Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Computer Science and Media

Large-scale Analysis and Comparison of Web Page Segmentation Approaches

Master's Thesis

Lars Meyer
Born Nov. 29, 1995 in Sondershausen

Matriculation Number 114719

1. Referee: Prof. Dr. Benno Stein
2. Referee: PD Dr. Andreas Jakoby

Submission date: December 20, 2019

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, December 20, 2019

. . . . . .
Lars Meyer

Abstract

Within the past two decades, several algorithms that tackle the task of web page segmentation have been proposed, pursuing different strategies based on the information used in the segmentation process. Due to the lack of both a suitably large and varied dataset as well as well-founded evaluation methodology, previous assessments of the performance of these algorithms were largely ad hoc evaluations performed on small datasets within the context of the specific use case they were created for. This thesis contributes an evaluation of five web page segmentation algorithms from the literature on *Webis Web Segments 2020*, the largest publicly available dataset of human-annotated web page segmentations to date, using precise metrics from clustering theory. We thereby improve the understanding of these existing approaches and their performance, and lay groundwork for further research in the field of web page segmentation. Our results show that (1) even the oldest examined algorithm, using a traditional DOM-based approach, still provides good performance on modern web pages; (2) a more recent approach, based on a radically different principle of purely visual segmentation, manages to approach this performance, positioning this category of algorithms as a viable alternative; (3) a parameter analysis performed for two of the approaches enabled significant performance improvements, indicating the importance of understanding the existing parameters and their influence on the resulting segmentation; (4) the combination of purely or predominantly visual segmentation algorithm results with DOM information harbors potential for further improvements in segmentation quality; and (5) employing a simple voting scheme for combining the best results obtained from the examined algorithms yields the overall best results out of all performed evaluations, establishing this approach as an effective way of obtaining high quality segmentations from multiple algorithms.

Contents

Acknowledgments

For their help in the creation of this thesis, I would like to thank:

- my supervisor Johannes Kiesel and the co-authors of the paper "Web Page Segmentation from First Principles" (2019) for providing the foundation for this thesis;
- Michael Cormier for providing an implementation of the edge-based segmentation approach presented in Cormier et al. [2017];
- Tomáš Popela for his Java implementation of VIPS [Popela, 2012] and his assistance in clarifying its licensing for our port;
- Florian Kneist for providing his implementation [Kneist, 2019] of the neural network approach presented by Meier et al. [2017];
- and my friends Theresa Elstner and Sebastian Höfer for proofreading and helpful comments.

Personal thanks:

I thank my family for their unconditional support, all the great friends I made during my Master's, and most of all my love Shabnam.

Chapter 1

Introduction

Web page segmentation has been researched and applied for a variety of uses in well over 1000 publications[1] to date, but still the existing approaches in the field have not been evaluated against a large annotated dataset for a comparative analysis of their performance. This is because the existing algorithms were created with a specific use case in mind, such as information retrieval or accessibility enhancements, and they were only evaluated in this context. While previous quantitative comparisons of web page segmentation approaches exist in the literature, our review of them following this introduction reveals flaws affecting interpretability, comparability and reproducibility of their results; we improve on these points in this thesis. This means that general, independent observations about the strengths and weaknesses of existing approaches, as well as meaningful performance figures and quality assessments, are lacking, a fact we aim to amend with this work.

As a first step for an evaluation, it is necessary to define what segments and segmentation actually are. The definition of a segment that this thesis follows is formulated by Kiesel et al. [2019]:

> A segment is a part of a web page containing the elements that belong together visually, semantically, and in purpose.

Figure 1.1 shows an example of a web page segmentation. Most of the segments present in this example fulfill the above definition, but the segmentation of the header (**1**) may be contentious: here, it is separated into three different segments. It is difficult to decide whether these segments are all visually, semantically and purposely coherent by themselves, or whether they should be part of one larger segment encompassing the entire header. The segmentation presented in the example is a ground truth segmentation from our dataset, meaning that it was combined from segmentations performed by five different

[1] Number obtained from Google Scholar using the query "web page segmentation".

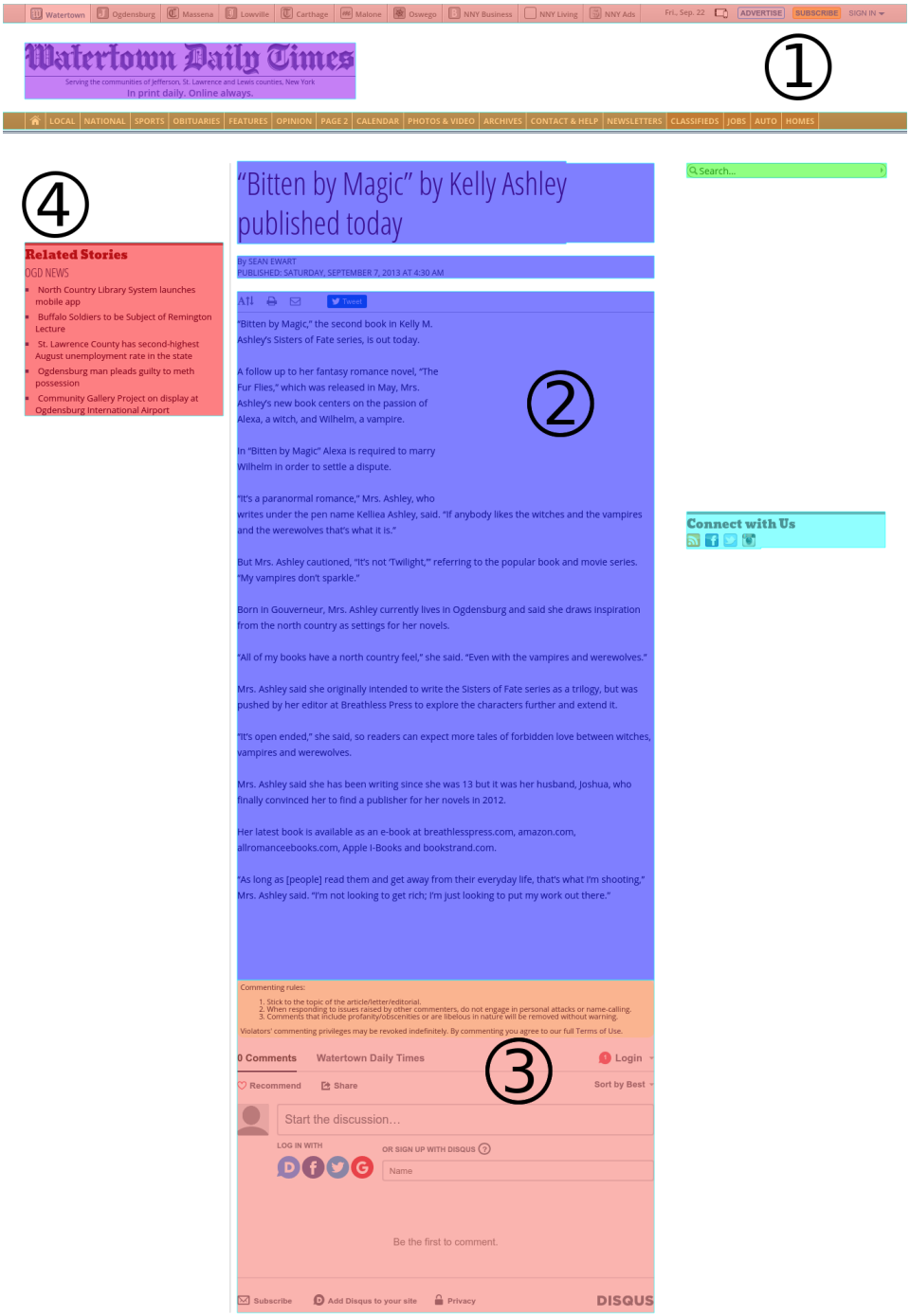


Figure 1.1: An example of a web page segmentation for a news article. The **header** (**1**) of the page consists of two navigation bars (light red and dark yellow) and the site's title (purple). The **main content** (**2**) is marked in blue. A **comment area** (**3**) is present at the bottom (light red). The left pane contains a section referencing **related articles** (**4**, red).

annotators. Thus, while Kreuzer et al. [2015] observe that "human beings are very good at partitioning: even if a website is in a language we are not familiar with, it is clear to us what is an advertisement, what is a menu, and so on", the visual, structural and semantic cues that humans use to determine the coherent semantic units a web page is made up of may not always be interpreted in the same way. The given definition therefore does not necessitate the existence of one true segmentation for a web page.

This is the origin of much of the task's difficulty, and the reason why a ground truth fused from *multiple* human annotations per page is best suited to evaluate segmentation quality. While web page segmentation algorithms attempt to use the same cues as humans to obtain a segmentation, the information that they use to identify these cues and their interpretation of them varies, requiring evaluation as performed in this thesis to determine which approach comes closest to human performance. Some algorithms use information from the web page's source code and the rendering engine (usually a web browser), which creates the visual representation of the web page that visitors interact with and stores the elements that make up the page as a DOM[2] tree. The hierarchy of elements and information about their visual appearance (their *style*) contained therein is used by such algorithms to identify the aforementioned cues. Other algorithms only use the rendered appearance of a web page in the form of a screenshot, and segment purely visually.

To further elaborate on the difficulty of web page segmentation, we consider other forms of media for which segmentation is performed. These are documents, such as print media, and real-world images. Web page segmentation can be seen as an extension of document segmentation. While the two tasks may seem superficially similar, web page segmentation is the more general of these challenges, due to the presence of dynamic media and more varied layouts and styles in web pages compared to documents. Segmentation is also performed on real-world images (for example using the MMDetection toolkit by Chen et al. [2019a] which is evaluated in this thesis); this is commonly referred to as "instance segmentation". In terms of complexity, Cormier et al. [2017, sections I, II] consider web page segmentation to be the middle ground between document and real-world image segmentation.

Despite the aforementioned plethora of information potentially available to a web page segmentation algorithm, it has no knowledge of how the author of the web page intended to form the coherent visual and semantic units on the page. The page's HTML source code is only semi-structured and does not accurately represent the intended layout, which is in most cases significantly influenced by the associated CSS stylesheet(s) and extended by JavaScript code.

[2] *Document Object Model*; technical specification at `https://www.w3.org/TR/domcore/`

Similarly, while the DOM tree represents the hierarchical structure of elements on the web page, it contains no information about the intended grouping of elements. Recently, some effort has been made to introduce the concept of (labeled) segments of a web page into its source code with a focus on semantics and accessibility. The HTML5 specification includes a new set of tags specifically meant to convey meaning starting from the page source (O'Hara et al. [2018, see e.g. section 3.2.1]). They are meant to enable new forms of machine processing on these documents. Figure A.1 shows an example of how this may aid segmentation, if used correctly. Similarly, *ARIA*[3] *roles* can help with this [see Faulkner, 2018]. In our dataset, 58.1% of pages implement one or more HTML5 semantic tags; 56.2% of pages contain one or more ARIA roles. However, this does not guarantee that these tags are used consistently, or that they are employed with a focus on delineating the coherent semantic units on the page. Using web page segmentation has been proposed for accomplishing the reverse: augmenting existing web pages with HTML5 semantic tags (see Section 2.1).

We have established the difficulty of the task of web page segmentation, and briefly mentioned approaches that segmentation algorithms take to tackle this problem. In this thesis, we evaluate five existing algorithms from the literature on the *Webis Web Segments 2020* dataset, consisting of 8490 web pages with ground truths created from 42450 crowd-sourced segmentations. We analyze the results of this evaluation in connection with the algorithms' operating principles, which gives a clear picture of the beneficial and detrimental aspects of their design. Additional experiments are performed to show the available potential for increasing segmentation quality further using the existing algorithms.

Following this introduction establishing the task complexity and necessity of evaluation, we give an overview of related work in the field and provide an extensive review of previous web page segmentation evaluations in Chapter 2. Afterwards, we give a detailed description of the five algorithms selected for analysis in this thesis, as well as the implementations we used, in Chapter 3. We then present our evaluation methodology and details about the dataset in Chapter 4. Chapter 5 continues with detailing the experiments performed for this thesis, and presents their results including analyses. Finally, we summarize the achieved results in Chapter 6 and provide identified and suggested areas for future work.

The impetus for this thesis was given by the paper "Web Page Segmentation from First Principles" [Kiesel, Potthast, Stein, Komlossy, Kneist, and Meyer,

[3] *Accessible Rich Internet Applications*

2019]. In addition to an updated and extended version of the evaluation performed in that paper, this thesis contributes (1) a methodological comparison to previous evaluations through literature review; (2) comprehensive overviews of the evaluated algorithms, in terms of design aspects and inner workings, and details about the employed implementations; (3) a new experiment regarding choices of atomic elements for the employed clustering similarity measure; (4) a comparison of segmentation similarity between the examined algorithms; (5) newly performed parameter analyses for two algorithms; (6) a new experiment concerning the combination of purely or predominantly visual segmentation algorithms with web page DOM information; (7) the evaluation of algorithm segmentation fusion with improved results from the new experiments in this thesis; and (8) detailed and exhaustive interpretations and comparative analyses of all obtained results, both in the context of segmentation quality among all evaluated algorithms, as well as within the specific experiment contexts.

Chapter 2

Related work

In the following, we present an overview of use cases that web page segmentation has been applied for in the literature, to demonstrate the plethora of fields that this technology is useful for, and to underscore the importance of an improved understanding of existing approaches. Furthermore, we perform an in-depth review of prior works in the evaluation of web page segmentation algorithms. We focus on reviewing the methodologies used for quantitative evaluations: in independent comparisons, and in evaluations performed by the algorithms' authors. Those algorithms mentioned here that will be used in our experiments are further detailed in their respective sections in Chapter 3.

2.1 Use cases for web page segmentation

Automatic web page segmentation is useful in many areas. Before the widespread adoption of smartphones, which are now being specifically targeted in web page design, automatically re-arranging the structure of web pages for more usable display on small screens was a typical field of application for web page segmentation algorithms, as shown for example in the works of Baluja [2006], Hattori et al. [2007] and Yang and Shi [2009].

A more contemporary use case is presented as their primary motivation by Cormier et al. [2017], who propose their segmentation approach, which is also examined in this thesis, as useful for increasing the accessibility of web pages for people who depend on using screen readers due to visual impairments. Akpinar and Yesilada [2013] also consider this an important field of use. Sanoja and Gançarski [2017] show an approach of using automatic segmentation on web archives to transform them from previous HTML standards to HTML5, with a specific focus on finding semantic elements introduced in the new standard.

Many publications use web page segmentation for information retrieval purposes. It is a primary motivation for the authors of the VIPS algorithm

(Cai et al. [2003a,b]), which we evaluate in this thesis, as well as Feng et al. [2016]. More specifically, summarization of content is a use case considered important by Pasupathi et al. [2012] and Manabe and Tajima [2015], the latter of whom propose the HEPS algorithm which is analyzed in this work. Yang and Shi [2008] use segmentation as part of learning the function of each identified block in the page's context. Page classification and page ranking are further fields of use for which automatic segmentation can be beneficial. For example, Kovacevic et al. [2002] and Bing et al. [2014] pursue the use of segmentation to improve classification, while ranking is considered as a use case by Chakrabarti et al. [2008] and Fernandes et al. [2011]. Kuppusamy and Aghila propose the possibility of obtaining more fine-grained usage statistics by identifying the precise segment in which user interaction occurred (2012a). Furthermore, they present a method of augmenting search results by extracting additional relevant information (2012b).

Automatic segmentation is also considered useful for web page similarity assessment. Cao et al. [2010] use it for detecting phishing attempts, while Saad and Gançarski [2010] and Law et al. [2012] propose its use in web archiving, applying it for detecting changes on pages over time.

Lastly, segmentation has been used by Wu et al. for measuring visual properties of a web page: its visual quality (2011) and its visual complexity (2013).

2.2 Evaluations and comparisons

In this section, we give an overview of previous works comparing web page segmentation approaches and algorithms, both qualitatively and quantitatively, as well as the evaluations performed by the authors for some of the algorithms presented in this thesis. For the quantitative comparisons and the authors' evaluations, we detail the methods used and the obtained results; where sensible, we reference our results for comparison. We also compare some methodological aspects to our approach in the following; we make additional references to them in describing our methodology in Chapter 4. Our findings underscore the importance of well-founded methods for measuring segmentation similarity, the importance of appropriately large and varied datasets, and the importance of a consistent evaluation environment including dataset reproduction, algorithm execution and calculation of evaluation results.

Yesilada [2011] provides an extensive *qualitative* review of web page segmentation approaches from the literature. The author details the terms and definitions used in the literature, the motivations and purposes of existing approaches, and their operation principles and assumptions, and discusses the

evaluations performed by the authors of the respective approaches. This work stands out due to the large number of approaches that are presented, and the structured way of examining them. However, because no independent evaluation was performed, no performance comparison is given, and thus the presented approaches are not regarded outside of the context of their original purpose and the evaluations undertaken within that context.

2.2.1 Review of quantitative comparisons

Previous quantitative comparisons of web page segmentation approaches can be found in Kreuzer et al. [2015] and Sanoja and Gançarski [2015].

Kreuzer et al. [2015] compare three approaches: *PageSegmenter*, an HTML-only approach presented by Vadrevu et al. [2005], *VIPS* [Cai et al., 2003a,b], and *Block Fusion*, an approach based on text density presented by Kohlschütter and Nejdl [2008]. Their dataset consists of 70 "popular" and 82 "random" pages. The "popular" pages were obtained as "the top 10 pages from the 10 top-level categories from the directory site `http://dir.yahoo.com/`" (directory now defunct), while the "random" pages were obtained from 100 links generated by an online service providing random websites from a database of "over four million pages" at the time of Kreuzer et al.'s writing. Rendering errors resulting from their archival process caused some pages to be removed from their dataset, leading to the previously stated final dataset size.

To obtain a ground truth, all pages from their dataset were segmented by three annotators. The process was performed in a browser reproducing the archived web page, which ran a JavaScript tool by the authors used for classifying DOM nodes. With this tool, the user graphically selects a DOM node on the rendered web page as a segment, similar to the "Inspect Element" functionality offered by modern browsers. The authors specify two categories for classification, along with what elements these categories correspond to. A block was to be classified as a "High-level block" if it is one of "Header, Footer, Content, Sidebar." A block was to be classified as a "Sub-level block" if it is one of "Logo, Menu, Title, Link-list, Table, Comments, Ad, Image, Video, Article, Searchbar." The authors claim that the resulting two-level segmentation is "sufficient in the majority of cases" in terms of the level of segmentation granularity. Compared to the annotation methods employed by Kiesel et al. [2019], we find these guidelines to be inflexible; it may not be possible to apply these categories to all types of web pages, and annotators may find themselves to be drawn away from their intuitive segmentation by having to follow these rules. Additionally, the restriction to DOM nodes as segments as opposed to the free rectangular segmentation employed by Kiesel et al. may further

influence the annotator, suggesting a relationship between elements that the annotator may have intuitively understood differently.

Furthermore, the authors touch on the subject of annotator agreement, but state that they did not measure it as part of the creation of their ground truth. Instead, they cite results from Vadrevu et al. [2005], where an agreement of 87.5% between eight annotators was claimed, and state that this "corresponds to our own anecdotal experience."

As comparison metrics, Kreuzer et al. use string matching measures in two flavors, "exact" and "fuzzy". They extract the raw text, free from HTML tags, white space and line breaks, for each segment. For the "exact" metric, the strings between algorithm and ground truth segments must be identical. For the "fuzzy" metric, the strings between algorithm and ground truth segments must match for at least 80% of the characters. These metrics are most comparable to the *chars* clustering measure proposed by Kiesel et al. [2019], but significantly less flexible. It is possible to calculate much more accurate precision and recall values when identifying *how much* of the text content of a segment matches, instead of merely whether or not they match (possibly with some tolerance).

To discuss the findings from Kreuzer et al. [2015], we focus on their results of comparing VIPS to their ground truth, since we perform the same comparison in this thesis, albeit with significant differences in dataset size and evaluation methodology.

To evaluate VIPS, they use the Java implementation by Popela [2012], which is also used as the basis of our port employed in this thesis. The authors do not state the *Permitted Degree of Coherence* (PDoC, granularity setting, see Section 3.1) that was used for the segmentations produced by VIPS. This significantly hampers comparability of their results to ours, although we might estimate the value that they used from the number of segments that they state. For the "popular" dataset, the authors state that VIPS produced 19.51 segments on average; for the "random" dataset, they state that VIPS produced 9.24 segments on average. Since our dataset contains a mix of what Kreuzer et al. would call "popular" and "random" web pages, we obtain a combined average number of segments for VIPS by summing the average segment numbers for the "popular" and "random" parts of the dataset weighted by the respective share of pages of the total dataset. Thus, we obtain $19.51 \cdot \frac{70}{152} + 9.24 \cdot \frac{82}{152} = 13.96$ average segments per page for VIPS across the authors' entire dataset.

According to our results shown in Table 5.7, a PDoC value of 5 comes closest in terms of average segment count for our dataset. For reference, we present our results of evaluating VIPS against our ground truth with PDoC 5 for the *chars* clustering measure in Table 2.1. The same calculation for obtaining the combined average number of segments from Kreuzer et al. [2015]

	P	R	F
"exact"	0.25	0.18	0.18
"fuzzy"	0.36	0.24	0.24

	chars		
	P_{B^3}	R_{B^3}	F_{B^3}
This work	0.74	0.76	0.68

Table 2.1: Weighted averaged results of Kreuzer et al.'s evaluation of VIPS against their ground truth (left), compared to our results for clustering measure *chars* (right). The PDoC value used by Kreuzer et al. is not known; we present our results for PDoC 5 due to the similar average number of segments per page.

was applied to their precision, recall, and F-score values.

A significant difference between Kreuzer et al.'s and our results is immediately obvious. We claim that this is mostly due to the difference in metrics; while Kreuzer et al. check exact string matches or matches $> 80\%$, the *chars* measure employed here calculates precision and recall based on concrete amounts of matched characters. Effectively, the metrics employed by Kreuzer et al. set the precision of a segment to 1 if the contained strings match exactly or with up to 20% tolerance, and 0 otherwise. This becomes apparent with the jump in VIPS' performance in the "fuzzy" metric versus the "exact" metric, suggesting that there are many cases where VIPS is not far off from the ground truth; in fact, this jump occurs for all algorithms evaluated by Kreuzer et al., but there is no way of learning from these numbers the exact threshold at which performance increases substantially.

Another factor is the difference in average segment counts per page, which is 9.1 for our ground truth and 14.02 for the authors' ground truth (weighted sum of average ground truth segment counts as above). This means their ground truth is more fine-grained on average, causing VIPS to fall just below the ground truth in terms of average segment count per page. We never encountered such a scenario in our evaluations, as the lowest average segment count per page that we obtained for VIPS is 12.20 for PDoC 1.

