An Empirical Comparison of Web Page Segmentation Algorithms

Johannes Kiesel[1] $^{[0000-0002-1617-6508]}$, Lars Meyer[1] $^{[0000-0002-7280-808X]}$, Florian Kneist[1] $^{[0000-0002-4111-662X]}$, Benno Stein[1] $^{[0000-0001-9033-2217]}$, and Martin Potthast[2] $^{[0000-0003-2451-0665]}$

[1] Bauhaus-Universität Weimar, `<first>.<last>@uni-weimar.de`
[2] Leipzig University, `martin.potthast@uni-leipzig.de`

Abstract Over the past two decades, several algorithms have been developed to segment a web page into semantically coherent units, a task with several applications in web content analysis. However, these algorithms have hardly been compared empirically and it thus remains unclear which of them—or rather, which of their underlying paradigms—performs best. To contribute to closing this gap, we report on the reproduction and comparative evaluation of five segmentation algorithms on a large, standardized benchmark dataset for web page segmentation: Three of the algorithms have been specifically developed for web pages and have been selected to represent paradigmatically different approaches to the task, whereas the other two approaches originate from the segmentation of photos and print documents, respectively. For a fair comparison, we tuned each algorithm's parameters, if applicable, to the dataset. Altogether, the classic rule-based VIPS algorithm achieved the highest performance, closely followed by the purely visual approach of Cormier et al. For reproducibility, we provide our reimplementations of the algorithms along with detailed instructions.

1 Introduction

When visiting a web page, a key step for human comprehension is to identify its semantic units. Eye-tracking studies show that participants identify such units immediately upon perceiving a web page, then inspect them one at a time, often starting with navigation elements [16]. To create a comprehensible web page, it is thus important for its author to group its content into such comprehensible semantic units that are easy to identify by its visitors. Though qualified web designers do so in a professional manner, every web page author possesses an intuitive understanding of the basic principles of Gestalt that apply here, as these principles form an integral part of human perception [8]. Naturally, these semantic units, then called web page segments, also form the basis for various web content analysis tasks, like content extraction [2], template detection [13], and design mining [11]. Consequently, several approaches for web page segmentation have been developed over the past two decades [10].

The ongoing and rapid development of web technologies like Cascading Style Sheets (CSS) and JavaScript (JS) has considerably increased the possibilities of web design over the past years. The elements of a web page encoded in its HTML source code can be more or less arbitrarily rearranged in its visual appearance in the browser, so that no correspondence between the linear order of elements in the source code and its visual ordering can be presumed. Since the focus of web page authors are mostly the human visitors and much less so web content analysis algorithms, there is hardly any incentive to emphasize the semantic units in the web page's HTML code. Web page segmentation algorithms thus increasingly focus on the visual rendition of a to-be-segmented web page; a recent algorithm completely disregards the HTML code [7]. But even the classic VIPS algorithm [3], which was introduced in 2003, uses the positions of elements in the rendered web page as features for its segmentation.

This reliance of algorithms on rendering the web page has limited the reproducibility of web page segmentation experiments, but the paper at hand demonstrates how to overcome this problem through the use of web archiving technology. In essence, several algorithms use JavaScript to segment the web page as it is rendered in a browser. However, to reproduce this situation properly, the following elements have to be kept constant: (1) the web page's complete source code (HTML, CSS, JS, images, etc.); (2) the browser, since different browsers and even different versions thereof render the same page differently; and (3) the browser's environment variables, like the date or random numbers, which the web page might request from the browser. These are not trivial requirements to meet, but modern web archiving technology can provide for a stable reproduction of web pages as they were rendered in the past [9].

We develop and present a reproducible empirical comparison of five segmentation algorithms, as well as an ensemble of them. The algorithms have been selected to represent and evaluate a variety of approaches and paradigms: two are rule-based, one is based entirely on visual edges, one has been originally developed for print documents, and one is a state-of-the-art approach in image segmentation of photos. For evaluation, we employ our Webis-WebSeg-20 dataset, which contains both a manually created segmentation ground-truth, and a web archive of 8490 web pages [10]. Moreover, we report on and show the importance of parameter tuning for the different algorithms. Documentation and provenance data of these experiments are available online.[3]

Among others, the results show that the classic VIPS algorithm still performs best when tuned to the dataset, but also that purely visual approaches can reach a competitive performance. Moreover, in adjusting the evaluation to the requirements of different downstream tasks of web page segmentation, we find that purely visual approaches are already the new state-of-the-art for downstream tasks that rely on pixel-based segments, like design mining. One of these purely visual approaches, the MMDetection algorithm, is able to reach this high performance despite being trained for a very different kind of input document than web pages: photos. The ensemble of four of the algorithms under consideration, however, does not outperform its base algorithms. Upon closer inspection, most of the ground-truth segments are identified by at least one of the algorithms.

After a brief literature review of web page segmentation experiments in Section 2, we detail our evaluation setup in Section 3 and the employed algorithms—including their parameter tuning—in Section 4. Section 5 discusses the empirical comparison of the algorithms.

