, footer, etc. (Alassi & Alhajj, 2013; Uzun, Agun, & Yerlikaya, 2013; Win & Thwin, 2014; Wu, 2016).
- Paragraph level – for some applications such as vision-based classification of logical parts of the published information Burget (2010); Weng, Hong, and Bell (2014), a finer granularity is required that corresponds to the individual logical parts of the content such as headings, paragraphs, list items, etc.
- Data field level – the finest granularity level is required usually in the information extraction area when the individual data fields have to be identified and extracted Milička and Burget (2015).

Current page segmentation methods such as VIPS and its successors (described in detail in Section 2) produce a hierarchical model of the segmented page that is created by a recursive division (in case of the top-down approaches) or grouping (for the bottom-up approaches) of the detected visual blocks. The required granularity level then corresponds to the size of the leaf nodes of the produced hierarchy and for most segmentation methods, it can be adjusted by setting different parameters of the particular segmentation method such as the *degree of coherence* parameter in VIPS. However, for most of the above mentioned applications, the leaf nodes of the hierarchy are actually the most important ones. The content classification or information extraction methods examine the visual segments of the required granularity and actually do not use the complete hierarchy produced. Therefore, for several applications we have investigated recently (Burget, 2010; Milička & Burget, 2015), we found it more efficient to directly obtain a list of visual segments of the required granularity instead a hierarchical model.

In this paper, we propose the Box Clustering Segmentation (BCS) method that meets the requirements presented in the beginning of this section. Our method is built from ground up and it has virtually nothing in common with existing tree-based algorithms. We embrace a different, so far very marginally explored approach to the page segmentation problem. It is based on processing the rendered page using only very general visual cues. In contrast to the most of current methods, our algorithm does not produce a hierarchy of areas; instead, it aims to put together tiles on the same level of hierarchy. If detected correctly, the tile representation is a much more accurate representation of a web page in terms of user perception and it is more suitable for many application as discussed above. In contrast to most of the existing methods, we don't use any tree-processing approach. Instead, we rely on clustering techniques with a proper distance model in place. The simplified tile representation also allows to achieve a significantly faster segmentation which is traditionally an important issue in case of the vision-based methods.

The rest of our paper is organized as follows: Section 2 introduces the state of the art in the area of web page segmentation. Section 3 then introduces the main concept of the Box Clustering Segmentation. Sections 4, 5 and 7 explain the individual parts of the algorithm in detail. Section 6 covers the metrics we use for the clustering algorithm. Section 8 presents the results of our algorithm and compares them to the reference algorithm and finally, Sections 9 and 10 sum up the achieved results.

2. Related work

Our research deals with the issue of splitting up a web page into smaller segments. There has been a lot of research in this area in recent years and several types of algorithms exist to address this problem. Note that we don't compare our method to any algorithms used for segmentation of scanned documents, as the nature of the input data is completely different.

Template detection algorithms belong to the first type. Their goal is to identify and filter out those parts of a web page that repeatedly occur on similar pages. The assumption is that these parts together constitute a template, sort of a skeleton of the page that can be dropped without loosing the relevant content (Alarte, Insa, Silva, & Tamarit, 2015; Barua, Patel, & Agrawal, 2014; Gao & Fan, 2014). Template detection methods (Kulkarni & Patil, 2014; Kulkarni & Kulkarni, 2015; Lundgren, Papapetrou, & Asker, 2015) are usually quite fast when compared with other methods for page pre-processing and they scale well. Another positive aspect is that they are usually unsupervised. The problem is that they usually need more than one web page to even work and their precision is often highly dependent on higher number of inspected pages. When considering our applications, their other problem is that they typically distinguish only between the useful content and the template (Barua et al., 2014) and they do not offer finer granularity levels.

Compared to template detection, *wrapper generation* is more similar to page segmentation. It is a procedure of creating wrappers – programs that can be later used for extracting particular areas from the given web page. From our perspective, each wrapper can be perceived as a descriptor of a particular segment on the page. However, compared to web page segmentation, literature concerned with wrappers mostly focuses only on information extraction tasks (Dalvi, Kumar, & Soliman, 2011; Ferrez, Groc, & Couto, 2013; Xiang, Yu, & Kang, 2015). The focus on information extraction is logical considering that each wrapper extracts a segment of the page.

The area of web page segmentation itself has also been researched extensively. In general, we may say that page segmentation gives the best results in terms of combination of granularity (superior to template detection) and generality of their subsequent usage (superior to both template detection and wrappers). Various methods belong to this group of algorithms. They can be divided into partially or fully supervised (Bing, Guo, Lam, Niu, & Wang, 2014; Bu, Zhang, Xia, & Wang, 2014; Fragkou, 2013) and unsupervised (Burget, 2007; Cai et al., 2003; Hong, Siew, & Egerton, 2010; Liu, Meng, & Meng, 2010; Shi, Liu, Shen, Yuan, & Huang, 2015) methods. In this work, considering the goals and applications we formulated in the introduction, we consider only the unsupervised page segmentation methods.

Depending on the underlying model used for the representation of the source documents, the web page segmentation methods are usually divided in up to four categories (Eldirdiery & Ahmed, 2015a): DOM-based approaches, text-based approaches, vision-based approaches and hybrid approaches. The DOM-based approaches (Hong et al., 2010; Jiang & Yang, 2015; Shi et al., 2015) operate on an object representation of the HTML code (Document Object Model) that represents the individual HTML elements contained in the code and their nesting. Some related information extraction (Uzun, Agun, & Yerlikaya, 2012) and document cleaning (Uzun et al., 2013; Wu, 2016) approaches share the same concept too. Because the information available in the DOM is very limited and it does not include the visual features of the individual elements, the DOM-based approaches include many heuristics that are used for an approximate estimation of the purpose of the individual HTML elements in order to identify those that form the page segments based on their typical usage in web design. Similarly, the text-based approaches (Bu et al., 2014; Eldirdiery & Ahmed, 2015b; Kohlschütter, Fankhauser, & Nejdl, 2010) focus on the properties of the text content, such as density, to detect the content segments. The segmentation methods based on both the DOM-based and text-based approaches are typically very fast because no complex document preprocessing (such as style analysis or rendering) is required. On the other hand, they operate on a simple approximation of the document based on its code and/or text content and therefore, the accuracy of the segmentation[1] greatly depends on the code properties and the used heuristics.

The vision-based approaches focus on the analysis of visual features of the document contents as they are perceived by a human reader. In case of HTML documents, obtaining the necessary visual information requires processing the document by an HTML rendering engine in order to compute the style and layout of the individual elements. Considering the visual information allows to achieve a higher segmentation accuracy in comparison to the DOM-based approaches. On the other hand, the necessity of page rendering and more complex document models processed typically in multiple steps make the vision-based approaches significantly slower and less scalable than the DOM-based ones.