The more fine-grained segmentation of the annotators in Kreuzer et al. [2015] may be a consequence of the guidelines they were given. It is conceivable that some elements would not have been naturally regarded as a separate segment by the annotators had they not been told so, for example in the case of titles, which may have otherwise been part of a segment with the content they belong to.

Sanoja and Gançarski [2015] compare their own *Block-o-Matic* approach [Sanoja and Gançarski, 2013] as well as *Block Fusion* [Kohlschütter and Nejdl, 2008] and both the reference implementation of *VIPS* [Cai et al., 2003a,b] and the Java implementation by Popela [2012] to a ground truth created by an

unknown number of human annotators.

Their dataset (titled the "*GOSH collection*", from **Go**ogle **S**earc**h**) contains 125 pages. Its composition is based on "The Conversation Prism"[1] by American digital analyst Brian Solis, which is a categorization of social networking websites based on their predominant use (for example "Reviews & Ratings" or "Messaging"). The authors chose 25 sites each from the categories "Blog, Forum, Picture, Enterprise, and Wiki". They state that the choice of sites was made based on their *PageRank* "using Google search." The authors further say that "within each of those sites, one page is crawled", but do not go into detail of how the page selection (as opposed to the site selection) was performed.

To annotate the dataset, the authors developed a tool they call "*MoB* (Manual-design of Blocks)", which is similar to the tool used in Kreuzer et al. [2015], but does not label the segments. As with the tool used by Kreuzer et al., the annotator graphically selects a DOM node on the rendered web page that they think corresponds to a segment. Likewise, we note that the restriction to segmenting using DOM nodes may influence the annotator and distract from their intuitive segmentation.

The authors measure segmentation quality in terms of "block correspondence" and "text coverage". For block correspondence, they "build a weighted bipartite graph called *block correspondence graph (BCG)*", in which the segments from the ground truth and the algorithm are connected based on their correspondence. If multiple algorithm segments are contained in one ground truth segment, all corresponding nodes of contained algorithm segments will be connected to the ground truth node; the reverse is true if multiple ground truth segments are contained in one algorithm segment. The weights of edges connecting segmentation nodes are set by the metrics "*htmlcover*" and "*textcover*"; *textcover* is the proportion of how much of the raw text of the larger containing segment is covered by a smaller contained segment, and *htmlcover* is a measure of how many of the larger containing segment's HTML elements are covered by a smaller contained segment.

Based on this graph, the authors count "total correct segmentations" T_c (number of identical segments between both segmentations, with a matching threshold t_r), "oversegmented blocks" C_o (segments that are further subdivided by the algorithm, but not in the ground truth), "undersegmented blocks" C_u (segments that are further subdivided in the ground truth, but not in the algorithm segmentation), "missed blocks" C_m (segments present in the ground truth but not in the algorithm segmentation), and "false alarms" C_f (segments present in the algorithm segmentation but not in the ground truth).

As with our discussion of the results from Kreuzer et al. [2015], we focus

[1] `https://conversationprism.com/`

on Sanoja and Gançarski's evaluation of VIPS for comparing methodologies. The authors do not state the PDoC value they used for evaluating VIPS. As opposed to Kreuzer et al. [2015], they also do not explicitly report average segment counts per page. The authors merely state that they aimed to "detect blocks of medium size", further saying that "we focus neither in detecting only large blocks, such as header, menu, footer and content, nor in detecting blocks at a too high level of detail (sentences, links or single images)." However, it is possible to obtain an average number of segments per page by summing up the values for T_c, C_o, C_u, C_m and C_f. For the VIPS reference implementation, the thereby obtained average number of segments per page is 6.82; for the Java implementation, it is 10.89. The exact reason for the large difference between implementations is unclear; we consider the underlying rendering engine to be a likely significant factor (Internet Explorer for the VIPS reference implementation, CSSBox for the Java implementation). The average number of segments per page for the Java implementation is lower than any of our values (see Table 5.7), which we attribute to the different underlying rendering engine and the much greater size and diversity of our dataset.

The authors give results in terms of block correspondence and text coverage of the whole page, that is, how much of the entire page's text is covered by the segmentation. Due to the different measures used in this thesis, we cannot directly compare their results of block correspondence to any of our results. The text coverage results are given in percent of the entire page's text covered by segmentations. The reference implementation of VIPS achieves 95% text coverage across their dataset; the Java implementation achieves 86% text coverage. The authors suggest that this proves the algorithm to be suitable for text extraction. However, a single segment covering the whole page would achieve 100% text coverage; this shows that the metric does not actually provide any information about segmentation quality. Furthermore, the task of extracting all text from a web page is trivial; the more challenging task of main content extraction can however benefit from a high-quality segmentation, even if it does not cover all of the page's text.

2.2.2 Review of algorithm authors' evaluations

VIPS. Cai et al. have evaluated the performance of VIPS in their technical report [2003b]. While they did not evaluate against a preexisting ground truth, we nevertheless discuss their methods and results to highlight the effect of a lack of ground truth and appropriate evaluation measures. They performed segmentations on 600 pages taken from 14 categories out of a link collection on `yahoo.com`. Five volunteers were asked to judge the result and determine whether the "semantic content structures [were] correctly detected". The

judges classified the segmentation result as "perfect", "satisfactory", "fair", or "bad". The authors claim that for $93.0\overline{3}\%$ of the pages, the segmentation was classified as "perfect" or "satisfactory". They further state that the "fair" cases ($6.1\overline{3}\%$) are caused by "insufficient visual cues" on the page, and that humans "get the structure by understanding the semantic meaning of the blocks". Lastly, they explain that the "bad" cases ($0.8\overline{3}\%$) are due to "wrong [element] position information" supplied by the browser (Internet Explorer) and the inability of the algorithm to detect images as content separators.

The authors do not state the Permitted Degree of Coherence (PDoC) used for this evaluation. Several screenshots of an interface showing the segmentation result appear in the report; they indicate a PDoC value of 5, but it is never explicitly stated whether this value was also used in the evaluation. Furthermore, the DoC values range from 1 to 10 in the implementation used for the report, while the Java implementation used as the base of our port uses values from 1 to 11.

Since the experiment setup and judgment criteria are not clear, the results of this evaluation are difficult to interpret and cannot be directly compared to the results presented here. However, it is immediately apparent that our results do not reflect the very high percentage of "perfect" or "satisfactory" segmentations claimed by the authors. This highlights the importance of proper evaluation methods for application in algorithm development: by being able to obtain more precise and clearly interpretable evaluation results, areas for improvement become more apparent, ultimately benefiting the quality of the approach.

HEPS. Manabe and Tajima have evaluated HEPS' performance in extracting headings and content blocks. Similar to the *Webis Web Segments 2020* dataset, they combine a diverse selection of web page archives with human annotations as their ground truth. However, they do not employ crowd-sourcing, instead relying on a fixed group of seven annotators. Out of their initial 11008-page selection from the ClueWeb 09 Category B dataset, 1219 pages remain as a result of the limited number of annotators and further filtering.

It is important to note that each page was only annotated by one person. The authors therefore provide an inter-annotator agreement measure, which was obtained through a separate annotation process on 102 pages with five annotators for each page. "Candidates" for headings and blocks were extracted, and the annotators classified them as either heading or block. The authors give mean Fleiss' Kappa values of 0.583 for content block annotation and 0.693 for heading annotation. They describe these values as "showing a fair to good agreement or a moderate to substantial agreement" for content block extrac-

tion, and "showing a good and substantial agreement" for heading extraction[2]. The authors' approach to determining inter-annotator agreement differs significantly from the measures employed by Kiesel et al. [2019]. There, clustering similarity measures based on different types of atomic elements are used, which is a more flexible method for assessing segmentation agreement through annotator segmentation similarity than the simple categorization utilized by the authors of HEPS.

In addition to comparing their algorithm to the human annotations, the authors also provide comparisons with two other heading-block-extraction approaches and VIPS. However, in the case of VIPS they only focus on content block detection performance. Furthermore, different PDoC (Permitted Degree of Coherence) values were employed for measuring precision and recall. The authors chose different PDoC values for each page, depending on which PDoC resulted in the highest precision, but used a fixed maximum PDoC for calculating recall. This effectively invalidates the given F-score (0.106, calculated from average precision over all pages and average recall over all pages), because its precision and recall components were obtained through separate evaluations with separate parameters. As VIPS segments the entire page and therefore also returns segments that are not comprised of headings and content blocks, but would nonetheless fit a general definition of a segment as given in Chapter 1, the calculation of precision and recall only for content blocks misrepresents the actual performance of the algorithm for general segmentation tasks.

Because of the explicit focus on heading and content block extraction, it is not possible to generalize these results to facilitate a comparison with our findings.

Edge-based approach (Cormier et al.). Unfortunately, the authors of this approach only evaluated their algorithm for the initial version published in 2016, and not the 2017 version examined in this thesis. Nevertheless, discussing their set-up and results is useful to compare their methodology to the one applied here. Beside a qualitative evaluation of their algorithm, the authors compare their approach to a segmentation in which each element is its own segment (created from element bounding boxes obtained from the DOM tree), a DOM-only version of their algorithm which they call the "control" version, and VIPS. We focus on their comparison with VIPS, since this is most related to our work.

The authors' evaluation is performed on a small dataset of only 50 pages, all being the main pages of the top 50 web sites in Canada as ranked by `alexa.com` on October 9, 2015. This makes their dataset orders of magnitude

[2]The authors give multiple interpretations for the Fleiss' Kappa value based on different guidelines from the literature.

smaller than the one used in this thesis, and considerably less diverse, both due to the size and the page selection. As input for their algorithm, they captured screenshots of these pages using the Firefox web browser. In contrast, they used the reference implementation by Cai et al. to evaluate VIPS, which requires Internet Explorer as the underlying browser; the authors used Internet Explorer 8 on Windows XP. As mentioned previously in Chapter 3 and shown in Section 3.1, the browser can have a serious impact on the rendering of a given web page, harming reproducibility and comparability of results. The authors state that three pages from the dataset were incorrectly rendered in Internet Explorer 8. It is unclear why the authors did not use screenshots from Internet Explorer to provide a maximally consistent environment; tools similar to *Screengrab*, which they used to capture screenshots in Firefox, exist for Internet Explorer[3].

With the changes between the two iterations of the algorithm, the available parameters have changed as well. The only parameter applicable to both versions is s_{min} (called w in the 2016 version), which is set to 15 for the authors' evaluation. In Cormier et al. [2017], the authors state that setting $t_p = 1$ causes the new algorithm to be equivalent to the older method; for lack of an implementation of the previous version, we did not verify this. Lastly, the 2016 version explicitly offers the possibility of stopping the segmentation at a certain tree depth; for their evaluation, the authors chose a maximum tree depth of 6 "to avoid excessively detailed segmentations." The parameters were "determined experimentally on an earlier dataset of 30 diverse web pages."

To measure the similarity between the evaluated algorithms, the authors use *Earth Mover's Distance* (EMD), a similarity measure over probability distributions. They use this measure in an image-based comparison, for which they "create edge maps including all edges across all levels of the segmentation trees" for all algorithms. The authors state that due to the computational intensity of EMD and the size of the full edge maps, they use sampling techniques to perform the comparison. The first sampling technique consists of calculating the EMD for 10 randomly placed 127×127 px patches between the edge maps of two algorithms, which the authors label as "useful for comparison of the small-scale structure of segmentations." As a second technique, they downsample the edge maps by a factor of 50 and compare these directly "for a large-scale comparison of segmentations." After the authors apply normalization to the EMD values to account for "image size or aspect ratio" variations, the result range is [0, 1]; 0 meaning identical images (segmentations), and 1

[3]for example the FireShot tool, which added support for Internet Explorer 8 in version 0.69 (source: `https://getfireshot.com/updated.php?full=1&h=0`), dating to February 22, 2009 (confirmed by checking timestamps for that version obtained from `https://legacycollector.org/firefox-addons/5648/index.html`)

	Upper 3 levels	**Upper 5 levels**
Full page	0.303 ± 0.032	0.308 ± 0.030
127×127 px patches	0.449 ± 0.043	0.633 ± 0.040

Table 2.2: Normalized EMD values (range [0, 1]) taken from Cormier et al. [2016] comparing the upper 3 and upper 5 levels of the resulting segmentation tree from their algorithm against VIPS [Cai et al., 2003a,b]. Smaller values mean larger similarity. 0 means identical segmentation, 1 means entirely different segmentation. The authors performed the comparison both on downsampled full pages as well as 10 randomly selected 127×127 px patches.

entirely different images (segmentations).

The authors compare to VIPS using a PDoC value of 5. They state that this results in "a slightly coarser segmentation than the 6-level segmentation trees" produced by their approach. Therefore, they decided to compare VIPS only to the upper 3 and 5 levels of their algorithm's segmentation tree separately. The results of their comparison are reproduced in Table 2.2.2. The values suggest that Cormier et al.'s algorithm and VIPS perform somewhat similar for large segments (the upper levels of the segmentation tree), but start differing more with increasing tree depth and the resulting more fine-grained segments. Whether this can be attributed to a general difference in granularity between the two algorithms is unclear, since no information about the number of segments produced in this experiment is given.

The EMD measure used by the authors is somewhat more comparable to the measures proposed by Kiesel et al. [2019] and employed in this thesis. We claim that it is most comparable to the F_{B^3} values for *pixels* as atomic cluster elements (see Section 4.2.1), since the methodology for calculating the EMD employed by Cormier et al. effectively compares the presence and extents of segment boundaries between the algorithms with distance measured in pixels.

Table 5.2 shows an F_{B^3} score of 0.51 for the agreement between VIPS with PDoC 5 and the approach by Cormier et al.. Considering the inverse relationship between EMD values and F_{B^3} values (where 0 is most similar for EMD, it is most dissimilar for F_{B^3}, and vice versa), we interpret the results given by the authors as close to our findings, especially for the 127×127 px patches. The results for the full page are not comparable, since the input used in the authors' comparison was heavily downsampled. While the results may be seen as similar to ours, the significant differences in dataset size and variety, the different scale of PDoC values compared to the reference implementation used by Cormier et al., the older version of the authors' algorithm used for

their evaluation, the different metric and the sampling techniques applied in their evaluation ultimately prohibit a direct comparison with our result.

Chapter 3

Approaches

This chapter presents a selection of existing web page segmentation approaches from the literature. First, we give an overview of the categories that segmentation algorithms can be divided into, based on the information they use in the segmentation process. Afterwards, each algorithm is described in a separate section, giving a general overview of its design and details on the implementation used for evaluation on our dataset. The approaches and implementations detailed here were used for the evaluation in Kiesel et al. [2019], as well as for the more in-depth evaluations and comparisons presented in this thesis.

Existing web page segmentation approaches can be categorized as follows:

DOM-only. The segmentation process uses only information stored in the DOM tree in the rendering engine. This includes the types of elements, their content, their relations to each other (parents, children, siblings), their location and size, and style information such as font properties or background color. Since building the DOM tree requires rendering the page, the resulting tree and, by extension, the result of the segmentation may be influenced by a number of factors, such as the rendering engine or browser that was used, or the size of the viewport. While the latter can be kept consistent while performing the segmentation multiple times, the pace of modern web browser development may make it difficult to achieve consistency over a longer period of time. For this reason, we employ archived web pages and a consistent reproduction environment to maximize reproducibility.

Visual. The only input is a partial or complete screenshot of the web page. While the appearance of the screenshot also depends on the rendering engine to some extent, the image can be created from a live web page. This means that, in contrast to DOM-only approaches, no further effort is required to ensure a consistent input.

Approach	Document	Features	Basic principle	Output
VIPS [Cai et al., 2003a,b]	Web page	Tree, style, location	heuristic	Rectangle tree
HEPS [Manabe and Tajima, 2015]	Web page	Tree, style, location	heuristic	Node set
Cormier et al. [2017]	Web page	Screenshot	probabilistic	Rectangle tree
Meier et al. [2017]	Article page	Screenshot, text mask	Machine learning	Mask
MMDetection [Chen et al., 2019a]	Photo	Screenshot	Machine learning	Mask

Table 3.1: Overview of the five compared segmentation algorithms: the kind of documents they were created for, the features they use, the basic principle of their operation, and the format of the output segmentation.

Hybrid. Multiple forms of input are combined. For example, the approach by Meier et al. [2017] uses a screenshot as well as a mask over the textual content of the document. This requires further preprocessing; in our application, the text mask is generated from the DOM, since this is more accurate compared to OCR employed for this purpose for other types of documents. Such methods may therefore also be susceptible to consistency and reproducibility issues.

A tabular comparison of the chosen algorithms is given in Table 3.1.

We wanted to present at least one approach from each of these categories in this thesis. In addition to judging the performance of each of the selected algorithms, we can therefore also make performance claims in terms of the categories the algorithms belong to, thereby contrasting these main branches of web page segmentation approaches. Another important aspect of our algorithm selection was the availability of a suitable implementation. For example, implementations of DOM-based web page segmentation algorithms that can be run on any underlying rendering engine are the exception rather than the norm. To encourage the development of such universal implementations, we contribute our TypeScript/JavaScript port of VIPS created for this thesis and the evaluations performed in Kiesel et al. [2019].

3.1 VIPS

The VIPS (*VIsion-based Page Segmentation*) algorithm presented by Cai et al. [2003a,b] belongs to the category of DOM-only approaches. It is the most well-known web page segmentation algorithm to date; many other works in web page segmentation reference it or evaluate their methods against it, such as those discussed in Section 2.2.

It operates on a fixed set of rules to segment a web page (see Cai et al. [2003b, section 4.1, table 1]). These rules take into account the structure of the DOM tree (e.g. the number of children of a node), the sizes and types of

nodes in the tree, and specific style features of an element/a node, namely font size, font weight and background color.

Algorithm. The algorithm defines a "vision-based content structure" for a web page, which is made up of *blocks*, *separators* and a block adjacency relation δ.

A block "is a basic object or a set of basic objects"; a basic object is a "leaf node in the DOM tree that can not be decomposed any more". Blocks are extracted using the aforementioned rules, deciding whether to assign a DOM node and its children as a block, or to move further down into the DOM tree and form blocks out of the child nodes (and possibly their children).

Separators are rectangular areas that exist between blocks that are classified as adjacent by δ. Their detection process is split into horizontal and vertical separators. For a given pool of blocks, at first one separator is created spanning the entire width or height of the pool; afterwards, separators are split, removed, or have their sizes changed based on whether a block is contained in the separator, crosses it, or covers it.

Every block is assigned a value called "*Degree of Coherence*" through application of the rules. This value has the following properties as described by the authors:

- "The greater the DoC value, the more consistent the content within the block."
- Within the vision-based content structure, the DoC of a child block within its parent block "is not smaller than that of its parent."

The algorithm works recursively. It begins with one block spanning the entire page, and applies the rules on it, deciding whether to divide further or not. After the division in each recursion step, the separator detection is performed as outlined above. The size of the detected separators is another influence on the DoC of the block containing them; if separators are small, the blocks within the parent block are seen as more coherent, and the parent block's DoC increases, while large separators indicate lower coherence and cause a lower DoC for the parent block. The recursion stops if the "*Permitted Degree of Coherence*" (PDoC), a predefined value influencing the segmentation granularity, has been reached for this block, or if no further division can be performed according to the rules. The detected separators between blocks with DoC <= PDoC form the final segment borders.

Implementation. Because VIPS uses information from the DOM tree of a web page for its segmentation, reproducible results necessitate its execution

within the Webis Web Archive Environment. To the best of our knowledge, no implementation that allows this, i.e. a browser-independent version, exists currently.

The authors provide[1] a DLL library, but it is meant for use on Microsoft Windows with Internet Explorer. Furthermore, it was last updated in March 2008. No source code is supplied with this implementation.

There is also a Java implementation of VIPS by Popela [2012]. Its source code is obtainable from GitHub[2]. This version served as the base for our port to the TypeScript programming language, a typed extension of JavaScript. Our implementation is available at `https://github.com/elmeyer/VipsTS`.

In the Java implementation, *CSSBox*[3] is used as the backing rendering engine. Compared to the Chromium browser used in the Webis Web Archive Environment, its rendering performance is poor, causing large differences between the resulting appearances of a page. An example of this is shown in Figure 3.1. This makes it impossible to compare the segmentation results to those obtained from the Chromium browser, or screenshots produced by it.

In our port, calls to CSSBox were changed to JavaScript calls to the browser's DOM, such as `element.getBoundingClientRect()` to obtain an element's bounding box coordinates. Apart from this, the algorithm's behavior remains unchanged in the port. The execution flow and naming conventions from the Java implementation were followed precisely. The XML output of the original implementation was discarded in favor of additional code producing JSON output compatible with the evaluation tools by Kiesel et al. [2019].

3.2 HEPS

HEPS (*HEading-based Page Segmentation*) is a DOM-only web page segmentation algorithm introduced by Manabe and Tajima [2015] as "a method that extracts logical hierarchical structure of HTML documents by exploiting hierarchical headings in them." The focus lies specifically on modelling how human readers recognize the structure of a web page through the headings present on the page. The authors state that their "main applications are information extraction and information retrieval" [Manabe and Tajima, 2015, section 3.1].