[3]Code + documentation: https://github.com/webis-de/ecir21-an-empirical-comparison-of-web-page-segmentation-algorithms
Provenance data: https://doi.org/10.5281/zenodo.4146889

2 Related Work

A number of publications that propose a new web page segmentation algorithm compare it with the classic VIPS algorithm [3] (e.g., [7, 14, 17]), which can thus be considered closest to a standard baseline. In the original publication, VIPS has been evaluated with a three-scale human assessment on only 140 web pages: According to the assessors, 61% of web pages were segmented "perfectly," whereas just 3% "failed." Such an assessment is unfortunately hardly reproducible. Zeleny et al. [17] perform an empirical comparison of their algorithms with VIPS on 800 semi-automatically annotated web pages. Their performance measure, F, is closely related to $F^{*}_{B^3}$ (nodes), employed in this paper (Section 5), and indeed, a similar performance is measured for VIPS: 0.71 by Zeleny et al., and 0.70 here. For their visual-edge-based algorithm, Cormier et al. [7] compare its segmentations with that of VIPS on 47 web pages using an adapted Earth Mover's Distance as performance measure. They find, that, though there is some agreement, their algorithm "tends to produce results significantly different from VIPS." Our evaluation in Section 5 also shows such a difference. Manabe and Tajima [14] compare the performance of their HEPS algorithm with that of VIPS for the task of identifying web page blocks—i.e., textual segments with headings. In their comparison on 1219 web pages, they find that HEPS clearly outperforms VIPS for exactly identifying such blocks: block precision is 0.59 (HEPS) vs. 0.22 (VIPS), and block recall is 0.56 vs. 0.07. This is in contrast to our results, which indicate a superior performance of VIPS over HEPS, not only for a text-based evaluation. A possible explanation lies in their different approach to ground-truth creation, which is tailored towards the mentioned header-based blocks.

However, no large-scale comparison of web page segmentation algorithms exists so far. Kiesel et al. [10] attribute this situation to a lack of generic, standardized datasets, a lack of a common view on how to measure algorithm performance, and a lack of reproducible evaluation procedures. Reviewing the related work beyond the aforementioned papers, evaluation datasets and performance measures have usually been created in an ad-hoc manner, and with respect to just one of the various downstream tasks of web page segmentation, which has led to several very focused datasets and many incompatible performance measures. The problem of reproducibility has, to the best of our knowledge, scarcely been tackled in the relevant literature so far: Only Zeleny et al. [17] attempt to reduce the influence of different browsers by using the same rendering engine for all algorithms. Recently, web archiving technology has been considered for web page segmentation, addressing its reproducibility problem for the first time [9]. This technology has been used to create the new Webis-WebSeg-20 dataset [10], which is nearly an order of magnitude larger than previous ones, and which has been annotated without specific downstream tasks involving web page segmentation in mind, based on human perception only. Moreover, the use of this dataset as a new evaluation framework is proposed, capturing the existing views on how to measure algorithm performance within a unified evaluation measure that can be adapted to various downstream tasks. This paper builds on this framework, and uses it for a first empirical comparison of segmentation algorithms.

3 Experiment Setup

For the empirical comparison of web page segmentation algorithms, this paper employs the 8490 web pages of the Webis-WebSeg-20 dataset [10]. The web pages have been sampled from a variety of sites, 4824 in total [9]. The dataset contains for each web page a ground-truth segmentation, which is fused from the segmentations of five human annotators. Furthermore, the dataset contains a web archive file for each web page, which allows to re-render the web page as if viewed at the time of the archiving. For algorithms that need no complete re-rendering, the dataset also provides for each page the DOM HTML, a screenshot, and the list of DOM nodes mapped to their coordinates on the screenshot. The latter allows to convert between segment descriptions as screenshot coordinates and as sets of DOM nodes. As the ground-truth uses a flat segmentation for all web pages but some algorithms produce hierarchical segmentations, we flatten such hierarchical segmentations for the evaluation.

Our evaluation discusses the achieved P_{B^3}, R_{B^3}, and $F^*_{B^3}$ for each algorithm. The have been introduced by Kiesel et al. [10]. They are straightforward adaptations of the respective extended BCubed measures from clustering theory [1]. In a nutshell, P_{B^3} is based on the elements that are segmented together in both the ground-truth and algorithmically created segmentations (the "true positives" in the usual definition of precision) divided by the number of all elements segmented together in the algorithmically created segmentations (the "positives"). R_{B^3} has the same numerator, but is divided by the number of all elements segmented together in the ground-truth segmentation. As usual, F_{B^3} is the harmonic mean of both for one web page. We here report the values averaged over all web pages, and $F^*_{B^3}$ is then the harmonic mean of the averaged P_{B^3} and R_{B^3}. As discussed by Kiesel et al., P_{B^3} decreases if algorithmically created segments extend beyond ground-truth segments, whereas R_{B^3} decreases in the inverse case. Put another way, P_{B^3} ignores cases of over-segmentation—where the algorithmically created segmentation is more fine-grained than the ground-truth segmentation—, whereas R_{B^3} ignores cases of under-segmentation. A segmentation of one segment that contains the entire page would thus achieve an R_{B^3} of the maximum value of 1, whereas a segmentation that puts every element into an own segment would achieve a P_{B^3} of 1.