VIPS (Cai et al., 2003) is probably one of the first and most popular vision-based algorithms. Even some template detection algorithms use VIPS instead of the usual DOM-based approach for detecting the visual structure of the page that is further analyzed in order to distinguish the template from the content (Alassi & Alhajj, 2013; Krishna & Dattatraya, 2015). The VIPS algorithm operates in the following steps: first, the page is divided into visual blocks; then, visual separators are discovered in the page and finally, the resulting page structure is constructed. Although the visual features of the individual elements (such as font sizes and colors) and their positions in the rendered page are taken into account, mainly the first visual block extraction step depends on a number of heuristic rules that are based on the underlying DOM and the HTML elements. Therefore, the VIPS method is sometimes considered as a DOM-based method with visual cues (Zeng, Flanagan, Hirokawa, & Ito, 2014). Since the heuristic rules strongly depend on particular usage of certain HTML elements, the VIPS method is no more directly usable for current web pages due to the evolution of web design techniques and the HTML language itself. Therefore, many extensions have been proposed in order to overcome these limitations: Akpinar and Yesilada (2012, 2013) extend the set of heuristic rules in order to cover the new tags introduced in modern versions of the HTML language and the new web design techniques. Li, Zhou, Fang, Liu, and Wu (2014) employ text statistics in order to recognize

[1] The accuracy is usually defined as a consistency between the result and human perception of the page.

similar visual blocks and Zhu, Dai, Song, and Lu (2015) use text features instead of the particular HTML tags in some VIPS heuristics. The latest approaches avoid the usage of the heuristics completely by using alternative ways of block detections. Burget (2007) uses a bottom-up grouping of visually aligned blocks, Alcic and Conrad (2011) use a clustering technique based on several distance metrics such as DOM-based, geometric and semantic distance. Liu, Lin, and Tian (2011) construct a graph of spatial relationships among the rendered elements that is later partitioned with a Gomory-Hu clustering algorithm and Zeng et al. (2014) compute a seam degree and content similarity of the individual blocks in order to divide larger visual blocks to smaller parts. Finally, Xu and Miller (2015) apply the Gestalt laws of grouping on the extracted visual blocks.

In contrast to VIPS and its successors, other vision-based approaches use entirely graphical representation of the input document that allows to abstract from the HTML-related implementation details. Cormier, Moffatt, Cohen, and Mann (2016) use an edge detection algorithm for detecting the visual separators between the content blocks. Similarly, Wei, Lu, Li, and Liu (2015) use Hough transform for the same purpose and Kong et al. (2012) recognize atomic objects using image processing methods and perform their grouping by using a spatial graph grammar.

The hybrid approaches combine the DOM-based and vision-based ones in order to obtain higher segmentation accuracy or for specific applications. Sanoja and Ganarski (2014) and Manabe and Tajima (2015) both combine the content structure (DOM) with the visual information obtained from a web browser in order to increase the accuracy in comparison to the VIPS algorithm. Safi, Maurel, Routoure, Beust, and Dias (2014) process the input document in two steps: first, a visual information analysis is performed and in the second step, DOM tree filtering is performed based on the analysis results with the aim of supporting visually impaired users. Finally, Fumarola, Weninger, Barber, Malerba, and Han (2011) combine the DOM with a visual information model in order to extract visually presented lists from the input documents. More generic content extraction use case is presented by Song, Sun, and Liao (2015)

In our Box Clustering Segmentation method, we strictly avoid using DOM and the HTML-based heuristics. We use a purely visual representation of the documents which makes our method closer to other methods based on the graphical document representation (Cormier et al., 2016; Wei et al., 2015). On the other hand, we don't detect the visual separators explicitly and the clustering approach is closer to the Web Content Clustering by Alcic and Conrad (2011).

Comparing our Box Clustering Segmentation to the VIPS and the Web Content Clustering, we find two major advantages of our approach. First, strictly utilizing just the visual information gives our algorithm an advantage of being more robust, as it doesn't depend on features of DOM model or HTML language which are subject to change. In the context of web page segmentation, this feature is especially visible on highly dynamic web pages where the DOM tree may actually be quite misleading because there is a lot of relative and absolute positioning utilized on these web pages. That makes the resulting positions and the relations between the visual areas quite different from the relations between their respective DOM nodes. Out of the context of web page segmentation, the advantage of our method it that it is applicable on other documents than web pages; it can be used on any document where we can retrieve information about position, size and color of all the elements with some content (e.g. images and text bounding boxes), which is true for example for most PDF documents available on the web. Second, we find our flat area model much more comprehensible and convenient for further processing than the tree model produced by VIPS. This is further discussed in Section 9.

3. Box clustering segmentation

The Box Clustering Segmentation (BCS) is designed to be a pure vision-based method. Also, in contrast to other segmentation methods, BCS is designed to give flat results. That means, it produces a tiled arrangement of segments rather than their hierarchy, which is the usual layout of segmentation results.

The entire process of the BCS is outlined in Fig. 1. Each box in the figure represents a state of the data being processed. Each transition between the states then represents an action that is taken. The first box marked *Web page* represents the input of the entire algorithm – a web page in a form of the HTML code or the corresponding DOM tree. The *rendering* step is done outside of the BCS and is therefore partially independent; any rendering engine can be used for this action. In our BCS implementation, we use the CSSBox rendering engine[2], as it offers the most convenient application interface for accessing the rendered page model. The rendering result is represented as a *rendering tree* that describes the final appearance of the rendered page as described in Section 4.

The Box Clustering Segmentation itself consists of three steps. The first one, called *box extraction*, takes the rendering tree and filters out those parts of the tree that are not useful for the subsequent clustering. The distances between the remaining boxes are computed in the second step based on the criteria described below. Several functions are created as a product of the distance computation. The *clustering* step finally processes the entities and identifies segments of the web page by clustering boxes belonging to the same segment.

4. Box extraction

The box extraction is the first step of the Box Clustering Segmentation. It is possible to think of it as a preprocessing step. Before we explain in detail what takes place during the preprocessing, let us inspect the rendering step and its output.

[2] http://cssbox.sourceforge.net/.

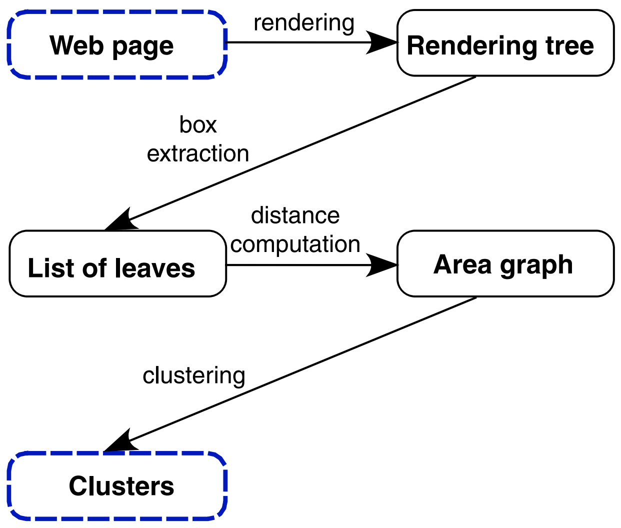

Fig. 1. Architecture of Box Clustering Segmentation.

A rendering engine generally transforms the input document represented as a DOM tree accompanied by Cascading Style Sheet (CSS) definitions and other additional data to a visual formatting model that is suitable for displaying the resulting page. The visual formatting model itself and the way how it is created is defined by the Cascading Style Sheet Specification (Bos, Celik, Hickson, & Lie, 2011). It is basically a tree of *boxes* where each box represents a rectangular area in the rendered page. We call this tree a *rendering tree* in this paper. Each box in the rendering tree corresponds to a particular node in the input DOM tree or its part; i.e. it is always possible to identify the source DOM node which generated that particular box. The leaf nodes of the rendering tree are elementary visual boxes that represent atomic units of the content, for example lines of text. Non-leaf nodes correspond to elements in the source DOM and they serve as either wrappers or groups of the visual boxes, for example paragraphs.

The box extraction algorithm performs a pre-order traversal of the rendering tree during which it selects the boxes that will be used in the next steps of the BCS. Among other things, each box contains some basic information that is necessary for computing the box similarity in the next clustering step. This information includes:

- The color of the box
- Box position in the page
- The size and shape of the box (derived from its width and height)

The idea of the box selection is to consider only those boxes that are actually visually rendered in the page. This is usually true for the leaf nodes of the rendering tree, as they represent the actual content of the page. However, several exceptions from this rule exist. The following list provides an explanation of the box extraction process:

1. **Text nodes** are always leaf nodes. They contain a line or its part. Graphically, each text box is a minimal bounding box of the text contained. These nodes are always selected.
2. **Image nodes** are always leaf nodes. They represent a particular image and therefore, they share its properties like position and size. These nodes are always selected.
3. **Childless boxes** that don't fall into previous categories are omitted.
4. **One-child boxes** that don't fall into previous categories are viewed as subtrees rooted at the child box. These subtrees are then inspected for branches and if no branches exist, the smallest box in the subtree with a non-transparent background is selected. If there is no such box, the leaf box is selected.