Algorithm. HEPS specifically regards headings and blocks separately, and the authors define them as follows:

[1] `http://www.cad.zju.edu.cn/home/dengcai/VIPS/VIPS.html`
[2] `https://github.com/tpopela/vips_java`
[3] `http://cssbox.sourceforge.net/`

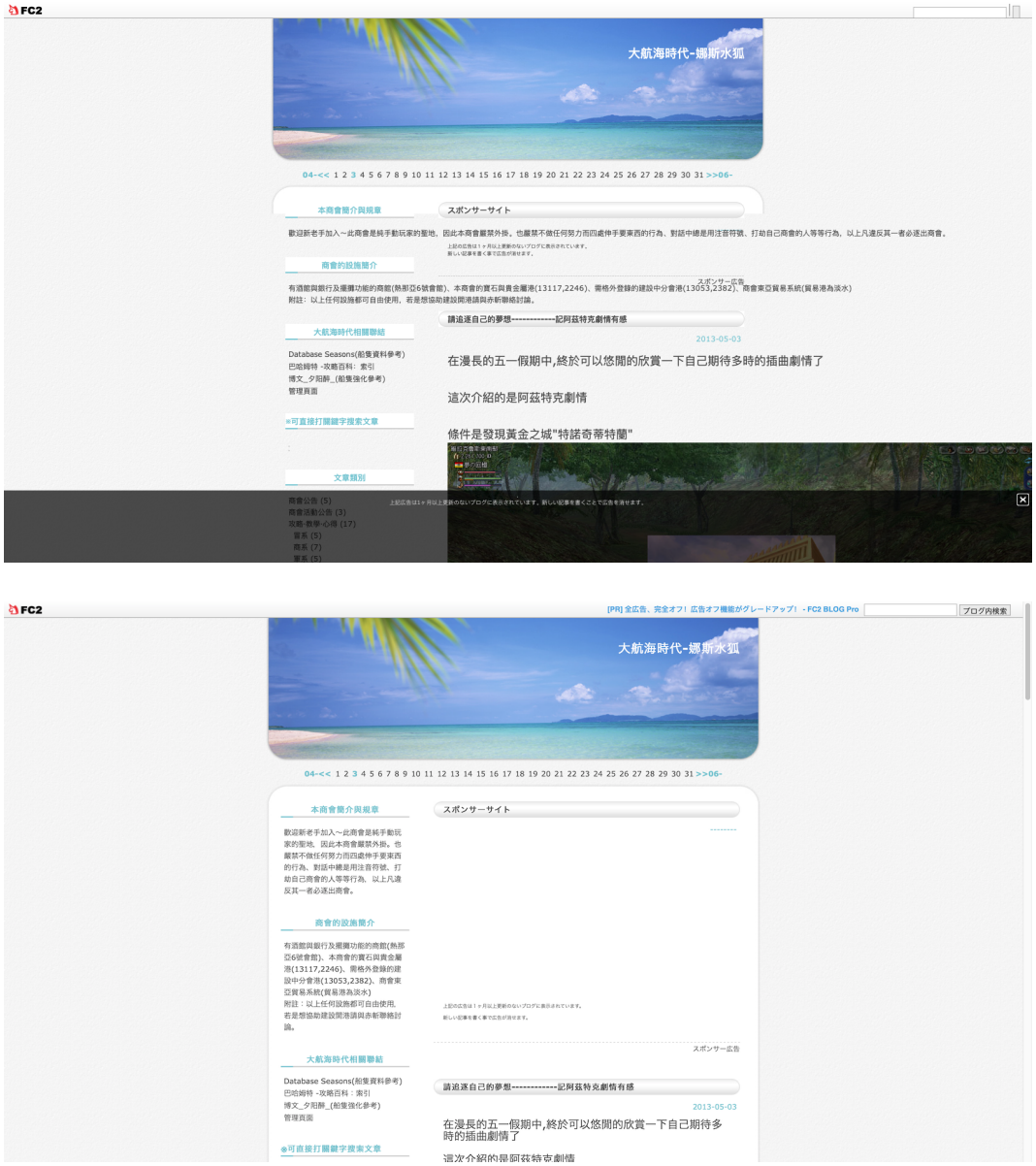


Figure 3.1: Example page showing rendering errors produced by CSSBox (top), the underlying rendering engine of the Java implementation of VIPS by Popela [2012]. A correct rendering (created with Safari 13.0.4 on macOS) is shown below.

Heading: "A heading is a visually prominent segment of a document describing the topic of another segment."

Block: "A block is a coherent segment of a document that has its own heading describing its topic."

The algorithm does not rely on the `<h{1..6}>` or `<dt>` HTML tags for headings, instead basing heading detection on CSS style properties. This is because the authors claim that less than $\frac{1}{3}$ of the pages in their dataset have headings marked up using the proper HTML tags, and that only about $\frac{2}{3}$ of occurrences of the aforementioned tags are really headings. The style properties examined by the algorithm are `font-size`, `font-style`, `font-weight`, `text-decoration`, and `color`. HEPS also considers images as heading candidates; the authors claim that candidate heading images usually contain text, and that "the height of such an image reflects the font-size of the characters."

The segmentation process is based on four assumptions about headings made by the authors. They state that (1) "Each heading appears at the beginning of its corresponding block", (2) "Headings and non-heading segments never have the same visual style", (3) "Two headings [...] have the same visual style if and only if they are at the same significance level", and (4) "Headings at higher levels are given more prominent visual styles than headings at lower levels."

The segmentation is performed as follows: After a preprocessing step, in which blank text nodes and sentence-breaking nodes[4] are removed, *candidate heading lists* are extracted by classifying the visual style of text nodes and their ancestors, and considering images and their `alt` texts[5]. The lists are then sorted based on the tree depth of the nodes associated with the candidate headings, their visual style and the order in which they appear in the document. As a result, the candidate heading lists are ordered by their *significance level*.

The blocks that correspond to headings are constructed from the DOM sub-trees that the heading belongs to. Blocks are defined as "node sequence[s] consisting of adjoining sibling nodes (or consisting of a single node) inclusive of their descendants." The authors reason that "it is very rare that a block only partially overlaps with a subtree in the DOM tree", justifying their block extraction methodology. The relationship between heading and block in the tree is therefore such that "the heading of a block is represented either by the first node of the corresponding node sequence or by its descendant."

[4] Defined by the authors as "nodes that divide a sentence into multiple text nodes", such as links in a sentence, which are usually presented in a different style.

[5] "Replacement text for use when images are not available" [O'Hara et al., 2018, section 4.7.5]

Finally, the web page is recursively segmented using the ordered candidate heading lists. This produces a hierarchical segmentation according to the significance level of each candidate heading list.

Implementation. We used the original JavaScript implementation as provided by the authors of the algorithm[6], but modified it as described below.

The paper introducing the algorithm is mainly focused on its use for content extraction. As such, the original implementation does not expose segment boundaries, but outputs extracted headings and content. To rectify this for our purposes, we obtained the segment boundaries from the bounding boxes of the DOM nodes stored internally in the implementation. Specifically, we combine a detected heading and its corresponding block to form one segment.

Segmentation initially failed for 211 web pages, or ~2.5% of the dataset. Debugging identified the cause for all failures to be external JavaScript libraries used on the affected pages with incompatible implementations of `Array.prototype.every` and `Array.prototype.map`. This was fixed by forcing the use of ECMA-262 version 5-compliant implementations for these functions through polyfills[7].

As with the VIPS port, the only other addition to the code was the production of JSON output in the necessary format.

3.3 Edge-based approach (Cormier et al.)

Cormier et al. introduced their purely visual web page segmentation approach based on edge detection in 2016, refining it in a further publication in 2017. This overview is based on the descriptions given in Cormier et al. [2017], since we use an implementation of this version of the algorithm, where major parts have been changed compared to the initial publication from 2016.

Being purely visual, the algorithm requires only a screenshot of the web page as input. It uses "a Bayesian framework to detect semantically significant edges and lines in the page image", which delineate the page segments. The authors' primary motivation is "to enable improved online experiences for users with assistive needs."

Algorithm. The algorithm goes through three stages in the segmentation process, as summarized by the authors:

1. "Calculation of the probability of a locally significant edge at each pixel"

[6] `https://github.com/tmanabe/HEPS`

[7] Obtained from `https://developer.mozilla.org/`

2. "Calculation of the probability of a semantically significant line segment between two points"

3. "Calculation of the most probable segmentation under specific structural assumptions"

Edge detection is at the core of this segmentation method. Since most web pages contain high-contrast text, commonly used edge detection methods such as the one presented by Canny [1986] tend to identify edges along the characters, but neglect "faint lines or slight changes in background color" [Cormier et al., 2017], which are actually important for identifying the higher-level divisions on the page. Therefore, the edge detection approach employed here is tuned for "axis-parallel edges and strong responses to faint but locally important edges." For each pixel of the input screenshot, "preliminary edge strengths" are estimated for horizontal and vertical edges using the Sobel-Feldman operator [Sobel and Feldman, 1968]. The operator is applied once to the entire image, but its result is interpreted locally. For this, a neighborhood is defined on either side of the proposed edge for the pixel, separately for horizontal and vertical proposed edges. The gradients in such a neighborhood, resulting from applying the Sobel-Feldman operator, are interpreted in terms of a probability distribution. Based on this, the probability of a "locally significant edge" at a given pixel is defined to be the probability that its preliminary edge strength is *not* "drawn from the same distribution as the gradients in the neighborhood"; that is, the gradient value of the current pixel is "significantly stronger" than that of the pixels in the neighborhood. The neighborhood includes the current pixel, but not pixels collinear with the current pixel's proposed edge (horizontal or vertical), since the authors found that this skews the probability in favor of edges that do not lay on "extended lines", meaning edges that are "isolated" in the image.

In the next step, "these local edges must be 'assembled' to produce the outlines of regions." The algorithm constructs line candidates from the previously determined locally significant edges, and determines whether they are "*semantically* significant." The probability that a line candidate is semantically significant is defined recursively. The authors introduce a "maximum segment length threshold" t_l; when this length is exceeded by a candidate line, the line is split and the product of its halves' probabilities is the probability for the original line. For line candidates shorter than t_l, e.g. after splitting a longer line candidate, the probability is determined by whether "sufficiently many pixels [...] along the line have locally significant edges", "sufficiently many" being defined by a threshold probability t_p. The authors use Monte Carlo simulations on all pixels of a candidate line to estimate this probability. They state that the Monte Carlo method was chosen over exact calculations

of this probability for efficiency reasons.

Finally, the web page is segmented according to the previously computed line probabilities. Similar to HEPS (see Section 3.2), the authors make a number of assumptions that the final segmentation is based on:

1. "The image axes [...] are guaranteed to correspond to the horizontal and vertical axes in the rendering process."
2. "Regions in a web page are typically represented well by axis-parallel rectangles."
3. "Each region is fully contained within its parent region"
4. "Each region is fully covered by its child regions"
5. "No regions overlap unless one region contains the other"
6. "The root of the segmentation tree is a region consisting of the entire page"

Assumptions 3–6, along with the restriction "that all subregion divisions at a given level of the tree be either horizontal or vertical", mean that the output segmentation is an X-Y tree. The segmentation is found recursively, with the entire page as the initial segment. For each segment, the horizontal or vertical line with the highest probability is used to divide the segment further, unless no line with a probability of being semantically significant > 0.5 exists within the segment. The division is further constrained by a parameter s_{min}, which the authors describe as the "minimum size" of a segment; in practice (in the authors' implementation), it dictates the minimum length of a segment border. Therefore, the line probability and size criteria are used to terminate the segmentation process.

Implementation. To perform our evaluations, we used an implementation of the algorithm kindly provided by the authors.

In the supplied source code, the authors make suggestions for the aforementioned parameters t_l and s_{min}. They provide values of $t_l = 256$, $t_l = 512$ and $s_{min} = 15$, $s_{min} = 30$. However, the implementation internally multiplies s_{min} by a factor of 3.[8] Therefore, the suggested s_{min} are 45 and 90, in practice. We perform an analysis of these parameters with the suggested values in Section 5.4. t_p was given as 0.5 and remained unchanged across all evaluations.

[8] In correspondence with the author, this was revealed to be left in the code unintentionally; as we had already performed the computationally intensive evaluations, we chose to keep the obtained results and adjust our references to the parameters accordingly.

In the form of the provided implementation, the algorithm can be characterized as computationally intensive. For reference, segmenting a 1366×5590 px web page takes approximately 45 minutes on an Intel Core i7-7700 at a clock speed of 4 GHz. The used implementation is single-threaded; it is likely that the more intensive first two stages of the algorithm (finding locally significant edges and semantically significant lines) could see major speed improvements from parallelization. Due to the large amount of time it takes to evaluate the algorithm on the entire dataset of 8490 pages, no analysis of the t_p parameter was performed.

3.4 Neural network (Meier et al.)

The neural network approach presented in Meier et al. [2017] belongs to the category of hybrid approaches. It was originally intended for segmenting newspaper articles. As the main challenge of this task, the authors identify "the diversity of the medium", stating that "different 'genres' have vastly different layouts", and that layout is also influenced by other factors such as "time and culture." However, they acknowledge that web page segmentation is "similarly challenging." Their motivation lies in the searchability of print media, for which they state that "optical character recognition (OCR) based digitization is insufficient."

Algorithm. The authors present "a fully convolutional neural network (FCN) that transforms the input page into a complete segmentation mask in one pass." Their approach is based on the FCN architecture presented in Long et al. [2015]. Meier et al. add refinements to this approach that exploit the properties of newspaper segmentations, namely the exclusively rectangular shape of segments (a property that also applies to web pages).

Their network takes as input an image of the newspaper page and a mask over the textual content of the page, obtained through OCR, in the form of black rectangles covering the lines of text. The output of the network is also a mask showing black rectangles that delineate segments. Similarly, the ground truth used for training is such a segmentation mask created by human annotators.

The optimizations applied by the authors serve to decrease training and classification time. Besides tuning for rectangular segments, the authors have also determined that input sizes larger than 256×256 px "do not increase the segmentation quality of newspaper articles." Within the network, the resolution decreases even further; the authors state that "the higher resolution is not required for segmentation, because the newspaper articles have a very

coarse, often rectangular outline", and present the fact that "fine details can be processed in the first layers, but subsequently lower resolutions of a page may be used internally" as an advantage of the employed FCN architecture. However, the network is described as suitable for "any input image size in the range of $64n \times 64m$."

Meier et al. have developed this approach with an industry partner for the purpose of "media monitoring." Their partner previously used another convolutional neural network (CNN) for segmenting newspaper pages. The authors claim that their new approach improves on this both in segmentation quality and speed, with an improvement of 46.3% in the *diarization error rate* metric they use, and a classification speed improvement "by a factor of 34.8 to only a few milliseconds" per page.

Implementation and training. We use the implementation provided by Kneist as part of his Master's thesis [Kneist, 2019]. It is an implementation of the model as described in Meier et al. [2017], with a correction reported by the authors (layers `conv6-1` and `conv7-1` have a kernel size of 3×3 instead of 5×5). The model is implemented using Keras[9].

In the following, we highlight the differences between the training process employed by the authors and our training. The primary difference is the dataset. Since the authors proposed this algorithm for newspaper segmentation, they used a dataset "of approximately 5,500 newspaper pages of the most influential Swiss newspapers from the year 2016." Of these 5,500 pages, only 426 were fully labeled (that is, they were completely segmented by the annotators). Furthermore, some of the newspaper pages contained non-rectangular labels, so the authors removed those pages for a final dataset size of 4,135 pages. To train, they split the input into a training set comprising 80% of the entire dataset, which also contains 80% of the 426 fully labeled pages, and a test set comprising the remaining 20% of pages.

In contrast, our dataset of 8490 pages is completely labeled and contains only rectangular segmentations, so it can be used in its entirety. We employ 10-fold cross-validation, such that every fold contains 80% of pages (6792 pages) for training, 10% of pages for validation and 10% of pages as a test set (849 pages each) to produce the evaluation results. One key difference is how we obtain the input text box masks: while the authors use OCR to identify text and mask it, we use text node information from the DOM tree. The advantage of the authors' application of OCR is that its output also "contains any horizontal or vertical line that appears in the newspaper page"; this is not the case for our DOM tree approach, which would face similar issues in identifying

[9] `https://keras.io`

lines on a page that Cormier et al. attempt to tackle with their method.

Common to both the authors' and our training methodology is the resizing of images as a preprocessing step. The authors scale all pages to a height of 256 px while preserving aspect ratio, and then place them "at a random x position within the 256×256 target frame." In contrast, our dataset contains pages of constant width and varying height; therefore, we decided to crop or pad all pages to a height of 4096 px, and we then downscaled them to a final size of 256×768 px.

The authors employ further preprocessing techniques; some are specific to partially labeled pages and thus not necessary for our dataset. For all pages, they shrink segments by 2 px to ensure a minimum segment border size of 5 px. Furthermore, "the newspaper pages are also randomly mirrored along the y axis with a probability of $p = 0.5$." We did not apply either of these techniques.

For the results presented in their paper, the authors trained the network for 210 epochs on the full training set, and then for another 150 epochs on only the 426 fully labeled pages. They state that "the network starts overfitting" on their data afterwards. Training each of our 10 folds for 150, 210 or even the combined 360 epochs was not possible due to time and resource constraints. Therefore, we decided to employ Keras' `EarlyStopping` callback that is meant to detect when overfitting occurs. We used a `patience` value of 10, meaning that if validation loss did not improve for 10 consecutive epochs, training was stopped. The resulting average training duration was 44.6 epochs.

To obtain the final segmentation, we perform a binarization of the network output, and use *connected-component labeling* as implemented in *ImageMagick*[10] to determine the bounding boxes of the segments. We perform this operation on the unscaled network output of size 256×768 px, since we found that prior upscaling leads to a high amount of artifacts (small holes in the segments). As parameters for the ImageMagick command, we adopt those given in the implementation by Kneist [2019] but lower the binarization threshold to 45%; the full command is as follows:

```
convert input.png -threshold 35% \
-define connected-components:verbose=true \
-define connected-components:area-threshold=50 \
-connected-components 1 output.png
```

We briefly explain the parameters below:

- `-threshold 35%` binarizes the grayscale image such that values below 35% of the maximum intensity turn black and values above this threshold turn white. The threshold greatly influences the resulting segmentation

[10] `https://imagemagick.org/script/connected-components.php`

in terms of segment size and number of segments; a combination of visual inspection on a sample of the pages as well as calculating the average number of segments per page was used to determine a suitable value.

- `-define connected-components:area-threshold=50` reduces artifacts caused by small areas in the output that are not likely to form a segment.
- `-connected-components 1` connects areas if any neighboring pixel has the same value; due to the previously applied binarization, a larger neighbor radius than 1 is unnecessary and may lead to incorrectly connected segments.

While an output image is produced, we do not use it except for visual inspection of the result; `-define connected-components:verbose=true` outputs bounding box coordinates which are parsed and transformed into our JSON format for evaluation.

3.5 MMDetection

MMDetection is "an object detection toolbox that contains a rich set of object detection and instance segmentation methods" [Chen et al., 2019a]; it falls into the category of hybrid approaches. It is a modular set of tools developed on top of *PyTorch*[11] designed to implement a large number of machine learning frameworks for detection tasks on real-world images. The authors claim it to be highly efficient, focusing on executing most operations on GPU(s) and enabling training speed "faster than or comparable to other codebases."

The project emerged from *MMDet*, a previous version that "won the detection track of COCO Challenge 2018." This is the primary reason for our selection of this algorithm as part of this thesis; at the time of writing, MMDetection leads the "SEGM" (instance segmentation) leaderboards of the COCO challenge [*Common Objects in Context*, see Lin et al., 2014].

Section 2 in Chen et al. [2019a] lists the 30 different frameworks that are implemented. All implemented frameworks share a common training pipeline, which the authors claim to be usable for "semantic segmentation" as well. This pipeline is designed to be highly customizable using a "hooking mechanism" that allows arbitrary operations to be inserted at any point of the training process. Because of the provided high flexibility, it would be possible to implement the approach by Meier et al. [2017] in this toolbox as well. Whether this would provide benefits in training or classification performance may be suitable as a topic for future work.

[11] `https://pytorch.org/`

Implementation. Following a recommendation for highest performance given in the project documentation on GitHub[12], we use a pre-trained model supplied in the projects "model zoo" that was implemented on the Hybrid Task Cascade framework presented in Chen et al. [2019b]. The model's architecture is based on a X-101-64x4d-FPN (*Feature Pyramid Network*) backbone and contains c3-c5 DCN (*Dynamic Capacity Network*) layers. It performs both object detection and instance segmentation. Because the model was trained on real-world images, it would only detect real-world objects, present for example in images on the web page, when classifying using the default parameters. Therefore, we had to disable the default detection threshold to allow the neural network to return all results. We then transformed both the object detection bounding boxes and the instance segmentation masks returned as output into the JSON representation suitable for our evaluation tools. However, we are mainly interested in the instance segmentation performance, and perform our evaluations using only this output.

[12] `https://github.com/open-mmlab/mmdetection/blob/master/configs/htc`

Chapter 4

Evaluation methodology

This chapter presents the methodology we use to evaluate the algorithms discussed in this thesis. After a summary of our criticism of the evaluation methodologies applied in the literature thus far, we describe the similarity measure used to assess algorithm performance in our work, as first presented in Kiesel et al. [2019]. We also detail our efforts in determining whether a best choice of atomic elements for this clustering similarity measure exists. Finally, we give an overview of how the dataset that serves as the ground truth for our comparisons was created.

4.1 Criticism of prior evaluation methodologies

In Section 2.2, we gave an overview of previous attempts at evaluating web page segmentation algorithms from the literature. There, we made a number of observations that led to criticisms of the methodologies that were previously employed. In the following, we summarize our observations and points of criticism that pave the way for the methodology applied in this thesis.

- To obtain meaningful results, algorithms must be compared to a preexisting ground truth using well-founded similarity measures; judging the quality of a segmentation purely visually without any reference suffers from imprecise judgment criteria and leads to incomprehensible results (cf. Cai et al. [2003a,b]).
- Tailoring the evaluation criteria to a specific use case may misrepresent the performance of an algorithm for general segmentation tasks (cf. Manabe and Tajima [2015]).
- The tool used for ground truth creation must influence the annotator as little as possible; for example, segmenting based strictly on DOM nodes

may suggest relationships between elements that the annotator may intuitively understand differently (cf. Kreuzer et al. [2015], Sanoja and Gançarski [2015]). For the same reason, instructions for the annotation process must be formulated carefully.

- A lack of precision in the employed similarity measure causes a loss of information that may be crucial when it comes to selecting an algorithm for a given application (cf. the text matching metric used in Kreuzer et al. [2015]).
- The chosen evaluation metric must actually correspond to segmentation quality; trivial segmentations must be appropriately punished (cf. "text coverage" used in Sanoja and Gançarski [2015]).
- Most importantly, evaluation results can only be fully interpreted, compared and verified if all parameters used for the evaluation are stated. For example, aiming for "medium sized" segments as in Sanoja and Gançarski [2015] is vague, and parameters to achieve this may be different from page to page.

4.2 Segmentation similarity and quality

In this section, we present the evaluation methodology proposed in Kiesel et al. [2019], which we apply in this thesis. We further investigate the agreement between the different choices of atomic elements for the clustering similarity measure on the basis of the fused ground truth annotation similarities.

4.2.1 Measuring segmentation similarity

Kiesel et al. identify "a strong connection between web page segmentation and clustering", and regard a segment as a cluster of atomic elements. Since we use the same measures and tools for assessing segmentation similarity (and, when comparing to a ground truth, segmentation quality), we reproduce the description of the similarity measures given in [Kiesel et al., 2019, Section 4] below.