In order to provide results that are applicable for various downstream tasks of web page segmentation, we execute all experiments for each of the five types of atomic elements defined by Kiesel et al. Different downstream tasks of web page segmentation weigh certain errors differently. For example, although for most downstream tasks it does not matter how background space is segmented, it is important for tasks that consider the spacing between segments, like design mining. P_{B^3} and R_{B^3} can be adapted to a downstream task by calculating them specifically for the type of elements of the web page that is relevant for that task. To cover a wide variety of tasks, this paper uses the five types suggested by Kiesel et al.: all pixels (*pixels*), all pixels at visual edges as per an edge detection algorithm in both a coarse (*edges*$_\mathrm{C}$) and fine settings (*edges*$_\mathrm{F}$), all visible DOM nodes (*nodes*), and all textual characters (*chars*).

We provide all code for the evaluation in the repository of this paper, and all generated segmentations as a new data resource (cf. Section 1). In very rare cases (at most 0.2% per algorithm), some algorithms failed (cf. Section 4): in these cases we used the baseline segmentation—a single segment that covers the entire page—as fallback.

4 Algorithms and Parameter Tuning

This section describes the segmentation algorithms that are compared in our experiments, and reports on the results of a corresponding parameter tuning for the algorithms. Table 1 gives an overview of the algorithms in the experiments, which are chosen as representatives for different segmentation paradigms and tasks. For web page segmentation, we evaluate the classic DOM-based VIPS (cf. Section 4.1), the specifically heading-based HEPS (cf. Section 4.2), and the purely visual algorithm of Cormier et al. (cf. Section 4.3). Inspired by the impressive recent advances in image understanding, we also evaluate the performance of one state-of-the-art algorithm of this field for the task of web page segmentation: MMDetection (cf. Section 4.4). Furthermore, as the tasks of web page segmentation is conceptually similar to the task of print document segmentation, we also evaluate the performance of a state-of-the-art approach for that task, the neural network of Meier et al. (cf. Section 4.5). Moreover, we report results for a voting-based ensemble of the algorithms (cf. Section 4.6). To contextualize the results, we include a naive baseline for comparison (cf. Section 4.7). We found that the algorithms do fail for a few web pages, for example, due to a web page's own JavaScript code interfering with the JavaScript code of the segmentation algorithm. As described in Section 3, we use the segmentation of the baseline in this case as a fallback.

Table 1. Overview of the five compared segmentation algorithms with respect to the kind of input documents they were created for, the features they use, and the format of the output segmentation.

Name	Ref.	Document	Features	Output
VIPS	[3]	Web page	Tree, style, location	Rectangle tree
HEPS	[14]	Web page	Tree, style	Node set
Cormier et al.	[6]	Web page	Screenshot	Rectangle tree
MMDetection	[4]	Photo	Screenshot	Pixel masks
Meier et al.	[15]	Article page	Screenshot, text-mask	Mask

4.1 VIPS

The "VIsion-based Page Segmentation algorithm" [3] is the de-facto standard for web page segmentation. Starting from one segment that covers the entire page, VIPS creates a hierarchical tree of segments based on the DOM tree of a web page. The rectangular segments are split based on their so-called degree of coherence, which is computed through heuristic rules based on the tag names, background colors, and sizes of DOM nodes, as well as visual separators: segments are split if their so-called degree of coherence is less than the permitted degree of coherence (PDoC), which is the single parameter of the algorithm. Previous implementations of VIPS rely on web rendering frameworks that are no longer maintained and render modern pages incorrectly. We thus ported one implementation[4] to JavaScript so that every modern browser can run it.

[4]Our port of https://github.com/tpopela/vips_java is available in the code repository of this paper.

For the experiments, we then used the reproduction mode of the Webis-Web-Archiver to have a Chrome browser run VIPS on the web pages as they are re-rendered from the web archives. Though Cai et al. [3] described the degree of coherence to range from 0 to 1, the implementation we ported and thus ours alike use an integer range from 1 to 11, since the heuristic rules suggest the corresponding 11 thresholds. The VIPS algorithm failed for 14 web pages (0.2%) due to rendering errors or due to interference of the web page's and VIPS' JavaScript code.

In a 10-fold cross-validation, the optimal value for PDoC was consistently 6. Figure 1 shows the average number of segments and performance for all values from 1 to 11 over all web pages. As the top graph shows, the number of segments stays almost the same for PDoC from 1 to 6, but increases considerably beyond that. The graphs are very similar for all types of atomic elements, with the notable exception of P_{B^3}—and thus also $F^*_{B^3}$—for *pixels*, which is considerable worse. We discuss this observation in Section 5. Compared to the default value for PDoC of 8 for the original implementation, $F^*_{B^3}$ increases by up to 0.20, which highlights the importance of parameter tuning.

4.2 HEPS

The "HEading-based Page Segmentation algorithm" [14] uses heading detection to identify segments. The authors define a heading as both visually prominent and describing the topic of a segment. HEPS does not solely rely on the HTML heading tags, as the authors found that headings are frequently defined by other means, and that heading tags are frequently used for other purposes. Instead, HEPS identifies headings and their corresponding segments through heuristic rules based on their position in the DOM tree, tag name, font size, and weight. The algorithm first identifies candidate headings using text nodes and images, and after that their corresponding blocks. It then creates a hierarchical segmentation based on the identified blocks. We use the original JavaScript implementation by the authors of the algorithm[5] in the same manner as our reimplementation of VIPS. For consistency with the other algorithms in this comparison, we merge the extracted headings with their associated segments. The HEPS algorithm originally failed for 211 web pages (2.5%) due to rendering errors or due to interference of the web page's and HEPS' JavaScript code, but we were able to reduce this amount to just 5 web pages (0.06%) through slight changes in handling of arrays in the code.