After the traversal is completed, it is remotely possible that some extracted boxes will visually contain other boxes. If this happens, all such cases are identified and the larger boxes are deselected.

5. Connecting the boxes

After all useful boxes are selected, we virtually connect them by detecting their adjacency. The first step to do that is to detect their semi-alignment. This is one of the important elements in the design of Box Clustering Segmentation. The following definitions describe box structure and semi-alignment of two boxes.

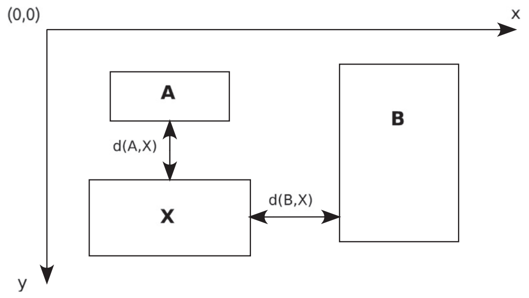

Fig. 2. Absolute distance measurement between boxes.

Definition 1 (Box structure)**.** Let the box be defined as seven-tuple $m = (left, right, top, bottom, width, height, color)$ where *left, right, top* and *bottom* are integer values that represent positions of respective edges of the box; $width = right - left$; $height = bottom - top$ and *color* represents dominating color of the box in RGB format.

Definition 2 (Projected overlap and semi-alignment)**.** Let m and n be two boxes on a web page. The projected overlap of boxes m and n is defined as a function pov: $(m, n) \rightarrow \{x, y, o\}$ where x and y indicate projected overlap on the respective coordinate axes of the web page and o designates "no projected overlap". The following rules apply:

$$pov(m,n) = \begin{cases} x & \text{if} \quad m.right \geq n.left \wedge m.left \leq n.right \\ y & \text{if} \quad m.bottom \geq n.top \wedge m.top \leq n.bottom \\ o & \quad\quad \text{otherwise} \end{cases} \tag{1}$$

The two boxes m and n are in semi-alignment if $pov(m, n) \neq o$.

Mutual position is a finer grained version of the projected overlap that is used when determining the distance between boxes.

Definition 3 (Mutual position of two boxes)**.** Let a set of possible positions be $P = \{a, b, l, r, o\}$ where a, b, l, r designate position above, below, left and right respectively and o designates "other position". The position of box m relative to box n is defined as a function pos: $(m, n) \rightarrow P$ where the following rules apply:

$$pos(m,n) = \begin{cases} a & \text{if} \quad m.bottom \leq n.top \wedge pov(m,n) = x \\ b & \text{if} \quad m.top \geq n.bottom \wedge pov(m,n) = x \\ l & \text{if} \quad m.right \leq n.left \wedge pov(m,n) = y \\ r & \text{if} \quad m.left \geq n.right \wedge pov(m,n) = y \\ o & \quad\quad \text{otherwise} \end{cases} \tag{2}$$

Besides being used for defining the neighborhood of each box, semi-alignment is also used for limiting the domain of the similarity function as described in Section 6. There are two reasons for limiting both the domain of the similarity function and the number of boxes included in the neighborhood detection. The first one is purely practical – it is easier for the clustering algorithm to extract the neighboring boxes when the number of candidates is limited. The same applies for calculating the similarity – it helps to reduce the number of similarity calculations. The second reason is based on our observation that boxes that are visually related are always organized this way (placed right next to each other or right below each other).

Now, for the adjacency itself. In this paper, we use a term *direct neighborhood* to express a set of boxes adjacent to a specific box. To understand the direct neighborhood, it's important to know how absolute distances between boxes are calculated. These distances, graphically outlined in Fig. 2, are formally expressed by the function *abs*() that is included in Definition 4.

Definition 4 (Absolute Distance, Direct Neighborhood of a Box)**.** Let B be a set of boxes on a web page and let $m, n \in B$. Absolute distance between two boxes is a function $abs : B \times B \rightarrow \mathbb{R}$:

$$abs(m,n) = \begin{cases} n.top - m.bottom & \text{if } pos(m,n) = a \\ m.top - n.bottom & \text{if } pos(m,n) = b \\ n.left - m.right & \text{if } pos(m,n) = l \\ m.left - n.right & \text{if } pos(m,n) = r \\ \infty & \text{otherwise} \end{cases} \tag{3}$$

Direct neighborhood of a box m is defined as $N_m = \{n | n \in B \wedge pos(m,n) \neq o \wedge \nexists k \in B : (pos(m,k) = pos(m,n) \wedge abs(m,k) < abs(m,n))\}$

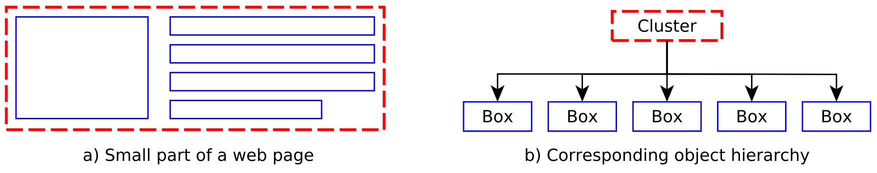
Cluster
Box
Box
Box
Box
Box
a) Small part of a web page
b) Corresponding object hierarchy

Fig. 3. Hierarchy of clusters and the corresponding boxes.

Direct neighborhoods of all the boxes in a web page essentially create virtual connections that are being used in further calculations and, subsequently, the clustering algorithm itself.

6. Similarity model

A proper model for evaluating the similarity of elements in a web page is a core piece of any segmentation algorithm. In this paper, we use a compound similarity model that consists of two parts. To understand both of them, it's necessary to understand the organization of elements in a web page as we represent it.

Boxes have already been described in Section 4. A *cluster* is the second element type that we use in BCS. Together, the two types of elements form a two-level hierarchy as outlined in Fig. 3.

As stated before, we use two different similarity metrics. The first one, called *base similarity*, is based on the visual features of boxes. The second one, called *cluster similarity*, is then used to express the similarity between two elements where at least one of them is a cluster.

6.1. Base similarity

The base similarity can be calculated for any pair of boxes. However, for the reasons explained in Section 5 we calculate it only for those pairs that are semi-aligned. We have chosen a simple similarity model based on several visual properties of the compared boxes. The reason for this choice was to make the algorithm both transferrable to other types of documents and resilient to any potential HTML, CSS or DOM changes in the future.

The base similarity is essentially an arithmetic mean of three components that are described further: *distance, shape similarity* and *color similarity*:

$$bsim(m,n) = \begin{cases} 0 & \text{if } distance(m,n) = 0 \\ 1 & \text{if } distance(m,n) = 1 \\ \dfrac{\begin{pmatrix} distance(m,n)+ \\ sim\_shape(m,n)+ \\ sim\_color(m,n) \end{pmatrix}}{3} & \text{otherwise} \end{cases} \tag{4}$$

6.1.1. Distance

Distance between elements of a web page is the primary way how web designers separate the visual and semantic blocks in the web page. As such, it is also often used as the primary indicator of the separation in segmentation algorithms, including VIPS Cai et al. (2003). Our distance model is based on some ideas that come from our prior work Burget (2007). In the current model, we use relative distances that are computed in two steps. The first step – detecting direct neighborhood – was already covered in Section 5. Using the direct neighborhood, we then transform the absolute distances to relative distances, expressed by the *distance*() function.