> "Given a web page p, let $E = \{e_1, \ldots, e_n\}$ be the set of its atomic elements. Then $S = \{s_1, \ldots, s_m\}$ denotes a (partial and/or overlapping) segmentation of p into segments $s_i \subseteq E$, with $\bigcup_{i=1}^{m} s_i \subseteq E$. Given two segmentations S and S^* of the same page p, we identify the extended BCubed F_1-score by Amigó et al. [2009] (an extrinsic

cluster evaluation measure) as particularly suited to measure segmentation similarity in general, and segmentation quality in particular. The latter depends on S^* being a ground truth segmentation.

Just like the well-known F_1-score, the extended BCubed F_1-score (F_{B^3}) is the harmonic mean of the extended BCubed precision (P_{B^3}) and recall (R_{B^3}):

$$P_{B^3}(S, S^*) = \frac{1}{|E^S|} \sum_{e \in E^S} \left(\frac{1}{|E_e^S|} \sum_{e' \in E_e^S} \left(\frac{\min\left(|S_e \cap S_{e'}|, |S_e^* \cap S_{e'}^*|\right)}{|S_e \cap S_{e'}|} \right) \right);$$

$$R_{B^3}(S, S^*) = P_{B^3}(S^*, S); \quad F_{B^3}(S, S^*) = 2 \cdot \frac{P_{B^3}(S, S^*) \cdot R_{B^3}(S, S^*)}{P_{B^3}(S, S^*) + R_{B^3}(S, S^*)};$$

where $S_e \subseteq S$ is the subset of segments that contain element e, $E^S \subseteq E$ is the subset of elements that are part of at least one segment in S, and $E_e^S \subset E$ is the subset of elements that accompany element e in at least one segment in S. More formally, $S_e = \{s \mid s \in S \wedge e \in s\}$, $E^S = \{e \mid S_e \neq \emptyset\}$, and $E_e^S = \{e' \mid S_e \cap S_{e'} \neq \emptyset\}$."

Five different atomic elements are defined: *pixels* with subtypes $edges_{\text{fine}}$ and $edges_{\text{coarse}}$, *nodes*, and *chars*. *pixels* simply regards every pixel on the web page screenshot as an atomic element. $edges_{\text{fine}}$ and $edges_{\text{coarse}}$ regard pixels after applying the edge detection algorithm by Canny [1986], differing in their parameter choices. The parameters for $edges_{\text{fine}}$ were chosen such that "edges include the outline of characters at 10pt font size"; the parameters for $edges_{\text{coarse}}$ were chosen such that edges include "only lines of text." *nodes* refers to nodes in the DOM tree, but incorporating only visible element and text nodes. *chars* refers to "all characters on a web page."

The properties of the employed BCubed cluster evaluation measures benefit the interpretability of the results and do not limit us to comparisons against a ground truth. The F_{B^3} value is symmetrical and expresses the similarity between two general segmentations, both of which can be algorithm segmentations. Consequently, we perform a comparison of all algorithm segmentations among each other in Section 5.2. Furthermore, since "BCubed precision ignores errors of strict oversegmentation" and "BCubed recall ignores errors of strict undersegmentation", this facilitates identifying "how much of the error stems from strict over- and undersegmentation" "by comparison with F_{B^3}."

When comparing these measures with the ones used in previous web page segmentation algorithm evaluations, as discussed in Section 2.2 and critically

summed up in Section 4.1, it becomes apparent that this newly proposed methodology provides strong advantages in the precision and interpretability of its results. The choice of granular and well-defined atomic elements provides a high level of accuracy while ensuring the generality of the obtained results, making it possible to interpret them for a variety of use cases, instead of being limited to a specific scenario. Furthermore, unlike for example the "text coverage" metric used in Sanoja and Gançarski [2015], the results obtained from these clustering similarity measures give a more complete picture of the nature of a segmentation, and trivial segmentations are punished by either P_{B^3} or R_{B^3}, affecting F_{B^3} appropriately.

4.2.2 Ranking atomic element similarity

To determine whether a best choice of atomic elements for the clustering similarity measure exists, we performed a limited experiment with two annotators on a sample of 59 pages from the dataset. Figure 4.1 shows the web interface that was designed for this purpose. The sample of pages was selected as follows:

Given is a pair of clustering similarity measures for different atomic elements ${s_{\text{elements}}}^1, {s_{\text{elements}}}^2$. We aim to find segmentations $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ that fulfill the following criteria:

$$ {s_{\text{elements}}}^1(\mathbf{A}, \mathbf{B}) > {s_{\text{elements}}}^1(\mathbf{A}, \mathbf{C}) \wedge {s_{\text{elements}}}^2(\mathbf{A}, \mathbf{B}) < {s_{\text{elements}}}^2(\mathbf{A}, \mathbf{C}) $$

The magnitude of difference d between the clustering similarity values ${s_{\text{elements}}}^1, {s_{\text{elements}}}^2$ for segmentations $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ can be calculated as follows:

$$ d = ({s_{\text{elements}}}^1(\mathbf{A}, \mathbf{B}) - {s_{\text{elements}}}^1(\mathbf{A}, \mathbf{C})) \cdot ({s_{\text{elements}}}^2(\mathbf{A}, \mathbf{C}) - {s_{\text{elements}}}^2(\mathbf{A}, \mathbf{B})) $$

The above criteria are fulfilled if $d > 0$.

The five different atomic elements result in a total of 10 unique pairs ${s_{\text{elements}}}^1, {s_{\text{elements}}}^2$. d was calculated for each pair ${s_{\text{elements}}}^1, {s_{\text{elements}}}^2$ for every triplet $\mathbf{A}, \mathbf{B}, \mathbf{C}$ for all pages in the dataset. The results were then ranked by d in descending order, and the top 10 triplets $\mathbf{A}, \mathbf{B}, \mathbf{C}$ and the associated pages were selected for each pair ${s_{\text{elements}}}^1, {s_{\text{elements}}}^2$.

This selection therefore initially yielded 100 pages. However, in many cases the same pages and associated segmentation triplets appeared in the top 10 of multiple atomic element pairs. This is expected, since for example $edges_{\text{coarse}}$ and $edges_{\text{fine}}$ produce similar values when used for the similarity measure, and therefore show similar d values when compared to other choices of atomic elements. It was therefore not necessary to annotate such cases multiple times, and they were removed from the sample. In total, there were 41 such cases; the final sample size was thus 59 pages.

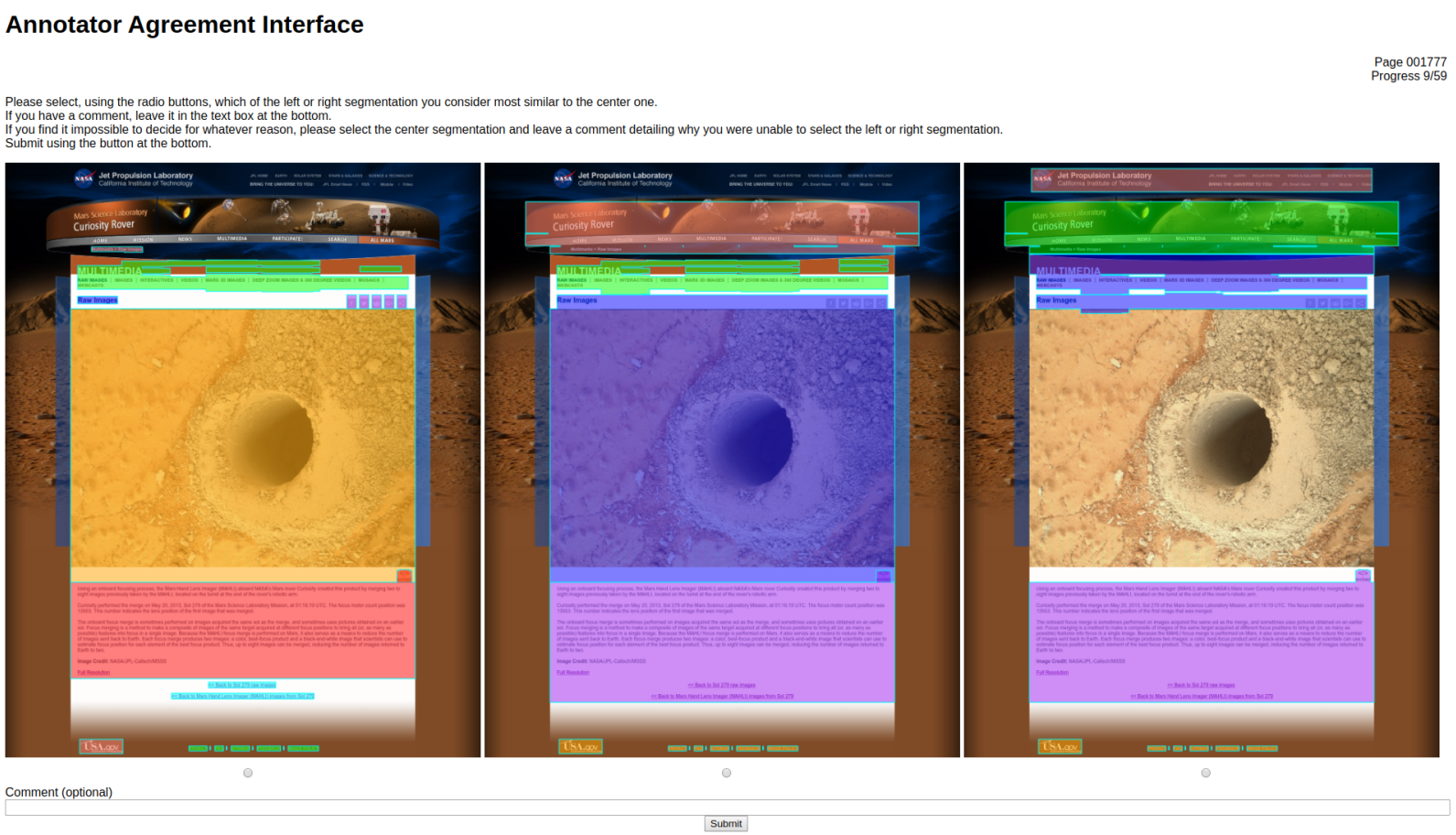


Figure 4.1: Screenshot of the interface used for experimentally determining clustering measure agreement. We refer to the **left** image as **B**, to the **center** image as **A**, and to the **right** image as **C**.

Atomic elements	Wins	Losses	Win ratio
pixels	16	12	0.57
$edges_{\text{fine}}$	15	13	0.54
$edges_{\text{coarse}}$	16	16	0.50
nodes	14	16	0.47
chars	14	18	0.44

Table 4.1: Results of the annotations for the experiment described in Section 4.2.2. Shown are only those cases for which both annotators agreed.

In the interface, the annotators were instructed to judge which of the segmentations **B** or **C** (left and right, respectively) they consider most similar to the center segmentation (**A**), thereby deciding which choice of atomic elements expresses the similarity to the center segmentation more strongly.

In many cases, the differences between the segmentations were small and hard to spot; still, both annotators managed to agree in approximately 75% of cases. We deem this level of agreement to be suitable to draw further conclusions from the obtained results. We do not incorporate the ~25% of cases where the annotators disagreed.

The results of this experiment see no choice of atomic elements for the clustering similarity measure emerging as a clear winner. Because of the high task difficulty for the annotators, we doubt that scaling up this experiment to more participants would provide clearer results.

As the results show no best choice of atomic elements for the clustering similarity measure, and the given choices for atomic elements are appropriate to measure the performance of web page segmentation algorithms in certain use cases (such as *chars* for text extraction), we find that evaluating the algorithms in all of these measures gives a more complete picture of the quality of the produced segmentations.

4.3 Dataset and ground truth creation

In this section, we describe the methodology with which the underlying dataset for this thesis was created. This shall serve to contrast the nature of our dataset to those used in prior evaluations, reviewed in Section 2.2. The *Webis Web Segments 2020* dataset including all annotations and resources is publicly available at `https://doi.org/10.5281/zenodo.3354902`. We highlight its large size and diversity, as well as the novel method for obtaining a ground truth by fusing the crowd-sourced segmentations. We apply the method for

segmentation fusion and the post-processing step of fitting segmentations to DOM nodes to the algorithms discussed in this thesis in our later experiments.

The dataset and ground truths used in this thesis were assembled on the basis of the Webis Web Archive 17 [Kiesel et al., 2018]. This process is explained in detail in [Kiesel et al., 2019, Section 5]; we provide a brief overview here.

The Webis Web Archive 17 contains 10000 pages obtained from sites with high and low rankings on Alexa[1], ensuring high diversity. The archiving process is tuned for archival quality and completeness, and thus provides "all resources required as input for the various existing web page segmentation algorithms." Preprocessing was performed on the Webis Web Archive 17 to obtain only those pages that are suitable for web page segmentation, removing "simple" pages (defined as "web pages for which a human onlooker would identify only one segment") and error pages (defined as pages with missing or wrong main content) to obtain the final dataset size of 8490 pages.

The human annotation was performed through crowd-sourcing on Amazon's Mechanical Turk platform, where "every web page has been annotated by five independent annotators." The annotators were provided with an interface that allowed for drawing rectangles on screenshots of the web pages to form segments. The instructions for the annotators were as follows:

- "Draw rectangles around parts of the page that belong together."
- "Draw separate rectangles for different parts, like for important content, controls, and ads."
- "Make sure not to miss any part."

Compared to the segmentation guidelines used in Kreuzer et al. [2015], these instructions are less restrictive while still explaining all necessary aspects of the task. Combined with the task of drawing rectangles as opposed to selecting DOM nodes, this gets out of the way of the annotator's intuition as much as possible, allowing them to focus on the page as a whole instead of specific types of elements or predefined groupings in the DOM tree. However, the inaccuracies resulting from freely drawing rectangles would be detrimental to the quality of the ground truth. For example, if two annotators produced the same segmentation, but one annotator's segments incorporated more blank space around the elements, this would affect their similarity (measured as described in Section 4.2.1) negatively for some choices of atomic elements, such as *pixels*. Therefore, the human segmentations are fit to DOM nodes in a post-processing step. A DOM node is treated "as part of a drawn segment

[1] `https://www.alexa.com`

	Atomic page elements				
Agreement measure	*pixels*	$edges_{\text{fine}}$	$edges_{\text{coarse}}$	*nodes*	*chars*
F_{B^3}	0.65	0.73	0.73	0.74	0.78
$\max(P_{B^3}, R_{B^3})$	0.94	0.96	0.96	0.95	0.97

Table 4.2: Annotator agreement by type of atomic elements and pairwise measure within the agreement measure, taken from Kiesel et al. [2019]. $\max(P_{B^3}, R_{B^3})$ replaces F_{B^3} in the agreement calculation (see Section 4.3).

if at least a fraction θ_c of its visible area overlaps." The optimal value for this threshold was determined to be 0.75, "which maximizes the F_1-score of area reconstruction", meaning this value led to the closest result to the original drawn segmentation in terms of area covered by segments.

To assess the quality of the human segmentations, an agreement measure is introduced. It is defined as "the average pairwise similarity of the segmentations" per page. Formally, it is calculated as follows:

$$\text{Agreement}(\mathcal{S}) = \frac{1}{|\mathcal{S}| \cdot (|\mathcal{S}| - 1)} \sum_{S \in \mathcal{S}} \sum_{S' \in \mathcal{S} \setminus S} F_{B^3}(S, S')$$

Table 4.2 reproduces the average annotator agreement results over the entire dataset. Apart from *pixels*, all other measures show high agreement between annotators, especially for text nodes (*chars*). The discrepancy between *pixels* and $edges_{\text{fine}}$ and $edges_{\text{coarse}}$ "shows that a significant portion of the disagreement for *pixels* is due to a different segmentation of blank space." We frequently observe similar discrepancies in our algorithm evaluations. From the $\max(P_{B^3}, R_{B^3})$ values and the property of P_{B^3} and R_{B^3} ignoring over- and undersegmentation, respectively, it is apparent that "nearly all of what little disagreement exists is due to annotators working at different levels of granularity." In comparison to the agreement considerations made in Kreuzer et al. [2015] and the evaluation of HEPS by Manabe and Tajima [2015], the given values are more clearly interpretable, as they are based on the annotator segmentations themselves, rather than an abstraction of them, and are obtained using the same clustering similarity measure used for our evaluations in Chapter 5.

To obtain a ground truth from the five annotations per page, the human segmentations are *fused* using a clustering algorithm. The result contains "those atomic elements which also the majority of annotators put in one segment." This means that for all pairs of atomic elements, "the fraction of annotators that put those two elements in one segment" is used as the similarity

measure, and the fused segments are formed "from those pairs of atomic elements with a similarity exceeding a threshold θ_s of 0.5." *pixels* were chosen as atomic elements, "as the annotators worked in a pixel-based interface." The chosen threshold $\theta_s = 0.5$ means that the fusion is essentially a majority vote, and "only those atomic elements that are in segments of at least three of the five annotators" are incorporated into the final fused segmentation.

Chapter 5

Identifying algorithm strengths and weaknesses

This chapter presents all evaluations and experiments performed for this thesis, along with analyses and interpretations of the obtained results. We first show the results of evaluating all discussed algorithms against the ground truth as in Kiesel et al. [2019], but with updated values and including the results of all further experiments performed here (Section 5.1). The results express segmentation *quality*, and show the strength of the 16 years old DOM-only VIPS algorithm, and the emergent performance of purely visual segmentation approaches tuned for web pages. In the remaining sections, we detail our experiments and analyze the performance of the algorithms in these more specific situations, establishing a connection between the measured performance and the algorithm's functional aspects responsible for it. Thus, we identify the algorithms' strengths and weaknesses: which design choices contribute to segmentation quality, and which work to its detriment.

Section 5.2 analyzes the similarity of algorithm segmentations among each other, to determine a possible correlation between algorithm performance and segmentation similarity; the results suggest such a correlation, but nevertheless further display the influence of a common fundamental operating principle on segmentation similarity. Sections 5.3 and 5.4 provide parameter analyses for VIPS and the approach by Cormier et al., respectively, to show both the effects of the examined parameters and their influence on segmentation quality; the results underscore the benefits of parameter optimization. Section 5.5 describes our experiment of fitting visual/hybrid algorithm segmentations to DOM nodes, to investigate potential benefits of combining purely or predominantly visual segmentation approaches with DOM information; results confirm that improvements in segmentation quality can be obtained through such a process. Section 5.6 shows the results of combining the best obtained algorithm

segmentations using a simple voting scheme in comparison to a previously performed fusion using unoptimized parameters, displaying the efficacy of such an approach and the combined influence of previous experiments.

In Appendix A, we show an example page with segmentation bounding boxes for all 18 different approaches and versions, as well as the ground truth (Figures A.3 to A.21). We make references to these figures where certain characteristics of the algorithm segmentations are visually prominent.

5.1 Evaluating segmentation quality

Table 5.1 contains all results from our evaluations of the five selected web page segmentation algorithms and their variations against the ground truth. In the following discussion and analysis, we separate the results into those achieved by the basic implementations of the algorithms, subdivided according to their categorization (see Chapter 3), and into those achieved by applying post-processing in the form of fitting to DOM nodes, and forming an ensemble using a simple voting approach. The table also contains baseline results that represent the performance of a single segment across the entire page for all pages of our dataset. The primary focus of our analysis is F_{B^3}, which represents segmentation *quality* per page averaged over the entire dataset. We add observations about P_{B^3} and R_{B^3} where they help to explain specific aspects of an algorithm's performance.

A general observation about the obtained results is that *chars* consistently presents the highest values. This can be explained with the fact that not all DOM tree nodes contain text, and that text commonly appears with some margin even to its parent element or the page as a whole; this reduces the influence of errors identified both when regarding *nodes* and the pixel-based atomic elements.

VIPS is best single algorithm The VIPS algorithm by Cai et al. [2003a,b] emerges as the winner among all single algorithms when used with PDoC 5, and beats the baseline for all atomic elements, also coming closest to the average segment count per page of the ground truth (9.1). For the atomic elements $edges_{\text{fine}}$, $edges_{\text{coarse}}$ and *nodes*, it offers the highest F_{B^3} values, and is only beaten among all algorithms by the Min-vote ensemble with new parameters for *pixels* and *chars*, by at most 0.02. Its precision and recall values are consistently at or above 0.63 and 0.70, respectively, except for *pixels*. The poorer performance for *pixels* is caused by the specific way the final segmentation is assembled by the algorithm. While the approach uses information from the DOM for the segmentation, and the segment borders do correspond to DOM

		Atomic elements																			
		pixels				$edges_{\text{fine}}$				$edges_{\text{coarse}}$				*nodes*				*chars*			
Algorithm	**Segments**	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}
VIPS (PDoC 5)	13.5	0.38	0.47	0.35	0.70	**0.59**	**0.67**	0.63	0.72	**0.60**	**0.68**	0.64	0.73	**0.63**	**0.70**	0.67	0.74	0.68	**0.75**	0.74	0.76
VIPS (PDoC 8)	80.2	0.32	0.40	0.46	0.36	0.46	0.53	0.81	0.39	0.47	0.54	0.83	0.40	0.43	0.49	0.89	0.34	0.50	0.57	0.93	0.41
HEPS	35.8	0.32	0.45	0.39	0.54	0.45	0.57	0.61	0.53	0.46	0.57	0.62	0.53	0.43	0.53	0.63	0.46	0.50	0.59	0.72	0.50
Cormier et al. ($s_{min} = 90$, $t_l = 512$)	18.4	0.35	0.43	0.29	0.86	0.53	0.62	0.49	0.85	0.54	0.63	0.50	0.84	0.56	0.65	0.54	0.83	0.63	0.71	0.60	0.87
Cormier et al. ($s_{min} = 45$, $t_l = 512$)	38.0	0.36	0.47	0.34	0.77	0.53	0.64	0.56	0.74	0.55	0.64	0.57	0.74	0.56	0.67	0.62	0.72	0.62	0.72	0.67	0.78
Cormier et al. (**fitted**) ($s_{min} = 45$, $t_l = 512$)	16.8	0.38	**0.54**	0.42	0.77	0.53	0.66	0.58	0.76	0.54	0.67	0.60	0.76	0.58	0.69	0.63	0.76	0.65	0.74	0.68	0.81
MMDetection	252.2	0.33	0.44	0.47	0.41	0.41	0.49	0.71	0.37	0.43	0.50	0.73	0.39	0.40	0.49	0.76	0.36	0.48	0.57	0.80	0.44
MMDetection (**fitted**)	14.7	0.35	0.48	0.67	0.38	0.40	0.50	0.76	0.38	0.42	0.52	0.77	0.39	0.48	0.57	0.75	0.46	0.56	0.64	0.80	0.54
Meier et al.	18.7	0.24	0.41	0.42	0.40	0.35	0.49	0.57	0.43	0.35	0.50	0.58	0.44	0.31	0.45	0.61	0.35	0.34	0.48	0.66	0.38
Meier et al. (**fitted**)	7.0	0.26	0.46	0.56	0.39	0.34	0.51	0.62	0.43	0.35	0.51	0.62	0.44	0.37	0.52	0.60	0.46	0.42	0.56	0.66	0.48
Old parameters																					
Min-vote@1	3.6	0.25	0.37	0.25	0.74	0.40	0.51	0.39	0.74	0.41	0.51	0.39	0.74	0.51	0.57	0.41	0.91	0.59	0.65	0.49	0.95
Min-vote@2	32.9	0.38	0.49	0.39	0.64	0.54	0.63	0.63	0.63	0.55	0.64	0.64	0.64	0.58	0.65	0.70	0.62	0.65	0.72	0.76	0.68
Min-vote@3	176.4	0.32	0.41	0.50	0.35	0.42	0.49	0.79	0.35	0.44	0.50	0.81	0.36	0.38	0.43	0.89	0.28	0.46	0.52	0.93	0.36
Min-vote@4	549.1	0.17	0.21	0.67	0.12	0.20	0.22	0.90	0.13	0.21	0.24	0.92	0.14	0.15	0.16	**0.94**	0.09	0.21	0.23	0.94	0.13
New parameters																					
Min-vote@1	6.0	0.30	0.37	0.23	**0.93**	0.49	0.56	0.40	**0.93**	0.50	0.56	0.40	**0.93**	0.52	0.57	0.41	**0.95**	0.61	0.66	0.50	**0.96**
Min-vote@2	16.0	**0.40**	0.50	0.37	0.77	0.58	**0.67**	0.59	0.77	**0.60**	**0.68**	0.61	0.78	**0.63**	**0.70**	0.64	0.77	**0.69**	**0.75**	0.71	0.80
Min-vote@3	44.3	**0.40**	0.52	0.57	0.47	0.51	0.59	0.80	0.47	0.53	0.61	0.81	0.48	0.53	0.59	0.83	0.46	0.60	0.66	0.89	0.52
Min-vote@4	52.8	0.20	0.27	**0.83**	0.16	0.24	0.29	**0.93**	0.17	0.25	0.30	**0.94**	0.18	0.26	0.30	**0.94**	0.18	0.33	0.39	**0.96**	0.24
Baseline	1.0	0.25	0.28	0.17	1.00	0.43	0.48	0.32	1.00	0.44	0.48	0.32	1.00	0.43	0.47	0.30	1.00	0.52	0.56	0.39	1.00

Table 5.1: Average number of segments per web page and evaluation results for each discussed algorithm on the dataset: 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. For reference, the ground truth contains 9.1 segments on average. The highest score in each column (excluding baseline) is highlighted in bold. The results of Meier et al. are shown in gray as its evaluation is not fully comparable.