4.3 Cormier et al.

Cormier et al. implement a purely visual algorithm to web page segmentation that uses edge detection to find semantically significant edges, used to synthesize a coherent segmentation [6]. The algorithm takes a screenshot of the web page as input, and therefore does not require to re-render the page. It first calculates for each pixel the probability of a "locally significant edge," which is based on how different the horizontal or vertical image gradients at the pixel are from those of the surrounding pixels. After that, the algorithm composes horizontal and vertical line segments from these edge pixels, up to a maximum length of t_l. Note that the larger t_l, the larger the "gap" that visual

[5]https://github.com/tmanabe/HEPS

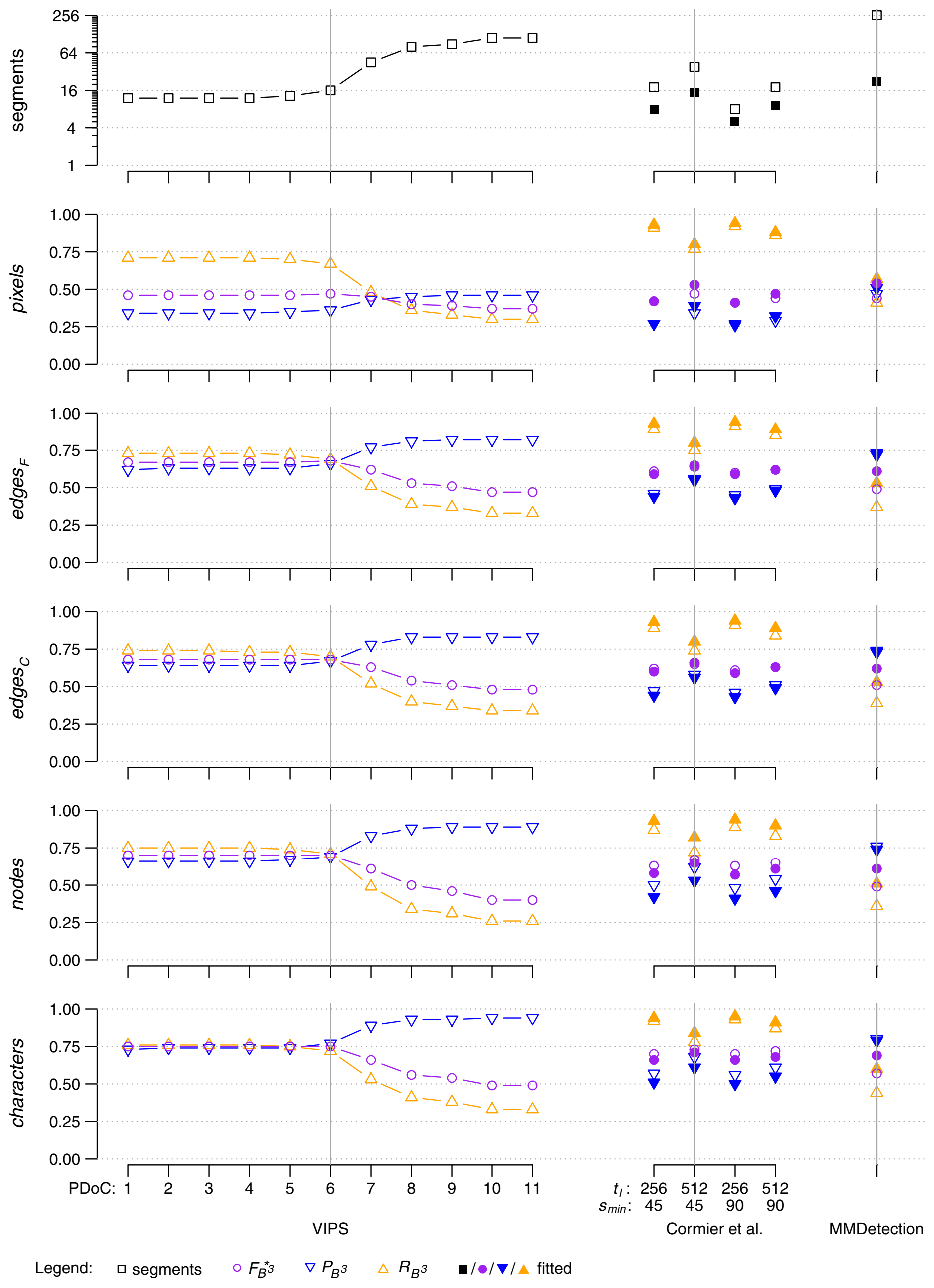

Figure 1. Number of segments (top plot), $F^*_{B^3}$, P_{B^3}, and R_{B^3} for different parameters for the algorithms of VIPS, Cormier et al., and MMDetection. Filled symbols correspond to the values after fitting the segmentation to DOM nodes. The vertical lines show the overall best-performing parameter setting for each algorithm after fitting, as measured by $F^*_{B^3}$.

edges can have to still be considered one line segment. The algorithm then starts with the entire page as one segment, and recursively splits the segments into two by choosing the vertical or horizontal line that is the most "semantically significant," i.e., that has the most and clearest edge pixels. The algorithm stops if there are no semantically significant lines in a segment, or if a split would result in a segment with one side being less than $s_{\min}$ long. The authors thankfully provided us with their implementation for our experiments. The algorithm is computationally expensive, and requires up to 1 hour for the larger web pages of the dataset on a modern CPU, but could likely be sped up considerably through the use of multi-threading and GPUs.