Definition 5 (Relative distance)**.** Let B be a set of all boxes on a web page. For each box $m \in B$ and its direct neighborhood N_m, there is a maximal neighborhood distance $maxd(m) = abs(m,k)$ where $k \in N_m \wedge \nexists l \in N_m\colon abs(m,l) > abs(m,k)$. For each $n \in N_m$ the relative distance, designated $distance(m,n)$, is calculated as:

$$rel_m(m,n) = \frac{abs(m,n)}{maxd(m)} \tag{5}$$

$$rel_n(m,n) = \frac{abs(m,n)}{maxd(n)} \tag{6}$$

$$distance(m,n) = \frac{rel_m + rel_n}{2} \tag{7}$$

Table 1
Colors that are assigned to basic box types.

Box type	Assigned color
Images	Color tone of the image
Text	Font color of the text
Other leaf boxes	Background color of the box

The relative distance expresses how far the two boxes are from each other in context of their direct neighborhoods. The function as formulated in Definition 5 can obviously assume values in the range $< 0, 1 >$ with direct correlation between the distance and the value $rel(m, n)$.

6.1.2. Shape

When comparing the shapes of two boxes, we base our calculation on a premise that the boxes that look similar are likely to belong to the same cluster. This metric was included after our observation that the shape similarity is often what visually binds together blocks of text or items in menus. On the other hand, some other cases exist in which we don't necessarily want to group the boxes that are similar in some way. To distinguish between the different types of similarity, we came up with a system utilizing the aspect ratio and the size of the two compared boxes.

Let's have two boxes $m, n \in B$. For the aspect ratio comparison of the boxes, we use the following formulas:

$$r_m = \frac{m.width}{m.height} \tag{8}$$

$$r_n = \frac{n.width}{n.height} \tag{9}$$

$$ratio(m,n) = \frac{max\{r_m, r_n\} - min\{r_m, r_n\}}{\frac{max\{r_m, r_n\}^2 - 1}{max\{r_m, r_n\}}} \tag{10}$$

The second part of shape similarity measurement is the size comparison. We use the following formulas for evaluating the size similarity of two boxes $m, n \in V$:

$$s_m = m.width * m.height \tag{11}$$

$$s_n = n.width * n.height \tag{12}$$

$$size(m,n) = 1 - \frac{min\{s_m, s_n\}}{max\{s_m, s_n\}} \tag{13}$$

The final shape similarity is simply calculated as a mean value of *ratio* and *size*:

$$sim\_shape(m,n) = \frac{ratio(m,n) + size(m,n)}{2} \tag{14}$$

6.1.3. Color

Color difference presents another method that both web developers and segmentation algorithms use to separate visual segments of web. The boxes in a web page may contain many colors such as the text (foreground) color, background color, borders or even more colors in case of images. For computing the color distance of two boxes, we assign each box a single color depending on its type as shown in Table 1.

The color distance itself is a metric used to quantify the difference between two colors. It is commonly denoted as ΔE and there exist many formulas to calculate it. The actual choice of the most suitable formula depends on the application (Sharma, 2004).

The International Commission on Illumination came up with several formulas that work on *Lab* and *LCH* color spaces. They are all based on the fact that the human eye is more sensitive to changes in chroma than to changes in lightness (Sharma, 2004). As opposed to *RGB* color space, both *Lab* and *LCH* color spaces allow a separate calculation for lightness and chroma.

In our application, we have experimented with both *Lab* and *LCH* based color distances; however, we have evaluated a simple euclidean *RGB*-based difference as the one with the best results. Our observations showed that the reason is most likely that the web designers most often use different hue rather than chroma to distinguish between the components that don't belong to each other.

Because the results of the color distance have to match all the other components of the box comparison, we have normalized the euclidean distance by dividing it by the maximal diagonal distance in the *RGB* color space. In our color representation, we use the standard *RGB* model where each color channel can assume values in the range $< 0, 1 >$ and the maximal diagonal distance is $\sqrt{3}$.

Definition 6 (Color similarity)**.** Let m, n be two boxes and let $m.color = (R_m, G_m, B_m)$ and $n.color = (R_n, G_n, B_n)$ be their color representations. The color similarity *sim_color* is defined as

$$sim\_color = \frac{\sqrt{(R_n - R_m)^2 + (G_n - G_m)^2 + (B_n - B_m)^2}}{\sqrt{3}} \tag{15}$$

6.2. Cluster similarity

When grouping single boxes into clusters, it is necessary to extend the similarity model to accommodate the clusters; that means, we need to evaluate the similarity of two clusters or a cluster and a box. Unfortunately, the characteristics of clusters cannot be simply inherited from the characteristics of the individual boxes constituting the clusters, mainly as it is difficult to interpret the contribution of individual boxes to the whole cluster.

Therefore, we use a model that is based on clusters' inner similarity indicators. This model builds on base similarity and follows the idea of Degree of Coherence in VIPS and box clustering in Burget (2007). The inner similarity is basically a mean value of base similarity that is calculated using the boxes within a cluster. As a prerequisite of this representation, the definition of direct neighborhood must be extended so it can accommodate both clusters and boxes. The model derives direct neighborhood of each cluster from direct neighborhoods of all the boxes contained in that cluster.

Definition 7 (Unclustered Boxes, Cluster Direct Neighborhood)**.** Let B and C respectively be sets of boxes and clusters on a web page. Furthermore, let $B_c \in C$ be a set of boxes constituting cluster c and let N_m designate direct neighborhood of box m. A set of unclustered boxes on the page is defined as $B_U = \{b | b \in B; \nexists B_c \in C : (b \in B_c)\}$.

For a cluster c, its direct neighborhood is defined as $N_c = \{m | m \in B_U \wedge \exists n \in B_c : n \in N_m\} \cup \{B_d | B_d \in C \wedge \exists m \in B_c : \exists n \in B_d : n \in N_m\}$.

Corresponding to the previous definition, the value of similarity between a cluster and any entity in its direct neighborhood represents the mean value of similarities between that entity and all the boxes contained in the cluster. The previous text implies only connections between pairs of boxes that are adjacent where each such pair has a corresponding similarity value. However, the concepts of connection cardinality and cumulative similarity introduce additional functions *card*() and *cumul*() which are defined as follows:

Definition 8 (Connection cardinality and cumulative similarity)**.** Let B, B_U and C respectively be sets of boxes, unclustered boxes and clusters on a web page. Also, let N_e designate direct neighborhood of entity e. Functions $card : C \times C \cup B_U \to \mathbb{N}$ and $cumul : C \times C \cup B_U \to \mathbb{R}$ respectively represent connection cardinality and cumulative similarity. Note that the cumulative similarity uses the function $s()$ which is defined in Section 6.3. Both functions are defined as follows:

$$card(c, e) = |\{m | m \in B_c \wedge e \in N_m\}| \tag{16}$$

$$cumul(c, e) = \sum_{\forall m \in B_c} s(m, e) \tag{17}$$

When a cluster is being created, all the unclustered boxes are scanned and those ones that are about to become the cluster neighbors are selected for processing. For each of these future neighbors, the value of the cumulative similarity and the cardinality is calculated and the final similarity *csim* is then calculated as their quotient:

$$csim(c, b) = \frac{cumul(c, b)}{card(c, b)} \tag{18}$$

6.3. Entity similarity

Now when both compounds of the similarity model are described, their combined usage is straightforward:

Definition 9 (Entity similarity)**.** Let B and C be sets of boxes and clusters on a web page respectively and let $e_1, e_2 \in B \cup C$ be two entities. The entity similarity s is defined as:

$$s(e_1, e_2) = \begin{cases} bsim(e_1, e_2) & \text{if } e_1 \in B \wedge e_2 \in B \\ csim(e_1, e_2) & \text{if } e_1 \in C \\ csim(e_2, e_1) & \text{if } e_1 \notin C \wedge e_2 \in C \end{cases} \tag{19}$$

7. Clustering

The set of boxes B and the value of a *Clustering Threshold*, designated CT and described further, are the only inputs of the clustering algorithm. It outputs a set of identified clusters C. The clustering algorithm has four main parts that will be closely described in this section:

1. Creation of cluster seeds
2. Entity selection for merging – creation of candidate clusters
3. Overlap handling
4. Cluster verification and commission

The main algorithm loop and its initialization are shown in Fig. 1. The main loop itself covers the first two parts of the entire algorithm – the creation of the cluster seeds and the selection of entities for further merging. They are almost equivalent; the only difference is in the type of their input entities. The idea is to find the most similar couples of boxes and then, to select them for merging. If at least one of the entities is a cluster, a new candidate cluster is created. If both entities are boxes, a new cluster seed is created instead. However, even the seed should be considered just as a candidate seed at first. After it is committed, it becomes a valid cluster seed.

The selection of the two entities for merging is performed on line 6 of the Algorithm 1. The selected relation is then removed from the set to avoid infinite loops. With the two entities selected, we have to check if the similarity between them is within an acceptable range before the candidate is created. This is simple for two boxes; however, for clusters, we have to employ the formula from Section 6.2 to get the real value of similarity. Picking the right similarity value is abstracted by function *sim*(). If the difference between the selected entities is too big, the candidate is not created at all.

Algorithm 1 The clustering algorithm.

```
1:  function BCS(IN: CT, IN OUT: G, OUT: C)
2:      loop
3:          if $|B_U| < 2$ then
4:              return
5:          end if
6:          $m, n \leftarrow m, n \in B_U \cup C : \nexists x, y \in B_U \cup C : (s(x, y) < s(m, n))$
7:          if $s(m, n) > CT$ then
8:              return
9:          end if
10:         create cluster candidate $cc$
11:         if $\exists c \in C : c\ overlaps\ cc$ then
12:             continue
13:         end if
14:         if $\exists b \in B : b\ overlaps\ cc$ then
15:             MERGEOVERLAPS($B, cc$)
16:         end if
17:         COMMIT($cc, G, C$)
18:     end loop
19: end function
```

The Clustering Threshold CT, used on line 7, is a static real number that can assume values in the range of $< 0, 1 >$. The Clustering Threshold corresponds to the Permitted Degree of Coherence (PDoC) used in VIPS algorithm. It has to be set in advance and it remains constant for the entire page. Picking the right value of CT is a difficult task and every web page has a different optimal value. If it's too low, many boxes will end up unclustered. On the other hand, if picked too high, some clusters that should be separate are merged instead. Compared to VIPS, there is also another consequence: The results can also look completely different with different values of CT. That is caused by the overlap merging phase. Selecting the right value of CT for the web page is out of scope of this paper, much like VIPS doesn't cover selection of the PDoC. In practical applications, we assume several approaches, for example an iterative or bisective approach with the number of unclustered boxes being used as an indicator when to stop. This solution is feasible with the support of our Cluster-based page segmentation (Zeleny & Burget, 2013) that can record the optimal value for one page and re-use it on other web pages that are similar.

Some tests are performed on the new candidate after it is created. These tests are the reason we create just a candidate – failing the tests prevents the cluster creation from being verified and in such cases it is easier to dispose of the cluster candidate than to undo the merging step.

1. An overlap with another cluster. Such overlaps are not allowed by definition in our method and therefore, if the candidate overlaps with another cluster, it is marked as invalid and it is removed.

2. An overlap with other boxes. This step is one of the most important ones. Both creating the cluster seed and extending the cluster by merging it with a nearby box can make the cluster overlap with other boxes. The overlap with a box can eventually end up being an overlap with another cluster. Since this is not acceptable, we have to consider the overlapped box or boxes for merging into the cluster right at the moment when the overlap is created. If the candidate boxes don't cause any new overlaps, they are accepted and merged into the cluster. Otherwise, the cluster candidate is removed as invalid. This is represented by the `mergeOverlaps()` function.

The entire cluster creation is then encapsulated in the function `commit()`. If the cluster candidate passes all the tests and it is not marked as invalid, it is marked as verified and committed to be a valid cluster. Several partial actions have to be carried out during this operation:

1. All new boxes in the candidate cluster are marked as members of the final cluster.
2. The inner similarity indicators of the cluster are re-calculated.
3. If the cluster candidate was created by merging other clusters, these clusters are deleted from the cluster set C and removed.
4. The new cluster is added to the cluster set C.

After the processing of all the boxes is finished, some boxes that don't belong to any group may still remain. These boxes are ignored in the result as they are likely not important in the web page in terms of information retrieval. If the usage context of the algorithm is content filtering, these boxes can be safely dropped.

8. Experimental evaluation

In order to verify our design, we have created a reference implementation. We use Java as a platform for the implementation for the reasons explained further. As a rendering engine, we use CSSBox that is written in Java. CSSBox is the most capable rendering engine in terms of access to internals and the platform independence.

The goal of the experimental evaluation is to compare our implementation with the existing algorithms. We use VIPS algorithm as a baseline. To make the comparison as accurate as possible, we use Java implementation of VIPS[3], which is based on the original paper (Cai et al., 2003). For achieving an accurate comparison, both segmentation programs are written in Java and both use the same CSSBox rendering engine. Therefore, the performance comparison is not influenced by the differences that can be caused by a different platform or rendering engine. Also the comparison of accuracy is more precise due to the same rendering engine being used.

In the comparison, we watch the following three criteria:

- Time the algorithm spent segmenting the web page.
- How accurate the results are.
- How stable across web pages the results are.

With respect to the watched criteria, the testing of every web page was performed as follows:

1. Pick a web page.
2. Let a user create reference segmentation of the web page.
3. Render box representation of the web page.
4. Run each algorithm multiple times, each time with different value of the Clustering Threshold and Permitted Degree of Coherence respectively.
5. Compute the mean run time of each algorithm and select the CT/PDoC that leads to the most accurate segmentation result.
6. Compare the run times and the accuracy of results.

The first step in the process is to pick a web page on which the segmentation is performed. To test the robustness of both compared algorithms, it was necessary to identify as wide variety of page layouts as possible and test at least one web page for every layout identified. There are several layout types of web pages we consider:

- *Complex index pages* – pages like news indexes or listings are characterized by a high degree of structure, as there are multiple topic areas covered on a single page, every area being represented by only a handful of boxes. E-commerce systems are also good representatives of this category of pages. The main difference is that the e-commerce systems usually have stronger structure in terms of similarity between individual elements.
- *Articles* – pages like these contain one main block of text, usually consisting of multiple paragraphs and image elements. Besides this one big block, there are some smaller areas that are usually related to navigation and some small pieces of generic information (like contact or news on company web sites).
- *Simple web pages* – this is a good example of some minimalistic web pages, usually educational ones. The main characteristics of web pages like these is a minimal amount of elements (and subsequently also visual areas) other than the main content.

[3] https://github.com/tpopela/vips_java.

Table 2
Algorithm run time comparison.

page	VIPS time	BCS time
businessinsider.com (article)	522 ms	20 ms
idnes.cz (index)	1079 ms	39 ms
idnes.cz (article)	723 ms	53 ms
novinky.cz (index)	28126 ms	699 ms
novinky.cz (article)	390 ms	18 ms
reuters.com (index)	475 ms	15 ms
reuters.com (article)	442 ms	37 ms
yahoo news (article)	342 ms	21 ms

We have created an evaluation dataset of real web pages containing all the mentioned page types as we describe further.