Old parameters describes the fusion of VIPS (PDoC 8), HEPS, Cormier et al. ($s_{min} = 90, t_l = 512$) and MMDetection, representing unoptimized or default parameter choices.

New parameters describes the fusion of VIPS (PDoC 5), HEPS, Cormier et al. ($s_{min} = 45, t_l = 512$, fitted, $\theta_c = 0.75$) and MMDetection (fitted, $\theta_c = 0.75$).

borders, the segmentation is actually formed from the *visual separators* detected on the page. As can be seen in Figures A.4 and A.5, especially the segment borders at the perimeter of the page extend to the very edge of the page, a behavior already observed by Sanoja and Gançarski [2015] in their evaluation. This is simply because no further visual separators are detected along the respective axis, and the block therefore extends to the corresponding page border. This explains the large jump in precision when moving from *pixels* to the edge-based atomic elements, where VIPS starts to fare much better.

As detailed in Section 5.3, moving to PDoC 8 (the default value set in the implementation by Popela [2012]) affects the segmentation quality negatively in all metrics except precision, which benefits from the finer granularity. From the PDoC parameter analysis, we concluded that the VIPS rules that segment specific types of elements do not increase the algorithm's overall performance. Considering the high performance exhibited by VIPS with PDoC 5, we find that the rules concerned with more coarse divisions of the page, as well as other aspects of the algorithm unrelated to specific types of elements, seem to have retained their validity even 16 years after the algorithm's creation.

HEPS' narrow focus is detrimental to performance HEPS by Manabe and Tajima [2015], the other DOM-only method, performs closest to VIPS with PDoC 8 in terms of F_{B^3}, but presents higher $F^*_{B^3}$ values. While precision is lower than that of VIPS with PDoC 8, it comes close to the precision values achieved by VIPS with PDoC 5. Compared to VIPS with PDoC 8, it presents higher recall scores, explaining the higher $F^*_{B^3}$ values. Figure A.6 exhibits HEPS' strong focus on segmenting heading-content-blocks, showing its strength in finding such segments but leading it to ignore much of the rest of the page; the higher recall of HEPS versus VIPS at PDoC 8 can be explained with the lower average segment count per page stemming from this narrow focus, which however still constitutes oversegmentation compared to the ground truth. When compared to the baseline, HEPS only clearly beats it for *pixels* in terms of F_{B^3}, while the scores for other atomic elements are at worst matched (*nodes*) and at best improved by 0.02. This leads us to conclude that the narrow focus on heading-content-blocks constitutes a weakness of HEPS in this context, which is in many cases unfit for segmenting full web pages. We argue *chars* to be the best choice of atomic elements for judging text extraction performance as is the original intention of the approach; even here, HEPS is beaten in all metrics by VIPS with PDoC 5.

Cormier et al.'s algorithm approaches VIPS' performance Moving on to the purely visual methods, we first examine the edge-based approach by Cormier et al. [2017]. A detailed analysis of the s_{min} and t_l parameters

is presented in Section 5.4; we incorporate only $s_{min} = 90, t_l = 512$ px and $s_{min} = 45, t_l = 512$ px into Table 5.1, since they represent "old" or unoptimized versus "new" or optimized parameter choices, respectively. The "old" parameters were chosen by visual examination of segmentations for a random sample of pages from the dataset; however, considering the "new" parameters' performance, it becomes clear that a visual judgment of parameter choices that interact with each other in a complex manner is not suitable for an appropriate selection. We focus on the performance of this approach with parameters $s_{min} = 45, t_l = 512$ px and compare primarily to VIPS with PDoC 5, as the two are closest in performance. F_{B^3} values are lower for the edge-based approach by between 0.02 for *pixels* and 0.07 for *nodes*. Precision is consistently lower, by up to 0.07 for $edges_{\text{fine}}$, $edges_{\text{coarse}}$ and *chars*. Recall is instead consistently higher than that of VIPS at PDoC 5 with the exception of *nodes*. Thus, $F^*_{B^3}$ values see a smaller reduction, being matched for *pixels* and lower by at most 0.04 for $edges_{\text{coarse}}$. As VIPS produces a hierarchical segmentation in which the existing segments get subdivided further with increasing PDoC, we observe the typical effects of the finer granularity with the increases in precision and the decreases in recall. The approach by Cormier et al. exhibits the same behavior with the reduction of s_{min}. However, the performance of the edge-based approach only gets close to that of VIPS at PDoC 5 with almost triple the average segment count per page, serving as further evidence of the fundamentally different operating principles of both algorithms. Figure A.8 shows that the high amount of small segments detected at the top of the example page increases the segment count, but does not contribute to the quality of the segmentation. Nevertheless, our results mean that the approach by Cormier et al. comes a close second to VIPS at PDoC 5 for single algorithm performance. The good performance shown here is evidence that visual-only web page segmentation algorithms are viable alternatives to the established DOM-only methods.

Real-world instance segmentation not directly transferable MMDetection as the other purely visual segmentation approach fares significantly worse than the edge-based approach. The baseline is only beaten for *pixels* in terms of F_{B^3}; while precision is consistently high, ranging from 0.47 for *pixels* to 0.80 for *chars*, recall is among the lowest for all single algorithms, trading blows with VIPS at PDoC 8. The high segment count comes from MMDetection's tendency to recognize objects, faces or persons in images on the web pages; while this severe oversegmentation benefits precision, it still falls short of VIPS at PDoC 8 which produces less than a third of average segments per page. These results show that approaches for real-world object detection and instance segmentation are not directly applicable for web page segmentation;

the good performance of MMDetection for instance segmentation can be seen as translating to a weakness for its application in web page segmentation.

Newspaper segmentation neural network underperforms The neural network by Meier et al. [2017] fails to produce results that beat the baseline. F_{B^3} values are consistently lower than those of the baseline, and both precision and recall are among the lowest for all single algorithms. We identify several possible reasons for the low performance: (1) the required pre-processing (downsampling and cropping or padding) which reduces the input fidelity, (2) the lower quality of our DOM text mask input, and (3) insufficient training duration despite the applied overfitting detection; we cannot be certain whether the chosen `patience` value of 10 for Keras' `EarlyStopping` callback was sufficient.

Promising combination of visual segmentation and DOM information The results of our experiment that fits the segmentations of the visual/hybrid algorithms to DOM nodes (Section 5.5) show improvements in F_{B^3} for *pixels*, *nodes* and *chars*, while mostly improving in the other metrics for all atomic elements. The effects are explained in more detail in the relevant section below; we focus on the comparison to our baseline and the DOM-only algorithms here. The approach by Cormier et al. [2017] benefits in both precision and recall across the board except for *pixels*, where recall is unchanged; after fitting, it now matches VIPS with PDoC 5 for *pixels* in F_{B^3}, and beats it in the other metrics for this choice of atomic elements. For all other atomic elements, the gap to VIPS with PDoC 5 is reduced further, with consistently higher recall.

MMDetection benefits especially for *nodes* and *chars*, where it now beats the baseline in F_{B^3} and $F^*_{B^3}$ owing to the 0.10 increases in recall from the severe reduction in segment count. For all other atomic elements, we observe increases in precision such that it now beats VIPS with PDoC 8 more significantly for *pixels*, once again largely due to a different segmentation of blank space, and it comes closer for $edges_{\text{fine}}$ and $edges_{\text{coarse}}$.

After applying the fitting to segmentations produced by Meier et al.'s neural network approach, it now beats the baseline in terms of F_{B^3} for *pixels*, but still falls short for all other atomic elements. Nevertheless, the achieved precision and recall values now come closer to those of HEPS, especially for *nodes* and *chars*. Precision is most improved for the pixel-based atomic elements, where HEPS is now surpassed or matched, while recall for those elements is most comparable to that achieved by VIPS with PDoC 8.

Best overall results through algorithm combination For the Min-vote ensemble, we focus on the results achieved by the new parameter choices resulting from our parameter analyses and fitting experiment; a comparison of the ensembles with different parameter choices is found in Section 5.6. Min-vote@2 most often achieves the highest results across the dataset, with wins in 8 of 20 metrics, compared to 6 wins for VIPS with PDoC 5. While 6 of those scores are tied for the lead with VIPS at PDoC 5, Min-vote@2 achieves the best F_{B^3} values for all atomic elements except $edges_{\text{fine}}$, where it only loses to VIPS with PDoC 5 by 0.01. Notably, it achieves a 0.02 lead in precision for *pixels* over VIPS with PDoC 5, while also offering higher recall; the F_{B^3} of 0.40 is the highest for *pixels* among all algorithms and variations. The balance between precision and recall for the other atomic elements is biased towards precision for VIPS with PDoC 5, while Min-vote@2 tends more towards recall. We also highlight the performance of Min-vote@3, which fares well in comparison with VIPS at PDoC 8. The latter is beaten in all metrics for *pixels*, while the other atomic elements show close P_{B^3} values (down by 0.06 at most), with consistently improved recall.

5.2 Cross-evaluating the algorithms

As described in Section 4.2.1, the evaluation methodology proposed in Kiesel et al. [2019] is based on a *similarity* measure that can express segmentation *quality* when comparing an algorithm segmentation to the ground truth, but also allows to express the similarity between two algorithm segmentations in terms of F_{B^3}. This allows us to quantitatively compare how similarly the algorithms examined in this thesis segment the web pages in our dataset. As a consequence of the parameter analyses we perform in Sections 5.3 and 5.4, we incorporate VIPS with PDoC 5 and the approach by Cormier et al. with parameters $s_{min} = 45, t_l = 512$ px, as well as HEPS and MMDetection into this evaluation. We do not incorporate the fitted versions of segmentations produced by MMDetection and the approach by Cormier et al. into this experiment, because doing so would distort the differences stemming from the distinct operating principles of these algorithms. The neural network proposed by Meier et al. [2017] is also excluded, due to its different evaluation requirements.

The results of this evaluation are shown in Tables 5.2 to 5.6. Note that, as described in Section 4.2.1, $R_{B^3}(S, S^*) = P_{B^3}(S^*, S)$, and that F_{B^3} is symmetric.

Best single algorithms are most similar From the F_{B^3} results marked in bold, we determine that VIPS with PDoC 5 and the approach by Cormier

P_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.49	0.53	0.58
HEPS	0.57	1.00	0.58	0.67
Cormier[2]	0.74	0.66	1.00	0.77
MMDet.	0.33	0.35	0.33	1.00

F_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.41	**0.51**	0.31
HEPS	0.41	1.00	0.50	0.31
Cormier[2]	**0.51**	0.50	1.00	0.37
MMDet.	0.31	0.31	0.37	1.00

Table 5.2: Average P_{B^3} and F_{B^3} values for *pixels*

P_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.61	0.59	0.69
HEPS	0.56	1.00	0.53	0.67
Cormier[2]	0.71	0.68	1.00	0.77
MMDet.	0.37	0.38	0.34	1.00

F_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.46	**0.54**	0.36
HEPS	0.46	1.00	0.46	0.36
Cormier[2]	**0.54**	0.46	1.00	0.37
MMDet.	0.36	0.36	0.37	1.00

Table 5.3: Average P_{B^3} and F_{B^3} values for $edges_{\text{fine}}$

P_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.62	0.60	0.70
HEPS	0.57	1.00	0.53	0.67
Cormier[2]	0.71	0.67	1.00	0.77
MMDet.	0.38	0.36	0.39	1.00

F_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.46	**0.54**	0.37
HEPS	0.46	1.00	0.46	0.36
Cormier[2]	**0.54**	0.46	1.00	0.38
MMDet.	0.37	0.36	0.38	1.00

Table 5.4: Average P_{B^3} and F_{B^3} values for $edges_{\text{coarse}}$

P_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.64	0.63	0.71
HEPS	0.46	1.00	0.42	0.52
Cormier[2]	0.69	0.65	1.00	0.74
MMDet.	0.33	0.35	0.33	1.00

F_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.43	**0.55**	0.35
HEPS	0.43	1.00	0.38	0.29
Cormier[2]	**0.55**	0.38	1.00	0.35
MMDet.	0.35	0.29	0.35	1.00

Table 5.5: Average P_{B^3} and F_{B^3} values for *nodes*

P_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.70	0.66	0.72
HEPS	0.50	1.00	0.45	0.54
Cormier[2]	0.76	0.73	1.00	0.77
MMDet.	0.42	0.45	0.38	1.00

F_{B^3}	S			
S^*	**VIPS**	**HEPS**	**Cormier**	**MMDet.**
VIPS[1]	1.00	0.48	**0.60**	0.41
HEPS	0.48	1.00	0.43	0.36
Cormier[2]	**0.60**	0.43	1.00	0.40
MMDet.	0.41	0.36	0.40	1.00

Table 5.6: Average P_{B^3} and F_{B^3} values for *chars*

[1] PDoC 5

[2] $s_{min} = 45, t_l = 512$ px

et al. produce the most similar segmentations, with F_{B^3} values between 0.51 for *pixels*: and 0.60 for *chars*. Furthermore, $P_{B^3}(\text{VIPS}, \text{Cormier et al.})$ values are high, ranging from 0.69 for *nodes* to 0.76 for *chars*. $P_{B^3}(\text{Cormier et al.}, \text{VIPS})$ is lower however: between 0.53 for *pixels* and 0.66 for *chars*. These results are noteworthy insofar as these two algorithms operate fundamentally differently; VIPS being a DOM-only approach that uses heuristics based on DOM information, and the approach by Cormier et al. being purely visual and based on edge detection. The high $P_{B^3}(\text{VIPS}, \text{Cormier et al.})$ values are achieved despite the approach by Cormier et al. producing almost three times the amount of average segments per page; in comparison, VIPS therefore undersegments on average, which affects this value negatively. When comparing to $P_{B^3}(\text{Cormier et al.}, \text{VIPS})$ and considering the differences between the values for *pixels* and $edges_{\text{fine}}$ and $edges_{\text{coarse}}$, it becomes apparent that a significant part of the difference is due to differently segmented blank space. Thus, the similarity between these two algorithms would increase further when fitting the segmentations produced by the purely visual approach to DOM nodes. However, as observed in Section 5.1, the fact that VIPS with PDoC 5 achieves better performance compared to the ground truth while only producing approximately one third of the number of segments of the edge-based approach expresses the significant differences in operating principles.

HEPS: influence of segment borders and fundamental principle When considering the other DOM-only approach examined in this thesis, HEPS by Manabe and Tajima [2015], we observe that it is more similar to the approach by Cormier et al. for *pixels* with respect to F_{B^3} than to VIPS. P_{B^3}(HEPS,Cormier et al.) is also significantly higher for *pixels* than P_{B^3}(HEPS, VIPS), but the difference reduces greatly for all other atomic elements. Once again, this suggests that these discrepancies are largely down to the differences in segmenting blank space. A part of the reason for this is that VIPS does not directly base its final segmentation on DOM node boundaries, but on the detected visual separators on the page. For all other atomic elements, HEPS is at least as similar to VIPS in F_{B^3} as to the approach by Cormier et al. (for $edges_{\text{fine}}$ and $edges_{\text{coarse}}$), and more similar to VIPS for *nodes* and *chars*. The lower F_{B^3} between HEPS and VIPS versus between VIPS and the approach by Cormier et al. can be attributed to HEPS' focus on finding and segmenting heading-content blocks, while VIPS and the edge-based approach are not specifically tuned for this purpose. HEPS' narrower focus is also expressed in the lower R_{B^3} values for *nodes* and *chars*. However, R_{B^3}(HEPS,VIPS) is higher than R_{B^3}(HEPS,Cormier et al.) for all but *pixels*, despite HEPS producing a significantly higher average number of segments per page than VIPS, while being more similar to the approach by Cormier et al. in average segment

count. P_{B^3}(HEPS,VIPS) is also higher than P_{B^3}(HEPS,Cormier et al.) for all but *pixels*. These results therefore suggest that despite the larger similarity in F_{B^3} between VIPS and the edge-based approach considering their fundamentally different operating principles, the common basic approach (DOM-only versus purely visual) remains a strong influence, causing HEPS to reach R_{B^3}(HEPS,VIPS) values comparable to those of R_{B^3}(VIPS,Cormier et al.) for the pixel-based atomic elements; this is accomplished even though HEPS is designed with a narrower focus, and oversegments compared to VIPS based on their average segment counts per page.

MMDetection is fundamentally different Finally, MMDetection offers results conforming to our expectations, with high precision to all other algorithm segmentations especially for the pixel-based atomic elements, but the lowest recall for all atomic elements, and consequently the lowest similarity to all other algorithms. The much higher average segment count per page expresses the oversegmentation that is in most cases caused by segmenting objects detected in images on the page, leading to the high precision but low recall values. For the pixel-based atomic elements, MMDetection is however most similar in terms of F_{B^3} to the equally purely visual approach by Cormier et al., though only by a small margin for all but *pixels*. This could be seen as further evidence that the category the algorithm belongs to has a strong influence on its similarity to other algorithms, even in the face of major differences in all other functional aspects. Nevertheless, these results further emphasize that MMDetection is normally unfit for web page segmentation and operates significantly differently to all other algorithms in this evaluation.

In summary, these results corroborate those shown in Section 5.1 insofar as the two algorithms that on their own produce some of the highest quality segmentations compared to the ground truth, VIPS and the approach by Cormier et al., also show the highest similarity in F_{B^3} when compared to each other. We highlight that the edge-based approach manages to reach this similarity without access to any of the information that VIPS uses in the segmentation process; in combination with the results from evaluating against the ground truth, this substantiates that the assumptions and design considerations made by Cormier et al. are well-suited for web page segmentation. The differences in segmentation approach between the two DOM-only methods are also expressed in these results, as is the fundamentally different nature of segmentations produced by MMDetection.

		Atomic elements																			
		pixels				$edges_{\text{fine}}$				$edges_{\text{coarse}}$				*nodes*				*chars*			
PDoC	**Segments**	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}
11	111.33	0.28	0.36	0.46	0.30	0.40	0.47	0.82	0.33	0.41	0.48	0.83	0.34	0.34	0.40	0.89	0.26	0.42	0.49	0.94	0.33
10	111.11	0.28	0.36	0.46	0.30	0.40	0.47	0.82	0.33	0.41	0.48	0.83	0.34	0.34	0.40	0.89	0.26	0.42	0.49	0.94	0.33
9	88.91	0.30	0.38	0.46	0.33	0.44	0.51	0.81	0.37	0.45	0.51	0.83	0.37	0.40	0.46	0.89	0.31	0.47	0.54	0.94	0.38
8	80.16	0.32	0.40	0.46	0.36	0.46	0.53	0.81	0.39	0.47	0.54	0.83	0.40	0.43	0.49	0.89	0.34	0.50	0.57	0.93	0.41
7	45.60	0.36	0.45	0.43	0.48	0.54	0.61	0.77	0.51	0.55	0.62	0.78	0.52	0.55	0.62	0.83	0.49	0.60	0.67	0.89	0.54
6	**16.09**	**0.38**	**0.47**	**0.36**	**0.67**	**0.59**	**0.67**	**0.66**	**0.69**	**0.60**	**0.68**	**0.70**	**0.67**	**0.63**	**0.70**	**0.70**	**0.71**	**0.68**	**0.75**	**0.77**	**0.73**
5	**13.52**	**0.38**	**0.47**	**0.35**	**0.70**	**0.59**	**0.67**	**0.63**	**0.72**	**0.60**	**0.68**	**0.64**	**0.73**	**0.63**	**0.70**	**0.67**	**0.74**	**0.68**	**0.75**	**0.74**	**0.76**
4	12.38	0.38	0.46	0.34	0.71	0.59	0.67	0.62	0.73	0.60	0.68	0.63	0.73	0.63	0.70	0.66	0.75	0.68	0.74	0.73	0.76
3	12.32	0.38	0.46	0.34	0.71	0.59	0.67	0.62	0.73	0.60	0.68	0.63	0.73	0.63	0.70	0.66	0.75	0.68	0.74	0.73	0.76
2	12.27	0.38	0.46	0.34	0.71	0.59	0.67	0.62	0.73	0.60	0.68	0.63	0.73	0.63	0.70	0.66	0.75	0.68	0.74	0.73	0.76
1	12.20	0.38	0.46	0.34	0.71	0.59	0.67	0.62	0.73	0.60	0.68	0.63	0.74	0.63	0.70	0.66	0.75	0.68	0.75	0.73	0.77

Table 5.7: Evaluation of all possible Permitted Degrees of Coherence (PDoC) of VIPS [Cai et al., 2003a,b] against the ground truth, using our port based on the implementation of Popela [2012]. Rows in bold show the best results.