Due to the runtime requirements of the current implementation, we only tested four parameter settings that the original authors suggested to us: each combination of $t_l \in 256, 512$ and $s_{\min} \in 45, 90$. The algorithm contains another parameter t_p that is used as a threshold for determining semantically significant line segments, but we always use $t_p = 0.5$ as suggested by the authors. Figure 1 shows the average number of segments and performance over all web pages. For a fair comparison, we follow Kiesel et al. and fit the visual segmentations to DOM nodes, which has for most cases just a minor effect on the performance, though it does increase $F^*_{B^3}$ for the best parameter setting ($t_l = 512$, $s_{\min} = 45$) for *pixels* by 0.06. This setting is used in our further experiments.

4.4 MMDetection

The Hybrid Task Cascade models [5] from the MMDetection toolbox [4] jointly segment real-world images (photos) and detect objects in them. At the time of our experiments, this algorithm led the MSCOCO [12] detection task leaderboard[6] and can thus be considered state-of-the-art for photo segmentation. The neural network model[7] features an intricate cascading structure. In spot checks, we found that the algorithm detected only segments within images that were included in the web pages. We found that this is due to a separate filtering step that classifies segments as containing real-world objects, so we disabled this step since its purpose does not exist in web page segmentation. Otherwise, the algorithm is the same as the original and no re-training is performed to investigate the similarities of photos and web pages. As segments can be arbitrarily formed in our evaluation setup, we use the corresponding instance segmentation output of the algorithm instead of the more coarse bounding boxes. Like for Cormier et al., we fit the resulting pixel mask segmentation to DOM nodes, which results in performance increases up to 0.12 in $F^*_{B^3}$. MMDetection found no segments for 103 web pages (1%), which we treated like segmentations of one segment that contains the entire page.

4.5 Meier et al.

The convolutional neural network by Meier et al. [15] is state-of-the-art in segmenting digitized newspaper pages. We reimplemented it in contact with the authors,[8] but in-

[6]https://cocodataset.org/#detection-leaderboard

[7]We use the model with X-101-64x4d-FPN backbone and c3-c5 DCN as available and suggested at https://github.com/open-mmlab/mmdetection/blob/master/configs/htc

[8]The authors reported an erratum in their publication to us, so we used the corrected kernel size of 3×3 instead of 5×5 for layers `conv6-1` and `conv7-1`.