8.1. Evaluation dataset preparation

For creating the evaluation data set, we have identified 8 different types of pages from 5 news web sites that are listed in Table 2. Note that the *businessinsider.com* and *yahoo news* sites don't use paging and we were therefore unable to statistically process their respective index pages. We have collected a set of 100 pages for every type, which means 800 pages in total. Then, we asked three volunteers to independently create a reference segmentation for every page from the set.

To facilitate the work of the volunteers and to ensure consistent annotation of all pages of each of the eight types by a single volunteer, we have used a semi-automatic annotation approach. This approach is based on the fact that all the pages of the same type share the same template that is used for generatig their HTML code. We have created a graphical tool that allows the volunteer to interactively mark the visual clusters in one sample page of every type. Then, the tool maps the manually created segments to a DOM model of the sample page and subsequently, it automatically creates equal segments in the remaining 99 pages of the same type. Finally, the volunteer is able to browse the results of the automatic annotation graphically in order to verify its correctness. The annotation results are stored as text files containing the positions of the annotated segments for every page as well as PNG images showing the annotated segments graphically for later verification.

As a result, we have obtained 2400 annotated pages in total from our three volunteers.

8.2. Performance evaluation

Table 2 demonstrates the first part of the algorithm evaluation – the mean run times of both algorithms on the evaluation data set. As the Table 2 demonstrates, our algorithm is superior to the VIPS in terms of time required to process a web page. This difference gets bigger with decreasing complexity of evaluated web page.

8.3. Accuracy and stability evaluation

Evaluating the accuracy is a more complex task. In the area of page segmentation, there is no commonly used method for evaluating the accuracy. In statistical analysis in general, the F-score is a common way how to evaluate the accuracy. In Kreuzer, Hage, and Feelders (2013), the F-score is used, even though the underlying method for matching segments is rather crude. There is, however, an alternative that we can use. As this paper proposes, the page segmentation task, regardless of how it's performed, is basically a clustering task – each segment being a cluster of page elements. In data clustering, Adjusted Rand Index (ARI) (Hubert & Arabie, 1985) is being used to measure similarity between two clusterings. In this paper, we compare BCS and VIPS using both methods.

For ARI, each segmented web page is viewed as a clustering and every segment in that web page is a cluster of boxes rendered on that page. In case of F-score, we take a similar approach. We create pairs of *automatically detected areas* and *manually annotated areas* which share at least one rendered box. For each such pair, we calculate the precision and recall of that pair. If there are any manually selected areas that do not share boxes with any automatically detected areas, we set the recall value for each of them to 0. The resulting F-score is calculated using average values of precision and recall for the entire page. In both cases, we measure the accuracy using rendered boxes and their pertinence to visual areas in the reference segmentation. There are several rules when creating the reference segmentation:

- Each box is assigned to at most one visual area.
- There are no empty visual areas in the web pages –i.e. those that would contain no boxes.
- There are no overlapping visual areas in the page.
- Every visual area has to meet the *semantic condition*: The boxes in the area have to constitute one unit of content that is coherent visually, semantically or (preferably) both.

Evaluating one reference web page of a given type from a given site might be misleading, as there is no guarantee or even indication that segmenting other pages would generate any kind of corresponding results. That's why we performed statistical evaluation on a large set of pages.

Table 3
Algorithm accuracy comparison using the ARI and F-score metrics.

page	BCS ARI	VIPS ARI	BCS F	VIPS F
businessinsider.com (a)	0,5704	0,7010	0,6345	0,7394
idnes.cz (article)	0,6629	0,7240	0,5570	0,5720
idnes.cz (index)	0,5954	0,7926	0,5522	0,7259
novinky.cz (article)	0,7670	0,7877	0,6446	0,7191
novinky.cz (index)	0,5303	0,9121	0,4265	0,9043
reuters.com (article)	0,6123	0,6786	0,5914	0,6943
reuters.com (index)	0,5832	0,8160	0,5145	0,7569
yahoo news (article)	0,7556	0,5626	0,7102	0,5446

Table 4
Algorithm stability comparison: Standard deviation of the results in the dataset.

page	BCS ARI	VIPS ARI	BCS F	VIPS F
businessinsider.com (a)	0,1275	0,1740	0,0358	0,0721
idnes.cz (article)	0,0646	0,0766	0,0424	0,0794
idnes.cz (index)	0,0733	0,0078	0,0108	0,0070
novinky.cz (article)	0,1274	0,1406	0,0404	0,0731
novinky.cz (index)	0,0529	0,0140	0,0558	0,0219
reuters.com (article)	0,1316	0,1687	0,0301	0,0847
reuters.com (index)	0,0328	0,0516	0,0203	0,0336
yahoo news (article)	0,2089	0,1658	0,1000	0,0539

One problem remains in the evaluation system presented above and that is the hierarchy of visual area presented by VIPS. To eliminate the ambiguity that the hierarchy presents, only the leaf areas of the hierarchy will be used for the evaluation.

The Table 3 shows the comparison of accuracy of both BCS and VIPS. The F-score value is a real number between 0 and 1, where higher values are better. ARI score is between −1 and 1, higher values are better.

The results show that the accuracy of VIPS is slightly better, especially when processing structured pages. The reason is that BCS is too aggressive when creating clusters, thus effectively overlooking the structure. VIPS on the other hand does much better job in finding repeating patterns in the web page. When processing pages with less structure, the accuracy of BCS and VIPS is comparable, in some cases BCS is even better than VIPS.

Stability of both algorithms can be also calculated using the same data that was used to populate Table 3. We calculated stability in each data set, specifically as standard deviation of results in the set. The results are displayed in Table 4.

Again, both algorithms are comparable. In some cases the stability of BCS is almost three times better than that of VIPS, in others, it's exactly the opposite. Not looking at the degree of superiority, the stability of BCS is better in 5 data sets, i.e. 62.5% of measurements.template but across the templates as well

9. Discussion

Looking at the results in Section 8, the most important practical implication of the results emerges. Significantly increasing performance of vision-base page segmentation algorithms while not losing their level of accuracy opens them way to practical application, as speed is very important in modern data mining systems. Using just generic visual cues supports that attractivity, as one algorithm can be used for processing multiple document types and it is resilient to possible future changes in technologies like HTML.

The flat structure BCS produces is as important for the practical application as performance. Being able to easily consume the output of segmentation algorithm significantly lowers the barrier for using it. Note that this was even proven in Section 8 where extra measures had to be taken to make the results of VIPS comparable to the results of BCS.

To better demonstrate the advantage of the flat output of the Box Clustering Segmentation (BCS) in the evaluation process, the difference between the two output models is displayed in Fig. 4. The BCS flat model is quite straightforward – it is just a set of groups, each of which can be further processed right away. The VIPS tree model on the other hand is not that simple. Fig. 4 visualizes the different levels of the output tree with different shades of gray and the internal consistency level by numbers in the leaf areas. It may be understandable for a human observer; however, in context of an automatic processing, one needs to performs a subsequent deep analysis of the segmented result to select the right area set, as we don't necessarily want to always pick the leaf nodes of the tree. Even though the tree model offers some bright sides like the possibility to compensate for some potential defects in the output, we don't find them that significant. Therefore, we consider the flat model to be the most distinct advantage of BCS when compared to VIPS and other hierarchy-producing algorithms.