5.3 Parameter analysis: VIPS

As detailed in our overview of the VIPS algorithm by Cai et al. [2003a,b], the granularity of the approach can be influenced by altering the *Permitted Degree of Coherence* (PDoC) parameter. We evaluate all possible PDoC values of our TypeScript/JavaScript port of Popela's Java implementation of VIPS; the results are shown in Table 5.7. This parameter acts as a terminating condition for the recursive division of blocks; the higher the PDoC, the more fine-grained the segmentation. This is easily visible when observing the average number of segments per page, which increases with greater PDoC, although not linearly. In fact, the average segment count makes large jumps from PDoCs 6 to 7, 7 to 8, and 9 to 10. This, in turn, has a significant effect on the algorithm's performance. PDoC values of 7 and 8 cause gains in precision for all atomic elements, but come with large sacrifices in recall. As described in Section 4.2.1, by comparison with F_{B^3} we can see from this that starting with PDoC 7, oversegmentation compared to the ground truth is the primary reason for the decreasing performance. This is consistent with the fact that with increasing PDoC, the blocks of a previous PDoC level get subdivided further, but no segments are added outside of the existing segment boundaries, which are already defined at PDoC 1. Increasing PDoC beyond 8 has only insignificant benefits for precision, if any, and causes further losses in recall due to oversegmentation.

This behavior can be largely attributed to the rules that VIPS operates with. We analyze which rules get applied for which PDoC, and from this infer the main causes for the large jumps in average segment count. In the following, we give rule numbers according to [Cai et al., 2003b, Section 4.1, Table 1]; however, in describing the effect of these rules, we adhere to the

actual implementation by Popela [2012] used as the base for our port, which notably omits rule 7 and applies rule 13 to `p` and inline text nodes in addition to `table`, `td` and `tr` nodes.

Rules 4, 8, 9, 10, and 13 influence the DoC of the block they are applied to. Rule 4 is applied if a block only contains text nodes as children; the DoC is set to 10 if all children have the same font size and font weight, and to 9 otherwise. Rule 8 sets the DoC of a block to 7 if the block contains children with a different background color. Rule 9 limits the divisibility of blocks smaller than a predefined size threshold (80×80 px in the implementation by Popela and our port), and sets the DoC to a value between 5 and 8; `Xdiv` and `code` elements are assigned a DoC of 7, the common `div` elements a DoC of 5, and other types of nodes a DoC of 8. Rule 10 limits the divisibility of blocks where all child nodes are smaller than the predefined size threshold; `Xdiv` nodes are assigned a DoC of 7, while other types of nodes are assigned a DoC of 8. Rule 13 marks certain types of nodes as indivisible and assigns them a fixed DoC; these are: `Xdiv` (DoC 7), `li`, `span`, `sup` and `img` (all DoC 8).

From these rules, we form an explanation for the jumps occurring between PDoCs 6 to 11. The number of segments makes its first large jump with PDoC 7. Rule 9 includes setting the DoC for small `code` elements to 7. 116 pages of our dataset contain one or more `code` elements. A visual review of the ground truth segmentations for all of these pages revealed no case where the `code` element was alone in a segment. In most cases, the `code` element only consists of a few lines at most, and appears alongside multiple other `code` elements broken by regular text on pages providing, for example, programming tutorials or code documentation. As such, the `code` elements are treated in the ground truth as part of the text around them, and are often part of the largest segment delineating the main content. On pages that mainly display code, such as some pages on GitHub, the `code` element is in all cases part of a segment containing additional information such as the filename of the displayed code, sometimes including its commit information and author, displayed above the code. An example of this can be seen in Figure 5.1. As such, all cases where VIPS puts `code` elements in segments of their own are oversegmented compared to the ground truth.

Since no page in our dataset contains any `Xdiv` elements, the only other rule responsible for the significant increase in average segment count per page at PDoC 7 is rule 8. From the decrease in performance compared to our ground truth, we can infer that a change in background color does not necessarily indicate the presence of a new coherent semantic unit; this rule therefore also leads to oversegmentation compared to the ground truth.

Another large jump in average segment count per page occurs from PDoC 7 to 8. Once a PDoC of 8 is reached, many common elements, such as `li`

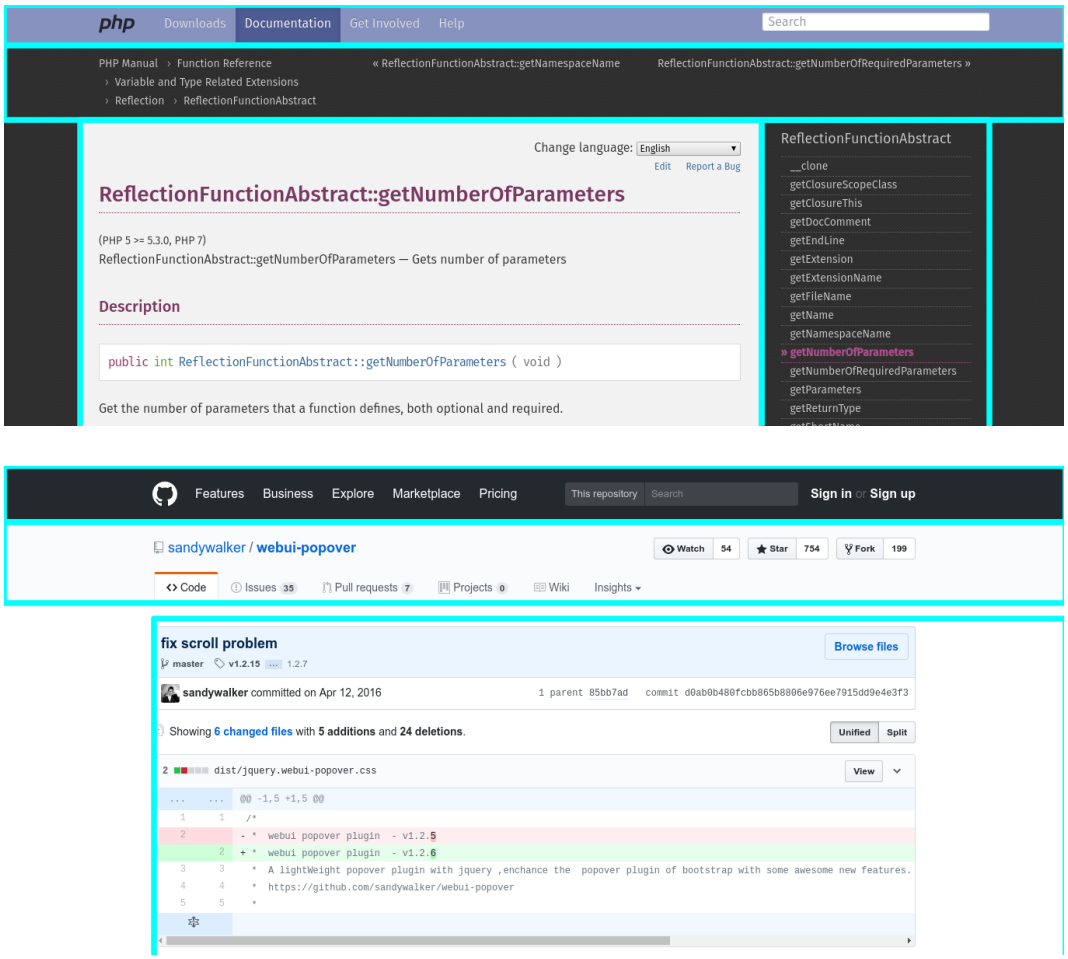


Figure 5.1: Excerpt of two web pages (top: PHP documentation, bottom: GitHub) containing a `code` element, showing the ground truth segmentation. In both cases, the `code` element is part of a large main segment that extends further down the page, and does not make up a segment on its own. In the top page, the `code` element is small enough to be affected by VIPS rule 9, which would put it in its own segment.

(list items), `span`s and images (`img`) make up their own segments following rule 13. This is also the point where P_{B^3} reaches its maximum (for *pixels*, $edges_{\text{coarse}}$ and *nodes*) or gets very close to it (for $edges_{\text{fine}}$ and *chars*), because most common types of elements now make up their own segment, and further increases in PDoC proceed to show the influence of rules that segment only text nodes, yielding no precision gains but causing further drops in recall from additional oversegmentation.

Finally, rule 4 is partially responsible for the increases in average segment count per page for PDoCs 9 and 10. Blocks with PDoC 9 contain text with different font sizes and font weights; thus, they may be larger than blocks with PDoC 10, where only text with a common font size and weight is contained.

PDoC 11 is set for only three types of blocks: blocks whose only child is an `em` (emphasis) element, `a` (hyperlink) nodes smaller than the predefined size threshold, and any blocks that cannot be further divided by application of the rules. Since the average segment count only barely increases from PDoC 10 to 11, such cases seem to be rare.

The final Degree of Coherence of a block produced by VIPS is however not solely dependent on the application of the aforementioned rules. When VIPS detects the separators between the identified blocks, weights are assigned to these separators based on their size; these weights are another influence on the DoC of the parent block in the visual structure. Smaller separators translate to a higher DoC, because the distance between the blocks they separate is low, implying higher coherence between the two blocks. Larger separators therefore cause a lower DoC.

When assembling the final segmentation, the DoC of a block is set to the minimum of its own DoC, its children blocks' DoCs and its contained separators' DoCs. While the rules are the largest direct influence on the DoC as shown above, the influence of separator weights on the DoC is solely or majorly responsible for the increases in average segment count per page occurring up to PDoC 6. Furthermore, since the aforementioned rules only set the DoC of a block to values >= 5, we conclude that VIPS' best performance is shaped largely by the application of rules that do not directly influence the DoC, and by the separator detection. The rules directly influencing the DoC for specific types of elements therefore do not serve to improve the algorithm's general performance and only improve precision at the cost of recall, overall diminishing the quality of the resulting segmentation.

			Atomic elements																	
			pixels				$edges_{\text{fine}}$				$edges_{\text{coarse}}$				*nodes*				*chars*			
s_{min}	t_l	**Segments**	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}
45	256px	18.00	0.34	0.42	0.27	0.90	0.53	0.61	0.46	0.89	0.54	0.62	0.47	0.89	0.55	0.64	0.50	0.87	0.63	0.70	0.57	0.91
45	**512px**	**38.01**	**0.36**	**0.47**	**0.34**	**0.77**	**0.53**	**0.64**	**0.56**	**0.74**	**0.55**	**0.64**	**0.57**	**0.74**	**0.56**	**0.67**	**0.62**	**0.72**	**0.62**	**0.72**	**0.67**	**0.78**
90	256px	8.86	0.33	0.41	0.26	0.92	0.52	0.60	0.45	0.91	0.53	0.61	0.46	0.90	0.55	0.62	0.48	0.89	0.63	0.69	0.55	0.93
90	512px	18.35	0.35	0.43	0.29	0.86	0.53	0.62	0.49	0.85	0.54	0.63	0.50	0.84	0.56	0.65	0.54	0.83	0.63	0.71	0.60	0.87

Table 5.8: Evaluation of all combinations of suggested values for parameters s_{min} and t_l for the approach presented in Cormier et al. [2017]

5.4 Parameter analysis: Cormier et al.

In the implementation of the approach by Cormier et al. [2017] kindly provided by the authors, a set of suggestions for the s_{min} (minimum segment border size) and t_l parameters are made. We evaluate all combinations of these suggestions and present our results in Table 5.8.

The parameter set $s_{min} = 45$, $t_l = 512$ px produces the best results in terms of F_{B^3}, $F^*_{B^3}$ and P_{B^3}, but scores lower in R_{B^3} than all other sets of parameters. The drop in recall is caused by oversegmentation, which is corroborated by the significant increase in average segment count per page.

The combination of $s_{min} = 90$ and $t_l = 256$ px produces the best results in terms of recall. However, this is in large part due to the nature of the segmentation produced by the algorithm (X-Y tree, entire page as root). Indeed, parameters $s_{min} = 90$ and $t_l = 256$ px produce only a single segment across the entire page for 2409 pages, or 28.4% of our dataset, while the choice of $s_{min} = 45$, $t_l = 512$ px does so for only 737 pages, or 8.7% of our dataset; $s_{min} = 45, t_l = 256$ px creates only one segment for 1943 pages or 22.9% of the dataset, and $s_{min} = 90, t_l = 512$ px creates only one segment for 1513 pages or 17.8% of the dataset.

To better explain the differences in segmentation between the parameter choices, we detail the influence of each parameter with the aid of Figures 5.2 and 5.3.

s_{min}: granularity and interaction with t_l Beside making apparent the finer granularity caused by decreasing s_{min}, Figure 5.2 shows the interaction between s_{min} and t_l. For instance, when comparing the segmentation around the IMDb logo, the existing segment borders shift slightly when decreasing s_{min}. Because smaller segments are now allowed, the leftmost border of the IMDb logo (transition from the yellow logo to the dark gray background) is now seen as part of an *extended line*, meaning a proposed line on the web page that is not visually continuous but may still be a significant segment border. This extended line also includes the left border of the movie ad below, and the

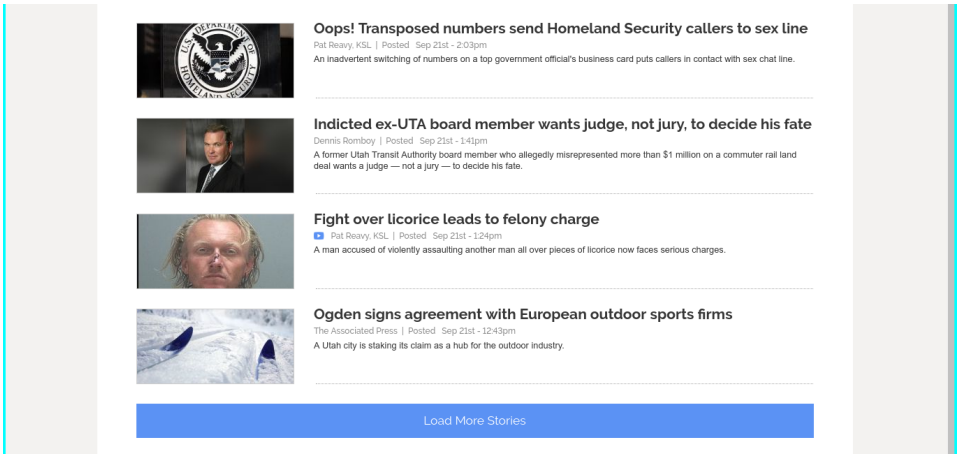
remaining space in the header to the left of the logo is now a segment. The smaller minimum region size also causes the new segments that exist between the vertical borders of text and the actual vertical edges beneath the search bar. Furthermore, the horizontal segment borders between the lines of text beneath the search bar are similarly part of extended lines; one extended line starting from the bottom edge of the IMDb logo text, and one extending past the bottom edge of the "Sign in with Facebook" text.

t_l: extended lines as segment borders The t_l parameter (maximum line segment length for Monte Carlo simulations) influences the detection of the aforementioned extended lines on the page. The primary observation is that when increasing its value, the segmentation becomes more fine-grained, as evidenced by the increase in average segment count per page. This is because it is now more likely that parts of extended lines are included in a line segment of length t_l. Longer line segments thus have a greater probability of bridging the gap between visually prominent parts of an extended line. The Monte Carlo simulation performed on such a line segment, estimating its significance probability from edge probabilities at each of the line's pixels, will therefore return a higher value even if the line segment crosses some amount of blank space. In contrast, smaller values for t_l cause line segments to more often contain only one part of an extended line, or to fall into the blank space between two parts of an extended line. In the latter case, the significance probability for that line segment becomes 0; because the significance probability for the entire proposed line is the product of the probabilities of its segments, this causes the entire proposed line to be discarded. In both cases, the amount of proposed lines with a significance probability > 0.5 is reduced, leaving fewer segment boundary candidates when forming the final segmentation tree and resulting in a coarser segmentation. Figure 5.3 visualizes this influence. The segmentation with $t_l = 256$ px detects no segment borders between the individual articles, while its counterpart with $t_l = 512$ px recognizes some of the gray edges at the articles' bottom as part of an extended line incorporating the bottom edge of the article image. However, note the missing borders below the first and last articles in the segmentation with $t_l = 512$ px. Their absence suggests that the properties of the article image are responsible, since the gray line is present for all articles. The bottom article is associated with a very bright picture containing snow, while the top article's picture sports the white seal of the U.S. Department of Homeland Security on a dark background. The low contrast of the bright snow picture compared to the white background causes lower edge probabilities at each pixel of the image's bottom edge, in turn lowering the significance probability of the proposed line segment of length t_l, and with it, the significance probability of the entire proposed line. Even the

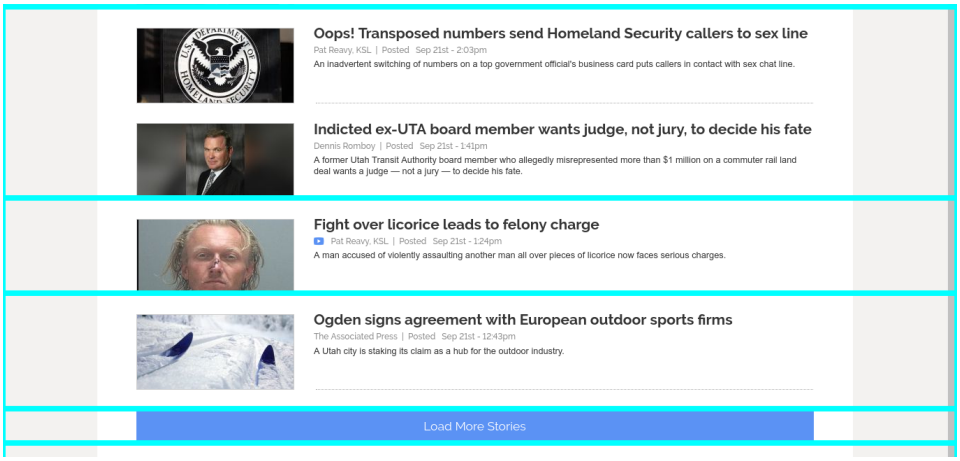
(a) $s_{min} = 90, t_l = 256$ px

(b) $s_{min} = 45, t_l = 256$ px

Figure 5.2: Header of an IMDb page, showing the difference in granularity between $s_{min} = 90$ and $s_{min} = 45$.



(a) $s_{min} = 45, t_l = 256$ px



(b) $s_{min} = 45, t_l = 512$ px

Figure 5.3: Excerpt taken from the bottom of the page shown in Figure A.2, showing the influence of the t_l parameter.

small section of the white seal that borders on the background is enough to reduce the probability of the proposed extended line there, such that it does not cross the threshold of $t_p = 0.5$, and is therefore not accepted as a segment border in the final segmentation.

From our findings related to these parameters, we conclude that the balance between s_{min}, t_l and the t_p parameter (not discussed here) is rather delicate. Especially the t_l parameter is dependent on the visual characteristics of each web page. This alone makes it impossible to determine a parameter set that is "ideal" for all web pages, constituting a weakness of this approach. However, the high number of parameters compared to the other approaches presented here highly benefits adaptability, which can be seen as a strength of this algorithm. Information from the DOM may be suitable to tune the t_l parameter on a per-page basis; we have not attempted this.

5.5 Visual segmentations and DOM information

The human annotators that helped create the ground truth segmentations for our dataset worked with a web interface in which they freely drew rectangular segments on a screenshot of the web page. This is fundamentally comparable to the visual approaches presented here in that no DOM information or any other input was used in the segmentation process. The raw human segmentations are fit to DOM nodes in a post-processing step, as described in Section 4.3. This can be interpreted as combining DOM information with a purely visual approach; for the annotations, the purpose of this process was to eliminate the influence of differently segmented blank space.

We examine the effects of applying this fitting to the purely visual and hybrid approaches discussed here. While the neural network by Meier et al. [2017] uses DOM information about the text nodes on the page in our scenario, it is fundamentally unaware of all other DOM node borders; therefore, we also incorporate this approach into the experiment. We fit the algorithm segmentations to DOM nodes using the same containment threshold $\theta_c = 0.75$ that was used for fitting the human segmentations. Different values for this threshold were not considered; we prioritized examining the effects of node fitting in general over a parameter optimization of θ_c.

Segment count reduction and general performance effects The results of evaluating the fitted segmentations of the visual and hybrid approaches examined in this thesis are presented in Table 5.9, alongside the non-fitted results for comparison. The most obvious difference is that average segment count per page is reduced for all algorithms. This is due to the removal of

		Atomic elements																			
		pixels				$edges_{\text{fine}}$				$edges_{\text{coarse}}$				*nodes*				*chars*			
Algorithm	**Segments**	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}
Cormier et al.																					
non-fitted	38.0	0.36	0.47	0.34	0.77	0.53	0.64	0.56	0.74	0.55	0.64	0.57	0.74	0.56	0.67	0.62	0.72	0.62	0.72	0.67	0.78
fitted	16.8	0.38	0.54	0.42	0.77	0.53	0.66	0.58	0.76	0.54	0.67	0.60	0.76	0.58	0.69	0.63	0.76	0.65	0.74	0.68	0.81
MMDetection																					
non-fitted	252.2	0.33	0.44	0.47	0.41	0.41	0.49	0.71	0.37	0.43	0.50	0.73	0.39	0.40	0.49	0.76	0.36	0.48	0.57	0.80	0.44
fitted	14.7	0.35	0.48	0.67	0.38	0.40	0.50	0.76	0.38	0.42	0.52	0.77	0.39	0.48	0.57	0.75	0.46	0.56	0.64	0.80	0.54
Meier et al.																					
non-fitted	18.7	0.24	0.41	0.42	0.40	0.35	0.49	0.57	0.43	0.35	0.50	0.58	0.44	0.31	0.45	0.61	0.35	0.34	0.48	0.66	0.38
fitted	7.0	0.26	0.46	0.56	0.39	0.34	0.51	0.62	0.43	0.35	0.51	0.62	0.44	0.37	0.52	0.60	0.46	0.42	0.56	0.66	0.48

Table 5.9: Comparison of non-fitted vs. fitted ($\theta_c = 0.75$) results for the visual/hybrid algorithms (Cormier et al. [2017], MMDetection [Chen et al., 2019a], Meier et al. [2017]).
For the approach by Cormier et al., parameters $s_{min} = 45$ and $t_l = 512$ were used. Results for the neural network by Meier et al. shown separately due to the different evaluation requirements.

segments that cannot be fit to a corresponding DOM node with respect to the given threshold. Above all, this affects MMDetection significantly, reducing its average segment count per page from 252.2 to just 14.7, meaning a 94.2% reduction. The previously high segment count comes from the object detection origins of the approach, as the algorithm often detects real-world objects or faces in images on the web page, which are usually significantly smaller than the image (and its DOM node) itself, causing such segments to be discarded when fitting. To a lesser degree, this reduction in segment count also applies to the approaches of Cormier et al. and Meier et al..