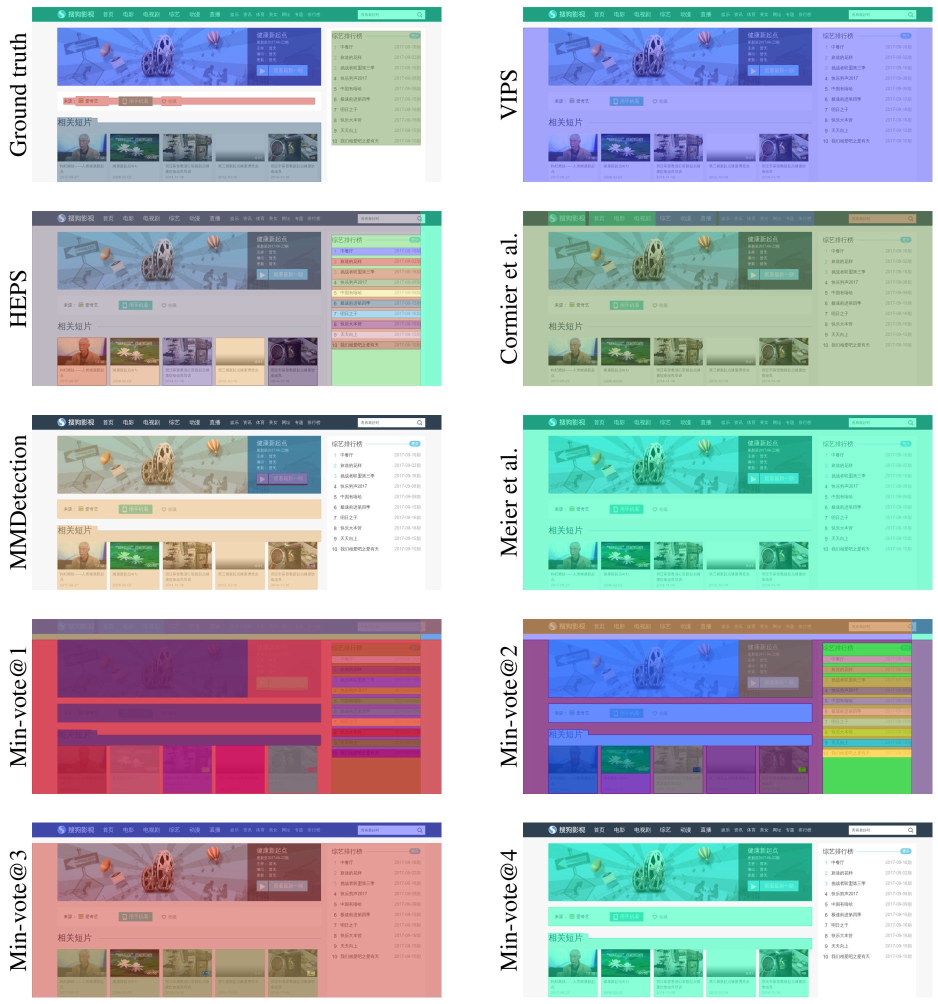

Figure 2. Ground truth and algorithmic segmentations of the top of the same web page.

stead of determining the position of text through optical character recognition (OCR) we use the positions of text nodes from the corresponding list of nodes that accompanies the Webis-WebSeg-20. As the algorithm requires the input to be always of the same size, we had to crop or extend the web page screenshots to a uniform height. As a compromise between extremes, we selected a height that covers about 2/3 of pages, namely 4096 pixels. We then scaled the pages to `256x768` pixels to match the input width of the original approach. Since no pre-trained model is available, we use standard 10-fold cross-validation in the evaluation, and assure that all pages of a website are in the same fold. The training stopped when the loss did not improve for ten consecutive epochs, which led to a training of 20.8 epochs on average.

As the algorithm processes cropped web pages, its results are not fully comparable to those of the other algorithms. For this reason, we report the obtained measurements with some reservations and do not include the segmentations in the ensemble described below. The algorithm found no segments for 4 web pages (0.05%), which we treated like segmentations of one segment that contains the entire page.

4.6 Min-vote@n

We also employ an ensemble of four of these algorithms, excluding the algorithm of Meier et al. as explained above. The ensemble algorithm is identical to the algorithm that was employed to fuse the human annotations to a single ground-truth for the Webis-WebSeg-20 [10]—just treating the algorithms as annotators. To filter out noise, the algorithm first removes all elements from consideration which less than n algorithms placed into segments. After that, the algorithm performs standard classic hierarchical agglomerative clustering, with the similarity of two elements being the ratio of algorithms that placed the elements in the same segment. In line with Kiesel et al., we use a similarity threshold of $\theta_s = \frac{n-0.5}{k}$, where $k = 4$ is the number of algorithms. The algorithm thus tends to put elements in one segment if at least n algorithms did so. We report results for all plausible values for n, namely 1 to 4.

4.7 Baseline

To put the performance of the algorithms into perspective, we report results for the naive approach of segmenting a web page into one single segment. This approach reaches always the maximum recall of 1 at the cost of the lowest possible precision. Both VIPS and the algorithm of Cormier et al. use this segmentation as their starting point.

5 Results of the Comparison

Table 2 shows the performance of each of the algorithms detailed in Section 4 on the Webis-WebSeg-20 dataset (cf. Section 3). The reported values all reflect the results after tuning the respective parameters of the algorithms.

The single algorithms, excluding baseline, Meier et al., and the ensemble, generate between 15.3 and 36.1 segments on average. This difference can be explained by the algorithms working at different levels of granularity. If successful, using more segments should increase the P_{B^3}. However, this is not necessarily the case: Though HEPS is clearly working at a finer level of granularity than VIPS (cf. Table 2 and Figure 2), both algorithms perform similar in terms of P_{B^3}.

The highest $F^*_{B^3}$ scores are reached for *chars* and the smallest for *pixels*. This difference is likely due to web page segmentation algorithms being developed for information extraction purposes mainly, and thus mostly optimized for text. However, for applications like design mining, even the spacing between elements needs to be segmented correctly. New algorithms will be required for such and similar downstream tasks.

Table 2. Average number of segments per web page and evaluation results for each discussed algorithm on the Webis-WebSeg-20 dataset (Baseline, VIPS, HEPS, Cormier et al., MMDetection, Meier et al., and the Min-vote@n ensembles): average F_1-score (F_{B^3}), precision (P_{B^3}), recall (R_{B^3}), as well as the harmonic mean of the averaged precision and recall ($F^*_{B^3}$) for each type of atomic elements. The ground truth contains 9.1 segments on average. The highest score in each row (excluding the baseline) is highlighted in bold. The results of Meier et al. are shown in gray as its evaluation is not fully comparable.