Section 8 briefly describes a brute force approach to selecting the best values of PDoC and CT. That highlights another aspect that is important in practical application – selecting the right value of the target granularity parameter. As the original

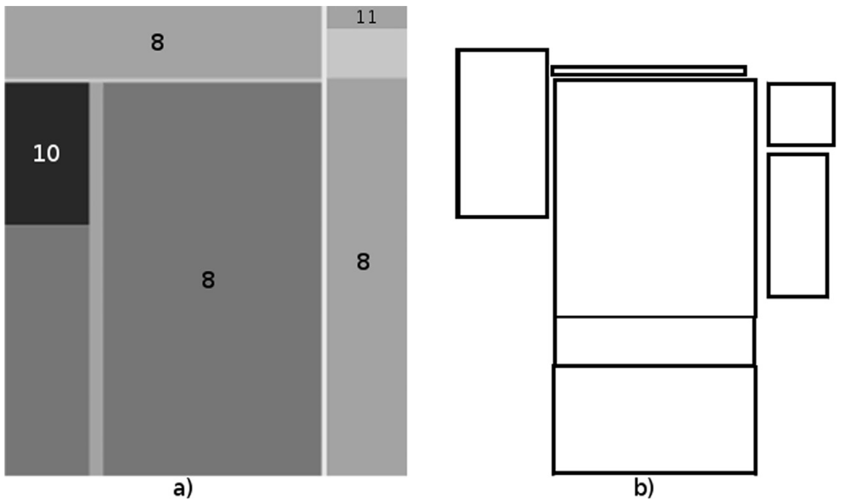

Fig. 4. Output model comparison: (a) VIPS tree model and (b) BCS flat model.

VIPS paper points out, different applications might need different output granularity and set the PDoC accordingly. That is our perception of CT as well. For that reason, any deeper investigation of the PDoC/CT selection process does not belong to this paper. But, since we are comparing BCS and VIPS, let's compare practical use of their respective granularity parameters. In both cases, the parameters are limited to fixed range of values and their change significantly influences the results of algorithms they are used in. We see the flexibility of CT being a real number to be a great advantage over PDoC which is integer. On the other hand, PDoC and how VIPS behaves when it changes is a great advantage over CT, as the results of changed PDoC are more predictable than the results of BCS when its CT changes.

In the field of theory, our paper opens new research area: new group of vision-based document segmentation can be explored. This area has a lot of potential, further research can improve both the accuracy and performance of vision-based clustering segmentation techniques. The overall potential of the Box Clustering Segmentation may be even greater considering it currently uses very low amount of information to perform the segmentation.

10. Conclusion

In this paper, we have presented a new web page segmentation method called Box Clustering Segmentation. We showed that its precision is comparable to VIPS in some cases and slightly worse in others. We have also shown that its performance is superior to the VIPS performance and we have presented two major advantages the Box Clustering Segmentation has over the existing algorithms. First of them is the strict usage of visual information only which makes our method transferrable to other document types. It also makes it resilient to changes in HTML, DOM and other technologies used on the web. The second advantage is the output structure that is more comprehensive and convenient for further processing.

The assumed applications of the proposed page segmentation method include the document cleaning, automatic adaptation of web pages for small screen devices, page preprocessing for information retrieval and document classification, logical structure discovery and information extraction tasks. By setting the appropriate values of the input parameters, the segmentation may be performed on any granularity level depending on the particular task.

Acknowledgment

This work was supported by The Ministry of Education, Youth and Sports of the Czech Republic from the National Programme of Sustainability (NPU II); project IT4Innovations excellence in science – LQ1602.

References

Aguado, J. (2015). Emerging perspectives on the mobile content evolution. *Advances in multimedia and interactive technologies: IGI global*. URL https://books.google.cz/books?id=Omm2CgAAQBAJ

Akpinar, E., & Yesilada, Y. (2012). Vision based page segmentation: Extended and improved algorithm. Tech. Rep. eMINE Technical Report Deliverable 2 (D2), Middle East Technical University, Ankara, Turkey.

Akpinar, M. E., & Yesilada, Y. (2013). Vision based page segmentation algorithm: Extended and perceived success. In *Revised selected papers of the ICWE 2013 international workshops on current trends in web engineering - Vol. 8295* (pp. 238–252). New York, NY, USA: Springer-Verlag New York, inc.

Alarte, J., Insa, D., Silva, J., & Tamarit, S. (2015). Temex: The web template extractor. In *Proceedings of the 24th international conference on world wide web, WWW '15 companion* (pp. 155–158). New York, NY, USA: ACM. doi:10.1145/2740908.2742835.

Alassi, D., & Alhajj, R. (2013). Effectiveness of template detection on noise reduction and websites summarization. *Information Sciences, 219*, 41–72.

Alcic, S., & Conrad, S. (2011). Page segmentation by web content clustering. In *Proceedings of the international conference on web intelligence, mining and semantics, WIMS '11, ACM, new york, NY, USA* (pp. 24:1–24:9). doi:10.1145/1988688.1988717.

Barua, J., Patel, D., & Agrawal, A. K. (2014). Removing noise content from online news articles. In *Proceedings of the 20th international conference on management of data, COMAD '14, computer society of india, mumbai, india, india* (pp. 113–116). URL http://dl.acm.org/citation.cfm?id=2726970.2726988

Bing, L., Guo, R., Lam, W., Niu, Z.-Y., & Wang, H. (2014). Web page segmentation with structured prediction and its application in web page classification. In *Proceedings of the 37th international ACM SIGIR conference on research & development in information retrieval, SIGIR '14* (pp. 767–776). New York, NY, USA: ACM.

Bos, B., Celik, T., Hickson, I., & Lie, H. W. (2011). Cascading style sheets level 2 revision 1 (CSS 2.1) specification. *W3c recommendation*.

Bu, Z., Zhang, C., Xia, Z., & Wang, J. (2014). An far-sw based approach for webpage information extraction. *Information Systems Frontiers, 16*(5), 771–785.

Burget, R. (2007). Layout based information extraction from HTML documents. In *Proceedings of the ninth international conference on document analysis and recognition - Vol. 02, ICDAR '07, IEEE computer society, Washington, DC, USA* (pp. 624–628).

Burget, R. (2010). Visual area classification for article identification in web documents. In *Proceedings of the 2010 workshops on database and expert systems applications, DEXA '10, IEEE Computer Society, Washington, DC, USA* (pp. 171–175). doi:10.1109/DEXA.2010.49.

Cai, D., Yu, S., Wen, J. r., & Ma, W. y. (2003). VIPS: A vision-based page segmentation algorithm. *Microsoft technical report MSR-TR-2003-79*.

Coondu, S., Chattopadhyay, S., Chattopadhyay, M., & Chowdhury, S. R. (2014). Mobile-enabled content adaptation system for e-learning websites using segmentation algorithm. In *Software, knowledge, information management and applications (SKIMA), 2014 8th international conference on* (pp. 1–8). doi:10.1109/SKIMA.2014.7083570.

Cormier, M., Moffatt, K., Cohen, R., & Mann, R. (2016). Purely vision-based segmentation of web pages for assistive technology. *Computer vision and image understanding*.

Dalvi, N., Kumar, R., & Soliman, M. (2011). Automatic wrappers for large scale web extraction. *VLDB Endowment*, 230.

Eldirdiery, H. F., & Ahmed, A. H. (2015a). Article: Web document segmentation for better extraction of information: A review. *International Journal of Computer Applications, 110*(3), 24–28.

Eldirdiery, H. F., & Ahmed, A. H. (2015b). Detecting and removing noisy data on web document using text density approach. *International Journal of Computer Applications, 112*(5), 32–36.

Ferrez, R., Groc, C., & Couto, J. (2013). Mining product features from the web: A self-supervised approach. In J. Cordeiro, & K.-H. Krempels (Eds.), *Web information systems and technologies, Vol. 140 of lecture notes in business information processing, Springer Berlin Heidelberg* (pp. 296–311).

Fragkou, P. (2013). Information extraction versus text segmentation for web content mining. *International Journal of Software Engineering and Knowledge Engineering, 23*(08), 1109–1137.