With *pixels* as atomic elements for the clustering similarity measure, all algorithms report gains in F_{B^3}, $F^*_{B^3}$ and P_{B^3}. Precision sees the most significant gains, increasing by 0.08 for the approach by Cormier et al., by 0.14 for the approach by Meier et al., and by 0.20 for MMDetection, while recall is not impacted or only slightly decreased. For $edges_{\text{fine}}$ and $edges_{\text{coarse}}$, we see consistent or only insignificantly reduced F_{B^3} values, while observing gains in precision in all cases, and consistent or slightly improved recall; thus, $F^*_{B^3}$ is also increased. When comparing the gains in precision for *pixels* versus $edges_{\text{fine}}$ and $edges_{\text{coarse}}$, we observe a stronger increase for *pixels* than for the edge-based measures. This reveals that precision for *pixels* originally suffered at least in part from differently segmented blank space compared to the ground truth, analogous to the observations made for annotator agreement in Section 4.3, Table 4.2.

nodes sees the largest improvements in recall and F_{B^3}, with precision only being affected by ± 0.01. The recall improvements are due to the reduced oversegmentation as a result of discarding segments that cannot be fit to any

element. As a result, $F^*_{B^3}$ increases for all algorithms. Finally, the choice of *chars* as atomic elements reports gains in all but P_{B^3}, most significantly for MMDetection and the neural network by Meier et al..

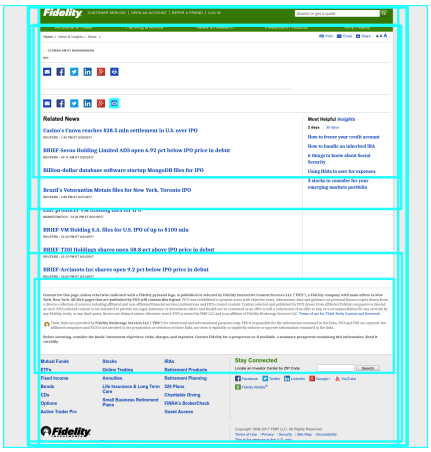
Visual analysis Figure 5.4 shows the before and after of fitting the algorithm segmentations to DOM nodes on an example page. For the approach by Cormier et al., we observe no significant reduction in segment count. The fitted segmentation only misses one segment from the original in the right pane showing links to related articles, since it did not contain at least 75% of the element's area. The segmentation mostly retains its appearance after fitting, meaning that the original segmentation was already well-aligned with element borders; this suggests that the visual edges detected by Cormier et al.'s approach correspond well to DOM node bounding boxes, and further substantiates that the algorithm is indeed well tuned for segmenting web pages.

The result of fitting the segmentation produced by MMDetection to DOM nodes shows a more apparent reduction in segment count. In the original segmentation, several segments that overlap to a large degree can be seen; the fitting process results in a simplified structure without overlapping segments. Furthermore, the chosen containment threshold causes the header (green) to be segmented in its entirety after fitting, as opposed to the previous segmentation that cut through it. This also applies to most other segment borders, which were previously not well-aligned with most DOM element borders. Thereby, the fundamentally different operating principles of MMDetection versus the approach by Cormier et al. are emphasized: while the latter's segmentation process is specialized for web pages through a number of restrictions, the former is not tuned for axis-aligned rectangles, but instead for free-form objects with no special awareness for axis-parallel edges, causing the segmentation to be misaligned with respect to DOM node borders and to cut through elements more often. For the atomic elements *pixels*, $edges_{\text{fine}}$ and $edges_{\text{coarse}}$, MMDetection therefore sees larger gains in precision than the approach by Cormier et al.. The very slight reduction in precision for *nodes* is caused by MMDetection cutting through elements which are thereafter not contained in a segment to at least 75%, resulting in an exclusion from that segment after fitting.

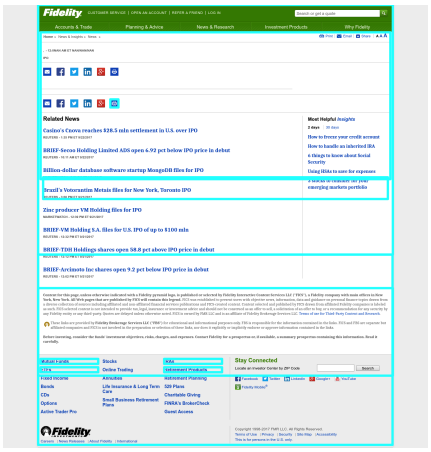
For the neural network proposed by Meier et al., the fitting process has a "clean-up" effect on the segmentation. Beside being somewhat misaligned originally, the segmentation also contains segments that do not correspond to any DOM node; these are artifacts of either the inference or post-processing (binarization and connected component discovery) steps. Such segments are removed in the fitted segmentation, best observed on the two roughly equally-sized small segments in the top quarter of the unfitted Meier et al. segmentation; one of them is fit to the rightmost social media button, while the other

Cormier et al., $s_{min} = 45, t_l = 512$ px

Cormier et al., $s_{min} = 45, t_l = 512$ px, fitted

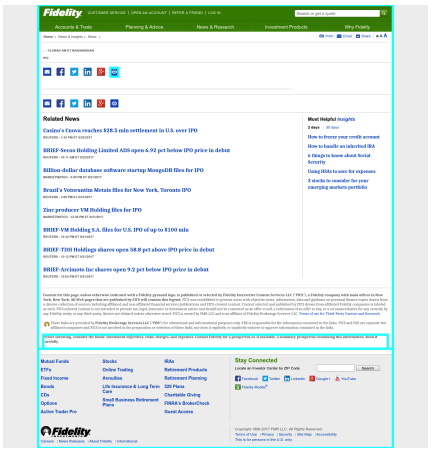


MMDetection



MMDetection, fitted

Meier et al.



Meier et al., fitted

Figure 5.4: Visual comparison of segmentations produced by non-fitted vs. fitted versions of the visual and hybrid approaches discussed in this thesis.

covers only blank space and is therefore removed. Similar to MMDetection, further gains in precision for *pixels*, $edges_{\text{coarse}}$ and $edges_{\text{fine}}$ are due to the exclusion of blank space at the borders in the fitted segmentation, and a slight reduction in precision for *nodes* can as well be attributed to segment borders cutting through elements. This occurs despite the algorithm obtaining additional input about text borders from the DOM, suggesting that either the post-processing or the DOM-based input can be improved. As mentioned in Section 3.4, the original application for newspaper segmentation also contains all horizontal and vertical lines on the page in the text mask input, and some additional post-processing is applied; similarly applying this to web pages may reduce misaligned segment borders.

We conclude that the addition of DOM information to the segmentation process for these visual and hybrid algorithms is largely beneficial to segmentation quality. This is especially true for precision with pixel-based atomic elements for the clustering similarity measure, due to the reduction of differently segmented blank space. From these findings and the conclusion made in Section 5.4, we see significant potential in augmenting the purely visual approach by Cormier et al. [2017] with DOM information to adjust parameters and improve the quality of the resulting segmentation. This of course eliminates the benefit of independence from the rendering engine; however, since the screenshot of the web page to be segmented is produced by a rendering engine, saving a snapshot of the DOM alongside it is sufficient and constitutes a lower complexity than producing a full archive of the web page.

5.6 Improved Min-vote ensemble

In [Kiesel et al., 2019, Section 6.1], the application of the segmentation fusion used to obtain the ground truth from the crowd-sourced human annotations is proposed for the *algorithm* segmentations, forming the *Min-vote ensemble*. In contrast to the ground truth creation described in Section 4.3, where the segmentations of all five annotators were fused, we examine the results of all possible $n \in [1, 2, 3, 4]$, where n is the *annotator threshold*, the minimum number of annotators (algorithms, in this case) that put a given atomic element in a segment. We therefore modify θ_s accordingly; where it was 0.5 for the ground truth creation to realize a majority vote, we adjust it according to n such that $\theta_s = \frac{n-0.5}{4}$. Because our implementation of the neural network by Meier et al. does not produce a segmentation over the entire page for all pages in our dataset, the approach is not included in our ensemble.

As we perform a parameter analysis for VIPS and the approach by Cormier et al. in this thesis, we show the difference between the ensemble results with

		Atomic elements																			
		pixels				$edges_{\text{fine}}$				$edges_{\text{coarse}}$				*nodes*				*chars*			
n	**Segments**	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}	F_{B^3}	$F^*_{B^3}$	P_{B^3}	R_{B^3}
Old parameters																					
1	3.6	0.25	0.37	0.25	0.74	0.40	0.51	0.39	0.74	0.41	0.51	0.39	0.74	0.51	0.57	0.41	0.91	0.59	0.65	0.49	0.95
2	32.9	0.38	0.49	0.39	0.64	0.54	0.63	0.63	0.63	0.55	0.64	0.64	0.64	0.58	0.65	0.70	0.62	0.65	0.72	0.76	0.68
3	176.4	0.32	0.41	0.50	0.35	0.42	0.49	0.79	0.35	0.44	0.50	0.81	0.36	0.38	0.43	0.89	0.28	0.46	0.52	0.93	0.36
4	549.1	0.17	0.21	0.67	0.12	0.20	0.22	0.90	0.13	0.21	0.24	0.92	0.14	0.15	0.16	**0.94**	0.09	0.21	0.23	0.94	0.13
New parameters																					
1	6.0	0.30	0.37	0.23	**0.93**	0.49	0.56	0.40	**0.93**	0.50	0.56	0.40	**0.93**	0.52	0.57	0.41	**0.95**	0.61	0.66	0.50	**0.96**
2	16.0	**0.40**	0.50	0.37	0.77	**0.58**	**0.67**	0.59	0.77	**0.60**	**0.68**	0.61	0.78	**0.63**	**0.70**	0.64	0.77	**0.69**	**0.75**	0.71	0.80
3	44.3	**0.40**	**0.52**	0.57	0.47	0.51	0.59	0.80	0.47	0.53	0.61	0.81	0.48	0.53	0.59	0.83	0.46	0.60	0.66	0.89	0.52
4	52.8	0.20	0.27	**0.83**	0.16	0.24	0.29	**0.93**	0.17	0.25	0.30	**0.94**	0.18	0.26	0.30	**0.94**	0.18	0.33	0.39	**0.96**	0.24

Table 5.10: Results of forming the Min-vote ensemble as proposed in Kiesel et al. [2019] (best overall n and best single results in bold).
Old parameters describes the fusion of VIPS (PDoC 8), HEPS, Cormier et al. ($s_{min} = 90, t_l = 512$) and MMDetection, representing unoptimized or default parameter choices.
New parameters describes the fusion of VIPS (PDoC 5), HEPS, Cormier et al. ($s_{min} = 45, t_l = 512$, fitted, $\theta_c = 0.75$) and MMDetection (fitted, $\theta_c = 0.75$).

unoptimized parameters (VIPS: PDoC 8, Cormier et al.: $s_{min} = 90, t_l = 512$ px) and optimized parameters (VIPS: PDoC 5, Cormier et al.: $s_{min} = 45, t_l = 512$ px). Furthermore, for the approach by Cormier et al. and MMDetection, we use the fitted segmentations for the new ensemble, since they offer an improvement over the algorithm's original performance as detailed in Section 5.5. The results of evaluating the ensemble with both old and new parameter choices are shown in Table 5.10. In the following, we focus on a comparison to the ensemble using the old parameters; a general interpretation of the results in the context of all evaluations performed in this thesis can be found in Section 5.1.

Segment count reduction and its effects Starting from the average segment count per page, the old and new ensemble present significant differences. For all but Min-vote@1, the segment count is significantly reduced in the new ensemble. This is mainly caused by the decreases in segment count from fitting the algorithm segmentations to DOM nodes, especially for MMDetection, which sees a 94.2% reduction as detailed in Section 5.5. Consequently, oversegmentation is drastically reduced, which reflects in all R_{B^3} results.

For all atomic elements, R_{B^3} is thus increased compared to the old ensemble; where the old parameter choices would cause recall to drop as low as 0.09 for *nodes* at $n = 4$, the lowest recall value for the new parameter choices is 0.16 for *pixels* at $n = 4$. In general, recall values between the pixel-based atomic elements *pixels*, $edges_{\text{fine}}$ and $edges_{\text{coarse}}$ are now more consistent than for the old ensemble with values differing by ± 0.02 at most, clearly showing

the influence of including fitted segmentations in the new ensemble.

Min-vote achieves highest observed precision and recall While the old ensemble was already capable of achieving impressive values for precision and recall at the extremes of $n = 1$ and $n = 4$, the new ensemble improves on this once more. Most notably, the new ensemble at $n = 4$ achieves a P_{B^3} of 0.83 for *pixels*, improving by 0.16 over the next best value among all evaluations, achieved by the old ensemble at $n = 4$. This is accomplished while also improving recall and thereby F_{B^3} and $F^*_{B^3}$. At worst, precision is unchanged for *nodes* at $n = 4$; the remaining precision values improve by 0.03 for $edges_{\text{fine}}$ and by 0.02 for the other atomic elements. The difference between precision for *pixels* and the edge-based atomic elements is also reduced, again showing the benefits of including fitted segmentations for MMDetection and the approach by Cormier et al..

Similarly, the new ensemble at $n = 1$ reaches unprecedented recall values between 0.93 for pixel-based atomic elements and 0.96 for *chars*. This is a marked improvement especially for *pixels*, $edges_{\text{fine}}$ and $edges_{\text{coarse}}$, where the new ensemble's recall is improved by 0.19 compared to the old ensemble.

However, the new ensemble is not an improvement over the old one in every single metric. For *pixels*, precision only improves over the old ensemble from $n = 3$. For $n = 2$ and $n = 3$, precision is lower by as much as 0.06 for *nodes* ($n = 2$ and $n = 3$), 0.05 for *chars*, 0.04 for $edges_{\text{fine}}$, 0.03 for $edges_{\text{coarse}}$, and 0.02 for *pixels* (all at $n = 2$). In the context of universal improvements in R_{B^3}, F_{B^3} and $F^*_{B^3}$, we find these to be minor losses.

Figures A.18 to A.21 show segmentations produced by the new ensemble on an example page. In comparison with the segmentations from the old ensemble (Figures A.14 to A.17), the differences become more apparent at $n = 3$ and $n = 4$, where the new ensemble produces far fewer segments and thus a less cluttered segmentation. The segmentation of the new ensemble at $n = 2$ is also visually significantly closer to the ground truth (shown in Figure A.3) than the corresponding segmentation produced by the old ensemble.

The observation made in Kiesel et al. [2019] that this "simple voting scheme can be employed to efficiently fuse the output of different algorithms" is strengthened by the improved results obtained through parameter analysis and post-processing performed in this work. The in some cases significant improvements of the ensemble with new parameters over the old parameter choices underscore the possible gains from well-chosen parameters, and the potential offered by combining purely visual segmentation algorithms with information from the DOM of a web page, a topic with large potential for future work.

Chapter 6

Conclusion

6.1 Insights

In this thesis, we conducted the first large-scale performance evaluation, comparison and analysis of five existing web page segmentation algorithms from the literature. For this purpose, we employed the *Webis Web Segments 2020* dataset, the largest publicly available dataset of human-annotated web page segmentations to date, created using a novel approach of crowd-sourcing segmentations. We utilized an evaluation methodology well-founded in clustering theory, improving significantly on previously used methods in terms of interpretability, comparability and reproducibility. Through parameter analysis and further experiments, we contribute a better understanding of the selected algorithms' operation and performance, and show the potential of future research into combining the benefits of different types of segmentation approaches.

To amend the shortcomings of prior quantitative web page segmentation algorithm evaluations, we performed a critical review of these previous works. We presented our criticism of these evaluations, which pertains to issues regarding dataset creation, a narrow evaluation context, imprecise evaluation metrics and incomplete information about the algorithm parameters that were used. To first provide a better understanding of the presented algorithms, we gave comprehensive overviews of their functional principles and detailed the implementations we used for our evaluations, in the case of the VIPS algorithm additionally contributing an implementation in TypeScript/JavaScript that is for the first time usable with a variety of rendering engines. We then elaborated on the clustering-theory based evaluation methodology and the nature of the *Webis Web Segments 2020* dataset and its creation presented in Kiesel et al. [2019], thereby showing how the previously identified shortcomings related to these points were improved on. In this context, we reflected on the choices of

atomic elements for the presented clustering similarity measure, performing an experiment to determine whether one choice most clearly matches the understanding of similarity of two human annotators. The results, however, show no clear winner, and we therefore evaluate the algorithms for all proposed atomic elements, which gives a more complete picture of algorithm performance for various purposes.

In total, we evaluated 18 distinct variations of algorithms and the resulting segmentations from further experiments against our ground truths. The results show the strength of the now 16 years old VIPS algorithm, provided its PDoC parameter is chosen appropriately. However, the purely visual approach by Cormier et al. comes a close second, showing that high-quality segmentations without any information provided by the rendering engine are achievable, again given proper parameter choices. We further find that approaches with a narrow focus (HEPS), as well as algorithms designed for other types of segmentation tasks, fall short in comparison.

To show the effects of combining information from the DOM of a web page with the segmentations created by the purely visual and hybrid approaches examined in this thesis, we performed an experiment evaluating segmentation quality when segments are fit to DOM nodes. The results show the potential benefits of such techniques, increasing the segmentation quality significantly under some types of atomic elements and, in the case of MMDetection, lifting the performance of the algorithm above the baseline in the first place.

Finally, we revisited the *Min-vote ensemble* approach for combining algorithm segmentations first proposed in Kiesel et al. [2019], comparing the combination of algorithms with unoptimized parameters against the fusion of segmentations with improved parameter choices and augmented results obtained from the fitting experiment. The new ensemble compares favorably to the old one, providing the best overall segmentation quality among all evaluations with Min-vote@2 and thereby solidifying the suitability of this voting scheme for an efficient combination of algorithm segmentations.

As a consequence of our results, our recommendations for high performance web page segmentation algorithms are:

VIPS with PDoC 5 if segmentation is performed within a browser on a live web page or within an environment utilizing a complete archive of the web page.

Cormier et al., $\mathbf{s}_{min} = 45$, $\mathbf{t}_l = 512$ px if (1) no complete archive of the web page is desired or available, or a purely visual segmentation is necessary for any other reasons, and (2) the high computational intensity is irrelevant for the application. **Fitting** should be employed if a DOM snapshot that corresponds to the rendered web page is available.

Min-vote@2 (new parameters) if the overall absolute best quality irrespective of the required computing power is desired and all necessary resources (complete web page archive) are available.

6.2 Future work

First and foremost, this thesis was only able to cover a small selection of existing algorithms in the field of web page segmentation; a logical continuation of this work would be the evaluation of more approaches on the *Webis Web Segments 2020* dataset using the same evaluation methodology, to further increase the available knowledge about existing algorithms for these tasks. This may necessitate the reimplementation of such algorithms independent of an underlying rendering engine, if applicable, which we find to be beneficial for web page segmentation research in general.

Furthermore, the interpretation of evaluation results for algorithm performance can be improved by regarding them in a more fine-grained fashion. We are aware of efforts to annotate the *Webis Web Archive 17*, the basis for the *Webis Web Segments 2020* dataset, with genres for each web page, also utilizing crowd-sourcing. Once these annotations are obtained, the web pages within a genre could be examined for prominent structural and design characteristics; a breakdown of algorithm evaluation results by genre may then reveal further insight into the performance of segmentation approaches for more specific types of web pages.

In Chapter 1, we mentioned the emergence of HTML5 semantic tags and ARIA tags for introducing a form of segmentation into the source code of web pages. We propose the evaluation of a naïve segmentation algorithm based thereupon to assess the suitability and performance of a web page segmentation incorporating these tags.

As mentioned in our overview of the method, our parameter analysis for the approach by Cormier et al. [2017] is incomplete and omits the t_p parameter; from our observations regarding t_l in Section 5.4, we find that this parameter likely presents a large influence on the resulting segmentation, and is therefore worth analyzing. Another parameter that we did not investigate further is θ_c, the containment threshold used for fitting segmentations to DOM nodes; we believe that the optimal value for this threshold varies from algorithm to algorithm, and that further improvements in segmentation quality are thereby possible.

As previously stated, the fitting experiment revealed that there is potential in combining the advantages of purely visual algorithms, especially ones tuned for web page segmentation such as that presented by Cormier et al.

[2017], with information from the DOM. Beyond using element bounding box information to fit segmentations, tuning algorithm parameters based on page characteristics extracted from the DOM is conceivable. The creation of an entirely new hybrid approach based on this concept could maximize the amount of information incorporated into the segmentation process, offering potential for significant quality increases. As a suitable next step in this direction, we see the combination of aspects from VIPS and the approach by Cormier et al.; a preliminary application of VIPS rules that cause coarse divisions on the page combined with focusing Cormier et al.'s approach to extended line detection on existing DOM element borders would serve to reduce the computational load of the edge-based visual approach while preserving the unique benefits of probabilistic segment border detection.