Measure		**Baseline**	**VIPS**	**HEPS**	**Corm.**	**MMD.**	**Meier**	**MV@1**	**MV@2**	**MV@3**	**MV@4**
Segments		1.0	16.1	36.1	15.3	23.0	4.6	6.5	18.7	36.5	69.5
pixels	F_{B^3}	0.24	0.38	0.33	0.36	**0.42**	0.32	0.30	0.39	0.30	0.28
	$F^*_{B^3}$	0.28	0.47	0.44	0.53	**0.54**	0.50	0.35	0.50	0.45	0.42
	P_{B^3}	0.16	0.36	0.36	0.39	0.51	0.48	0.22	0.38	0.60	**0.68**
	R_{B^3}	1.00	0.67	0.56	0.80	0.57	0.52	**0.96**	0.72	0.36	0.30
edges$_\mathrm{F}$	F_{B^3}	0.44	**0.59**	0.48	0.51	0.53	0.41	0.50	0.56	0.39	0.34
	$F^*_{B^3}$	0.49	**0.68**	0.58	0.65	0.61	0.55	0.56	0.66	0.49	0.45
	P_{B^3}	0.32	0.66	0.61	0.55	0.73	0.55	0.40	0.61	0.81	**0.87**
	R_{B^3}	1.00	0.69	0.55	0.80	0.53	0.55	**0.96**	0.71	0.36	0.30
edges$_\mathrm{C}$	F_{B^3}	0.45	**0.61**	0.49	0.53	0.54	0.42	0.51	0.57	0.39	0.35
	$F^*_{B^3}$	0.49	**0.68**	0.59	0.66	0.62	0.56	0.56	0.67	0.50	0.46
	P_{B^3}	0.32	0.67	0.62	0.56	0.74	0.55	0.40	0.63	0.82	**0.88**
	R_{B^3}	1.00	0.70	0.56	0.80	0.53	0.57	**0.96**	0.72	0.36	0.31
nodes	F_{B^3}	0.42	**0.63**	0.43	0.52	0.52	0.44	0.49	0.54	0.34	0.31
	$F^*_{B^3}$	0.46	**0.70**	0.54	0.65	0.61	0.56	0.55	0.65	0.44	0.42
	P_{B^3}	0.30	0.69	0.63	0.53	0.74	0.52	0.38	0.64	0.85	**0.88**
	R_{B^3}	1.00	0.71	0.46	0.82	0.51	0.61	**0.96**	0.65	0.29	0.27
chars	F_{B^3}	0.52	**0.67**	0.50	0.61	0.61	0.50	0.59	0.62	0.40	0.39
	$F^*_{B^3}$	0.57	**0.75**	0.60	0.71	0.69	0.61	0.64	0.71	0.50	0.49
	P_{B^3}	0.39	0.77	0.73	0.61	0.79	0.59	0.48	0.72	0.90	**0.92**
	R_{B^3}	1.00	0.72	0.51	0.84	0.60	0.63	**0.96**	0.71	0.35	0.33

Conversely, the results differ only marginally between *edges*$_\mathrm{F}$ and *edges*$_\mathrm{C}$, despite the visually very different edge detection [10]. This result is very convenient for future evaluations, as it indicates that (1) the parametrization of the edge detector does not play a major role, and (2) it is sufficient to evaluate for one parametrization of the edge detector. We recommend to employ *edges*$_\mathrm{F}$ in the future, as it produces fewer segments that have no edges and which are thus not considered in the evaluation.

The best-performing algorithm from the literature for most types of atomic elements is the VIPS algorithm, reaching a $F^*_{B^3}$ of up to 0.75 and convincingly beating the baseline in all cases. It thus comes closest to human annotators—and also relatively close in terms of the average number of segments, which is 9.1 for the ground-truth. Moreover, for a higher value of PDoC it can reach a very high P_{B^3} of up to 0.94 for *chars* (cf. Figure 1), which is close to human agreement (cf. [10]). Therefore, PDoC can indeed be used to adjust the level of segmentation granularity. Nevertheless, P_{B^3} is considerably lower at the optimal value for PDoC, which suggests that VIPS can benefit from an adaptation of PDoC to the (part of the) web page at hand. Though VIPS performs similarly well for most types of atomic elements, its precision is rather low for *pixels*. This difference is likely due to background pixels on the left and right of the actual content of the web pages: whereas VIPS includes such pixels in the segments, the human annotators did not (cf. Figure 2 for one example).

However, both the algorithm by Cormier et al. and MMDetection reach a similar performance to VIPS in terms of $F^*_{B^3}$, which demonstrates the viability of purely visual approaches to web page segmentation. By comparison, the algorithm by Meier et al. fails to compete with the other algorithms, even though it had a clear advantage over the other algorithms by being trained on the data. Its poor performance might be due to the required adjustment of the input screenshots.

The results for the min-vote ensembles show that even a basic voting scheme can be employed to efficiently fuse the output of different algorithms. Remarkably, Min-vote@2 reaches a $F^*_{B^3}$ scores very similar to those of VIPS. Like PDoC for VIPS, the parameter n here fulfills the role of selecting the desired level of granularity. This is especially helpful as some algorithms, like HEPS, do not have such a parameter. The ensemble therefore allows to incorporate the HEPS heuristic (and others without such a parameter) and still to select a level of granularity.

A special ensemble is that of Min-vote@4, which puts elements in one segment if and only if all four single algorithms did so. We want to highlight that P_{B^3} is about 0.9 for all types of atomic elements except *pixels*, which indicates that most segments of these types of elements are indeed separated from others by at least one of the algorithms. However, *pixels* are an exception here, which shows a deficit that needs to be addressed by future algorithms.