Fumarola, F., Weninger, T., Barber, R., Malerba, D., & Han, J. (2011). Extracting general lists from web documents: A hybrid approach. In *Proceedings of the 24th international conference on industrial engineering and other applications of applied intelligent systems conference on modern approaches in applied intelligence - Vol. Part I, IEA/AIE'11, Springer-Verlag, Berlin, Heidelberg* (pp. 285–294).

Gao, B., & Fan, Q. (2014). Multiple template detection based on segments. In *Advances in data mining. applications and theoretical aspects, Springer* (pp. 24–38).

Hong, J. L., Siew, E.-G., & Egerton, S. (2010). Information extraction for search engines using fast heuristic techniques. *Data & Ledge Engineering, 69*(2), 169–196.

Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification, 2*(1), 193–218. doi:10.1007/BF01908075.

Jiang, K., & Yang, Y. (2015). Noise reduction of web pages via feature analysis. In *Information science and control engineering (ICISCE), 2015 2nd international conference on* (pp. 345–348).

Kohlschütter, C., Fankhauser, P., & Nejdl, W. (2010). Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on web search and data mining, WSDM '10, ACM, New York, NY, USA* (pp. 441–450).

Kong, J., Barkol, O., Bergman, R., Pnueli, A., Schein, S., Zhang, K., & Zhao, C. (2012). Web interface interpretation using graph grammars. *IEEE Transactions on Systems, Man, and Cybernetics. Part C (Applications and Reviews), 42*(4), 590–602.

Kreuzer, R., Hage, J., & Feelders, A. (2013). *A quantitative comparison of semantic web page segmentation algorithms*. Universiteit Utrecht, Faculty of Science Master's thesis.

Krishna, S. S., & Dattatraya, J. S. (2015). Schema inference and data extraction from templatized web pages. In *Pervasive computing (ICPC), 2015 international conference on* (pp. 1–6).

Kulkarni, A., & Patil, B. (2014). Template extraction from heterogeneous web pages with cosine similarity. *International Journal of Computer Applications, 87*(3), 5.

Kulkarni, H. H., & Kulkarni, M. K. (2015). Template extraction from heterogeneous web pages. *International Journal of Electrical, Electronics and Computer Engineering, 4*(1), 125.

Li, L., Zhou, A. M., Fang, Y., Liu, L., & Wu, Q. (2014). An improved VIPS-based algorithm of extracting web content. In *Material science, civil engineering and architecture science, mechanical engineering and manufacturing technology II, Vol. 651 of applied mechanics and materials, Trans Tech Publications* (pp. 1806–1810).

Liu, W., Meng, X., & Meng, W. (2010). ViDE: A vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge and Data Engineering, 22*(3), 447–460.

Liu, X., Lin, H., & Tian, Y. (2011). Segmenting webpage with gomory-hu tree based clustering. *Journal of Software, 6*(12), 2421–2425.

Lundgren, E., Papapetrou, P., & Asker, L. (2015). Extracting news text from web pages: An application for the visually impaired. In *Proceedings of the 8th ACM international conference on PErvasive technologies related to assistive environments, PETRA '15, ACM, New York, NY, USA* (pp. 68:1–68:4).

Manabe, T., & Tajima, K. (2015). Extracting logical hierarchical structure of html documents based on headings. *Proceedings of the VLDB Endowment, 8*(12), 1606–1617.

Milička, M., & Burget, R. (2015). Information extraction from web sources based on multi-aspect content analysis. In *Semantic web evaluation challenges, semwebeval 2015 at ESWC 2015, Vol. 2015 of communications in computer and information science, Springer International Publishing* (pp. 81–92).

Safi, W., Maurel, F., Routoure, J.-M., Beust, P., & Dias, G. (2014). A hybrid segmentation of web pages for vibro-tactile access on touch-screen devices. *3rd workshop on vision and language (VL 2014) associated to 25th international conference on computational linguistics (COLING 2014), Dublin, Ireland*. 95–02

Sanoja, A., & Ganarski, S. (2014). Block-o-matic: A web page segmentation framework. In *Multimedia computing and systems (ICMCS), 2014 international conference on* (pp. 595–600). doi:10.1109/ICMCS.2014.6911249.

Sharma, A. (2004). *Understanding color management*. Graphic Design/Interactive Media Series, Thomson/Delmar Learning.

Shi, S., Liu, C., Shen, Y., Yuan, C., & Huang, Y. (2015). Autorm: An effective approach for automatic web data record mining. *Knowledge-Based Systems, 89*, 314–331.

Song, D., Sun, F., & Liao, L. (2015). A hybrid approach for content extraction with text density and visual importance of dom nodes. *Knowledge and Information Systems, 42*(1), 75–96. doi:10.1007/s10115-013-0687-x.

Uzun, E., Agun, H. V., & Yerlikaya, T. (2012). Web content extraction by using decision tree learning. In *2012 20th signal processing and communications applications conference (SIU)* (pp. 1–4). doi:10.1109/SIU.2012.6204476.

Uzun, E., Agun, H. V., & Yerlikaya, T. (2013). A hybrid approach for extracting informative content from web pages. *Information Processing & Management, 49*(4), 928–944.

Wei, T., Lu, Y., Li, X., & Liu, J. (2015). Web page segmentation based on the hough transform and vision cues. In *2015 asia-pacific signal and information processing association annual summit and conference (APSIPA), IEEE* (pp. 865–872).

Weng, D., Hong, J., & Bell, D. A. (2011). Extracting data records from query result pages based on visual features. In *Advances in databases: 28th British national conference on databases, BNCOD 28, Manchester, UK, July 12–14, 2011, revised selected papers, Springer, Berlin, Heidelberg* (pp. 140–153).

Weng, D., Hong, J., & Bell, D. A. (2014). Automatically annotating structured web data using a svm-based multiclass classifier. In *Web information systems engineering – WISE 2014: 15th international conference, thessaloniki, greece, october 12–14, 2014, proceedings, part I, Springer International Publishing, Cham* (pp. 115–124).

Win, C. S., & Thwin, M. M. S. (2014). Web page segmentation and informative content extraction for effective information retrieval. *International Journal of Computer & Communication Engineering Research, 2*(2), 35–45.

Wu, Y.-C. (2016). Language independent web news extraction system based on text detection framework. *Information Sciences, 342*, 132–149.

Xiang, Z. L., Yu, X. R., & Kang, D. K. (2015). Wrapper induction of news information for feeding to social networking service on smartphone. In *2015 17th international conference on advanced communication technology (ICACT)* (pp. 292–295). doi:10.1109/ICACT.2015.7224806.

Xu, Z., & Miller, J. (2015). Identifying semantic blocks in web pages using gestalt laws of grouping. *World Wide Web*, 1–22.

Yu, S., Cai, D., Wen, J.-R., & Ma, W.-Y. (2003). Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th international conference on world wide web, WWW '03, ACM, New York, NY, USA* (pp. 11–18). doi:10.1145/775152.775155.

Zeleny, J., & Burget, R. (2013). Cluster-based page segmentation – A fast and precise method for web page pre-processing. In *Proceedings of the 3rd international conference on web intelligence, mining and semantics, WIMS '13, ACM, New York, NY, USA*.

Zeng, J., Flanagan, B., Hirokawa, S., & Ito, E. (2014). A web page segmentation approach using visual semantics. *IEICE Transactions on Information and Systems, E97-D*(2), 223–230.

Zhu, W., Dai, S., Song, Y., & Lu, Z. (2015). Extracting news content with visual unit of web pages. In *Software engineering, artificial intelligence, networking and parallel/distributed computing (SNPD), 2015 16th IEEE/ACIS international conference on* (pp. 1–5).