6 Conclusion

As we contrast and discuss the results of our evaluation for each type of atomic page elements, it becomes clear that the classical VIPS algorithm is still the overall best option, unless the downstream task requires pixel-based segments. In that case, purely visual page segmentation performs better, whereas otherwise it is a close second to VIPS. MMDetection performed especially well for being designed and trained for photographic images. Interestingly, the state-of-the-art approaches for such images as well as for newspaper page segmentation both employ deep learning, while the approaches for web page segmentation rely mostly on hand-crafted heuristics and observations. We believe that this difference mainly stems from the fact that no large-scale datasets for web page segmentation have been available in the past. With this paper, we lay the foundation for the development of new approaches that may improve over the long-standing, yet heretofore unknown champion, VIPS.

Acknowledgements

We thank the anonymous reviewers for their helpful comments and the authors of the respective algorithms for providing us with either their code and/or their support for our re-implementations, as stated in the respective section.

Bibliography

[1] Amigó, E., Gonzalo, J., Artiles, J., Verdejo, F.: A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information Retrieval **12**(4), 461–486 (2009), https://doi.org/10.1007/s10791-008-9066-8

[2] Arias, J., Deschacht, K., Moens, M.F.: Language independent content extraction from web pages. In: Proceedings of the 9th Dutch-Belgian information retrieval workshop, pp. 50–55, University of Twente; Enschede, The Netherlands (2009)

[3] Cai, D., Yu, S., Wen, J., Ma, W.: Extracting content structure for web pages based on visual representation. In: Web Technologies and Applications, 5th Asian-Pacific Web Conference, APWeb 2003, Xian, China, April 23-25, 2002, Proceedings, pp. 406–417 (2003), https://doi.org/10.1007/3-540-36901-5_42

[4] Chen, K., an Jiangmiao Pang, J.W., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., Zhang, Z., Cheng, D., Zhu, C., Cheng, T., Zhao, Q., Li, B., Lu, X., Zhu, R., Wu, Y., Dai, J., Wang, J., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: MMDetection: Open mmlab detection toolbox and benchmark. CoRR **abs/1906.07155** (2019), URL http://arxiv.org/abs/1906.07155

[5] Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., Loy, C.C., Lin, D.: Hybrid task cascade for instance segmentation. CoRR **abs/1901.07518** (2019), URL http://arxiv.org/abs/1901.07518

[6] Cormier, M., Mann, R., Moffatt, K., Cohen, R.: Towards an improved vision-based web page segmentation algorithm. In: 14th Conference on Computer and Robot Vision, CRV 2017, pp. 345–352 (2017), https://doi.org/10.1109/CRV.2017.38

[7] Cormier, M., Moffatt, K., Cohen, R., Mann, R.: Purely vision-based segmentation of web pages for assistive technology. Computer Vision and Image Understanding **148**, 46–66 (2016), https://doi.org/10.1016/j.cviu.2016.02.007

[8] Goldstein, E.B.: Sensation and Perception. Cengage Learning, 8 edn. (2009), ISBN 9780495601494

[9] Kiesel, J., Kneist, F., Alshomary, M., Stein, B., Hagen, M., Potthast, M.: Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. Journal of Data and Information Quality (JDIQ) **10**(4), 17:1–17:25 (Oct 2018), https://doi.org/10.1145/3239574, URL https://dl.acm.org/authorize?N676358

[10] Kiesel, J., Kneist, F., Meyer, L., Komlossy, K., Stein, B., Potthast, M.: Web Page Segmentation Revisited: Evaluation Framework and Dataset. In: d'Aquin, M., Dietze, S., Hauff, C., Curry, E., Cudré-Mauroux, P. (eds.) 29th ACM International Conference on Information and Knowledge Management (CIKM 2020), pp. 3047–3054, ACM (Oct 2020), https://doi.org/10.1145/3340531.3412782

[11] Kumar, R., Satyanarayan, A., Torres, C., Lim, M., Ahmad, S., Klemmer, S.R., Talton, J.O.: Webzeitgeist: design mining the web. In: Mackay, W.E., Brewster, S.A., Bødker, S. (eds.) 2013 ACM SIGCHI Conference on Human Factors in Computing Systems, CHI '13, pp. 3083–3092, ACM (2013), https://doi.org/10.1145/2470654.2466420

[12] Lin, T., Maire, M., Belongie, S.J., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft COCO: common objects in context. In: Computer Vision - ECCV 2014 - 13th European Conference, pp. 740–755 (2014), https://doi.org/10.1007/978-3-319-10602-1_48, http://cocodataset.org/

[13] Ma, L., Goharian, N., Chowdhury, A.: Automatic data extraction from template generated web pages. In: Arabnia, H.R., Mun, Y. (eds.) Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA '03, pp. 642–648, CSREA Press (2003)

[14] Manabe, T., Tajima, K.: Extracting logical hierarchical structure of HTML documents based on headings. PVLDB **8**(12), 1606–1617 (2015)
[15] Meier, B., Stadelmann, T., Stampfli, J., Arnold, M., Cieliebak, M.: Fully convolutional neural networks for newspaper article segmentation. In: Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on, vol. 1, pp. 414–419, IEEE (2017)
[16] Nielsen, J., Pernice, K.: Eyetracking Web Usability. Pearson Education (2010), ISBN 9780321714077
[17] Zeleny, J., Burget, R., Zendulka, J.: Box clustering segmentation: A new method for vision-based web page preprocessing. Inf. Process. Manage. **53**(3), 735–750 (2017), https://doi.org/10.1016/j.ipm.2017.02.002