Appendix A

Appendix

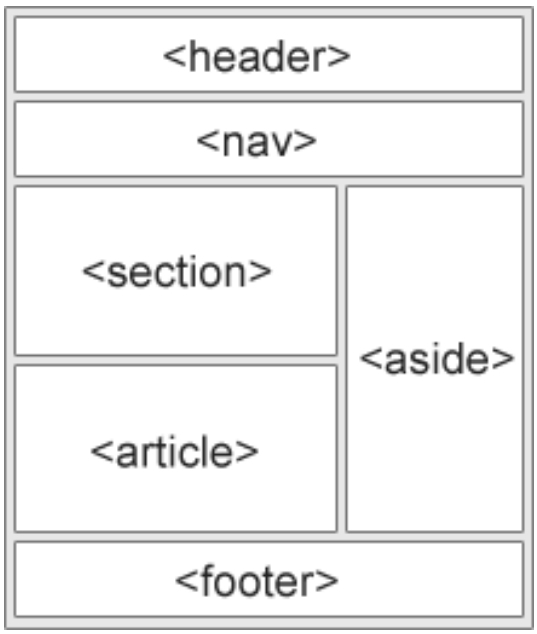


Figure A.1: An example of how using HTML5 Semantic Elements can provide information for segmenting a web page.
Source: `https://www.w3schools.com/html/img_sem_elements.gif`

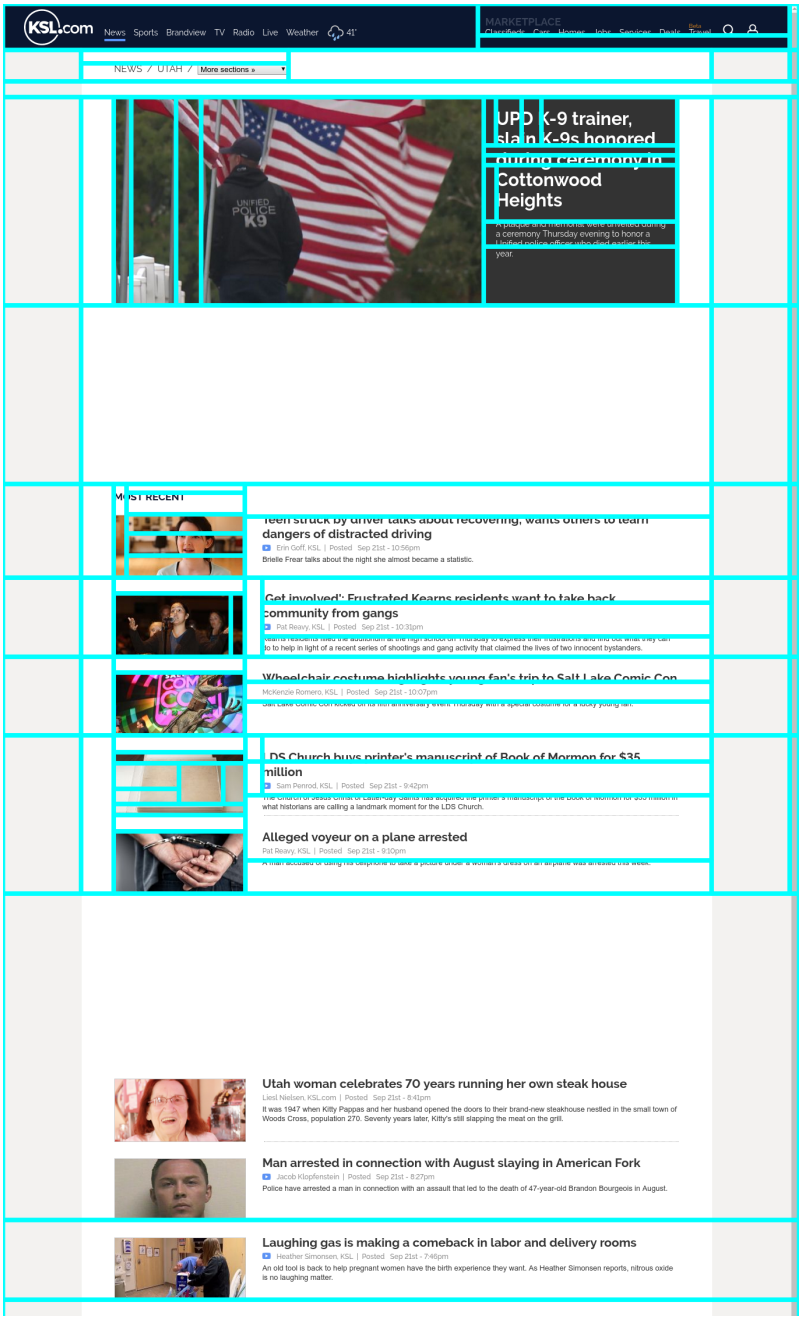


Figure A.2: Excerpt of a page segmented using the approach by Cormier et al., parameters $s_{min} = 45, t_l = 512$ px. This example shows that the edge detection often also incorporates edges within images on the page, such as the flag poles in the first picture. Additionally, lots of excess edges are detected along lines or vertical borders of text. The blank space following the first set of articles causes the significance probabilities for all vertical lines beyond this space to drop to 0 (see Section 5.4 for an explanation), and thus no further vertical segment borders are found for the remainder of the page.

Figure A.3: Example page showing segmentation bounding boxes for the ground truth

Figure A.4: Example page showing segmentation bounding boxes for VIPS with PDoC 5

Figure A.5: Example page showing segmentation bounding boxes for VIPS with PDoC 8

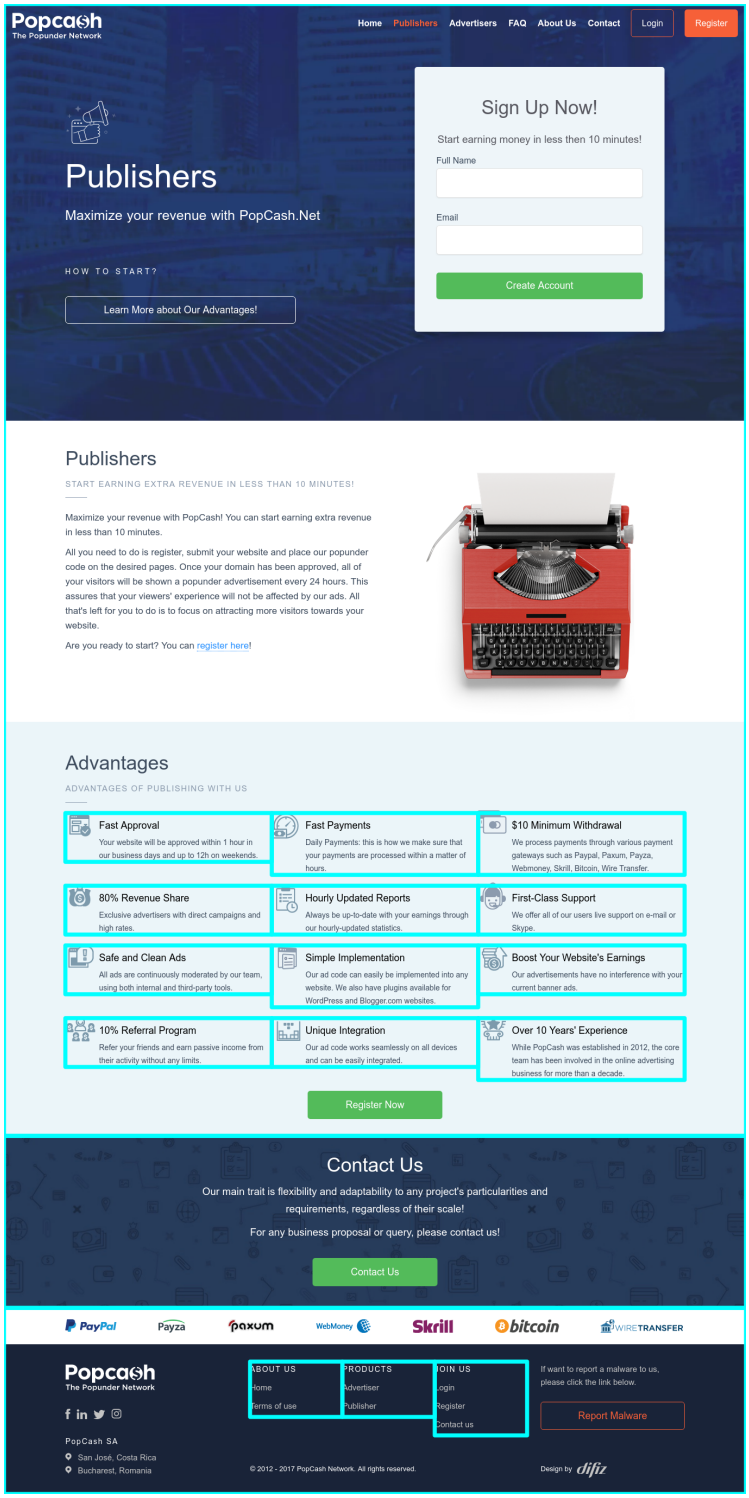


Figure A.6: Example page showing segmentation bounding boxes for HEPS

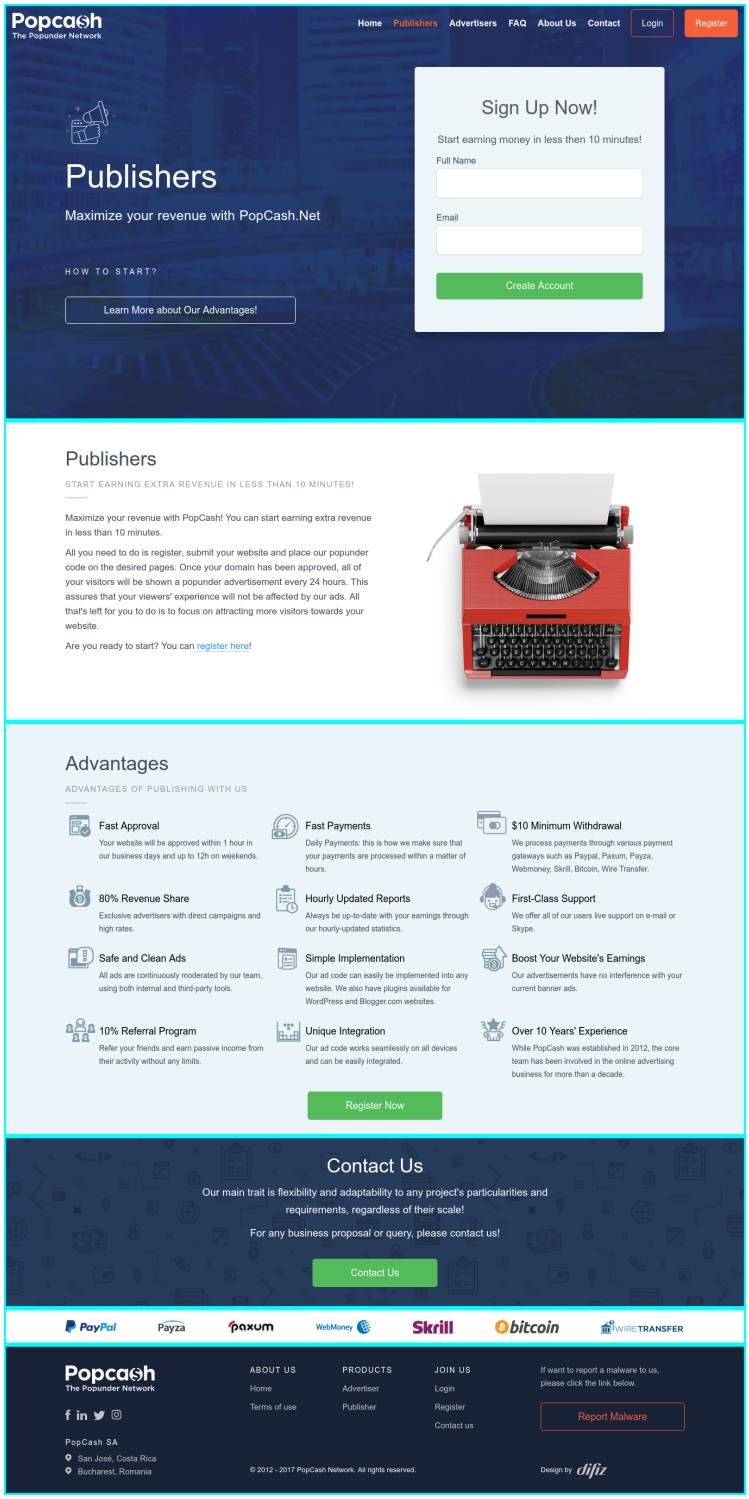


Figure A.7: Example page showing segmentation bounding boxes for the approach by Cormier et al. [2017], $s_{min} = 90, t_l = 512$ px

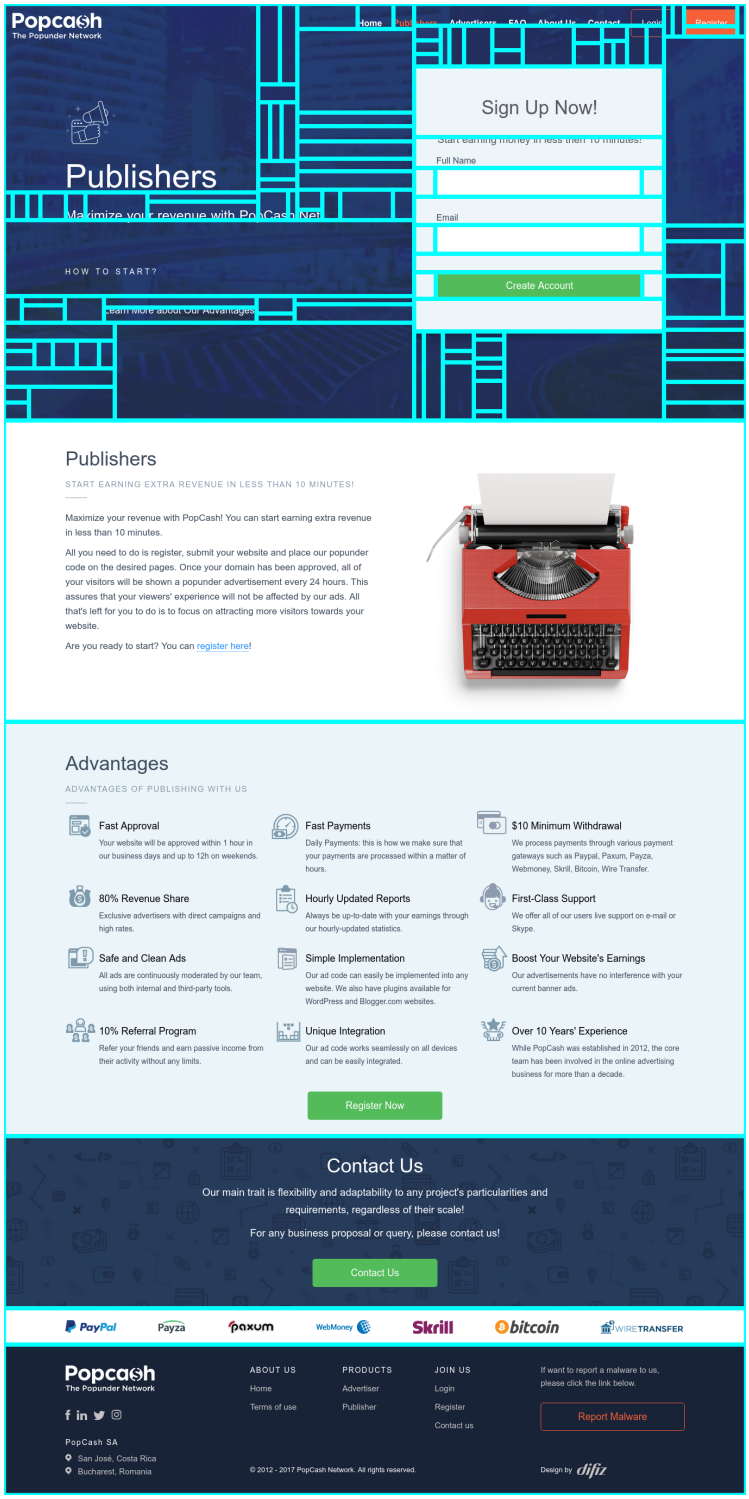


Figure A.8: Example page showing segmentation bounding boxes for the approach by Cormier et al. [2017], $s_{min} = 45, t_l = 512$ px

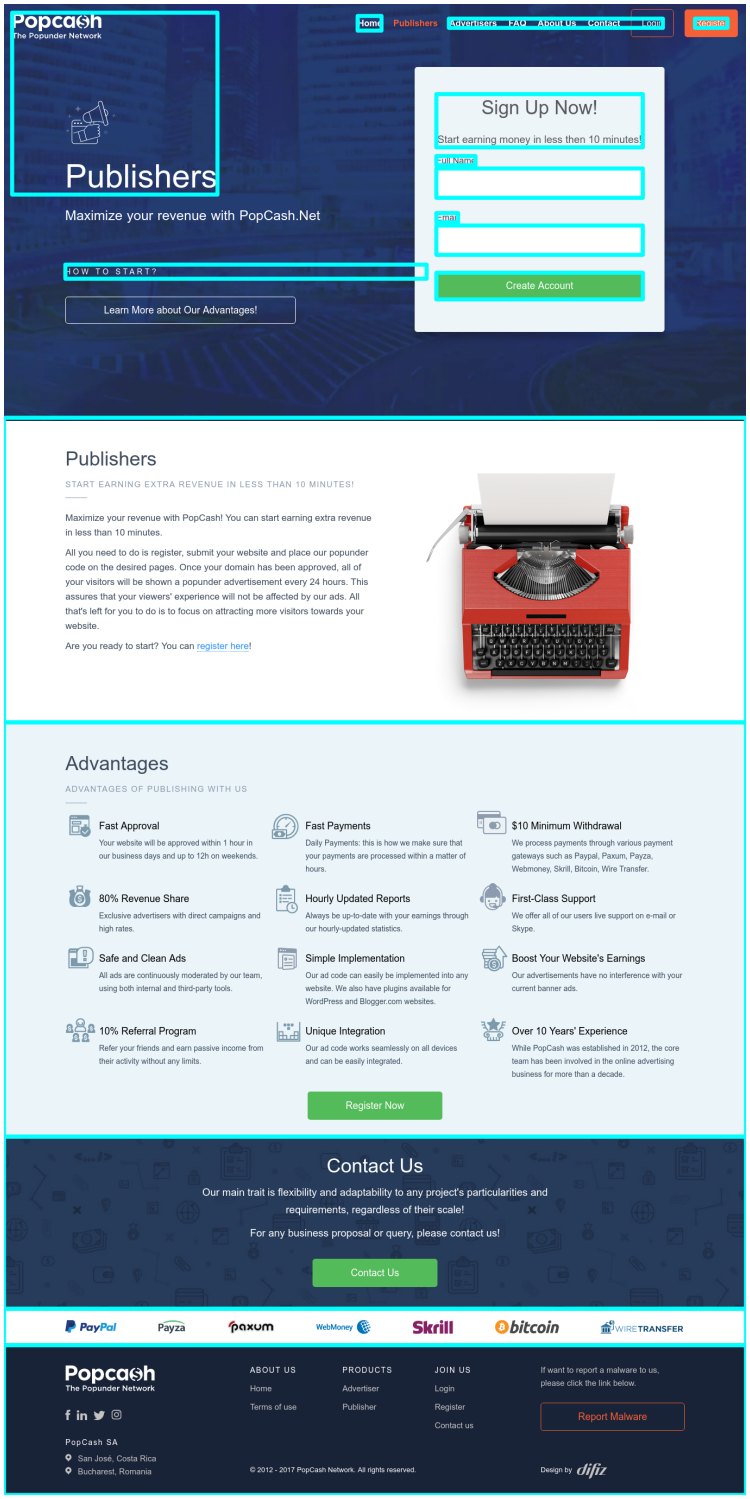


Figure A.9: Example page showing segmentation bounding boxes for the approach by Cormier et al. [2017], $s_{min} = 45, t_l = 512$ px, **fitted** with $\theta_c = 0.75$

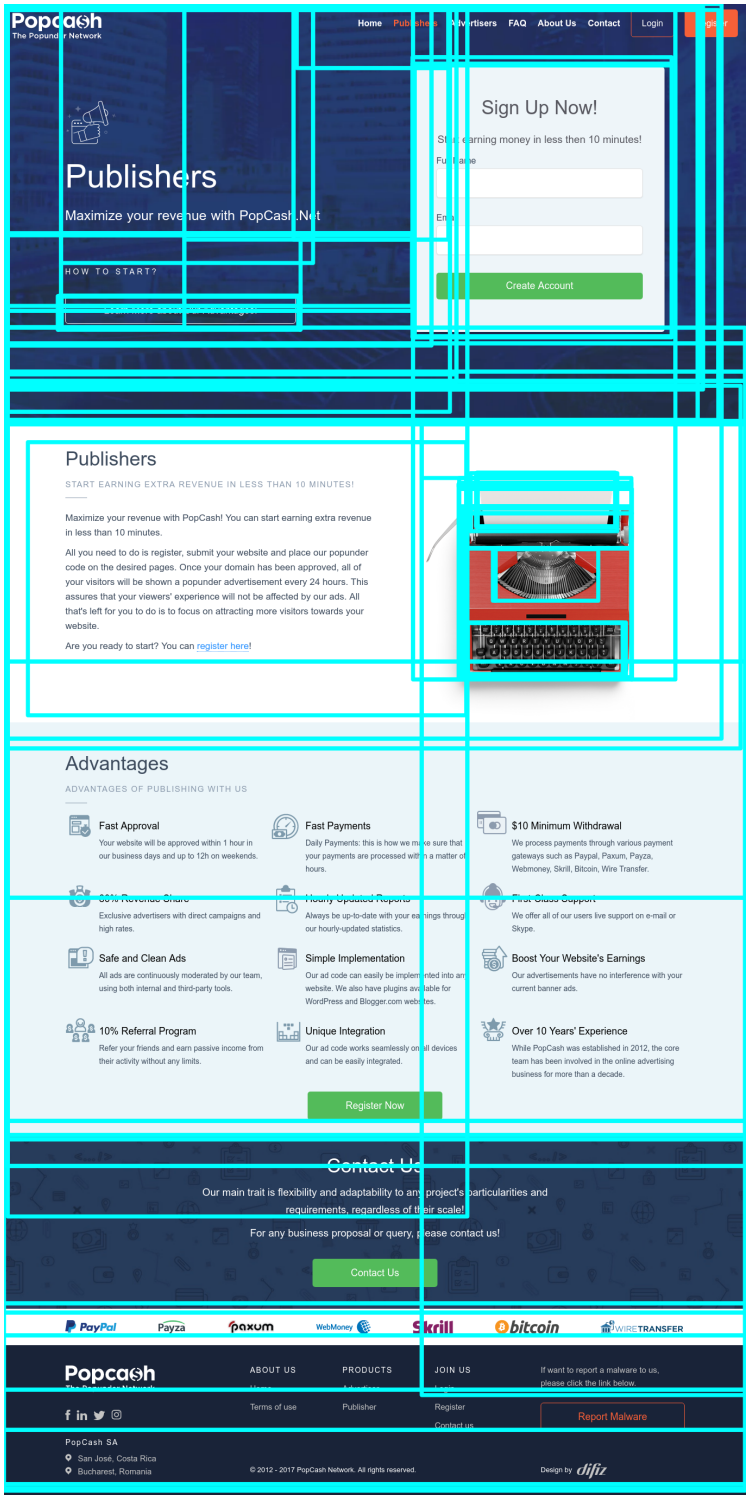


Figure A.10: Example page showing segmentation bounding boxes for MMDetection

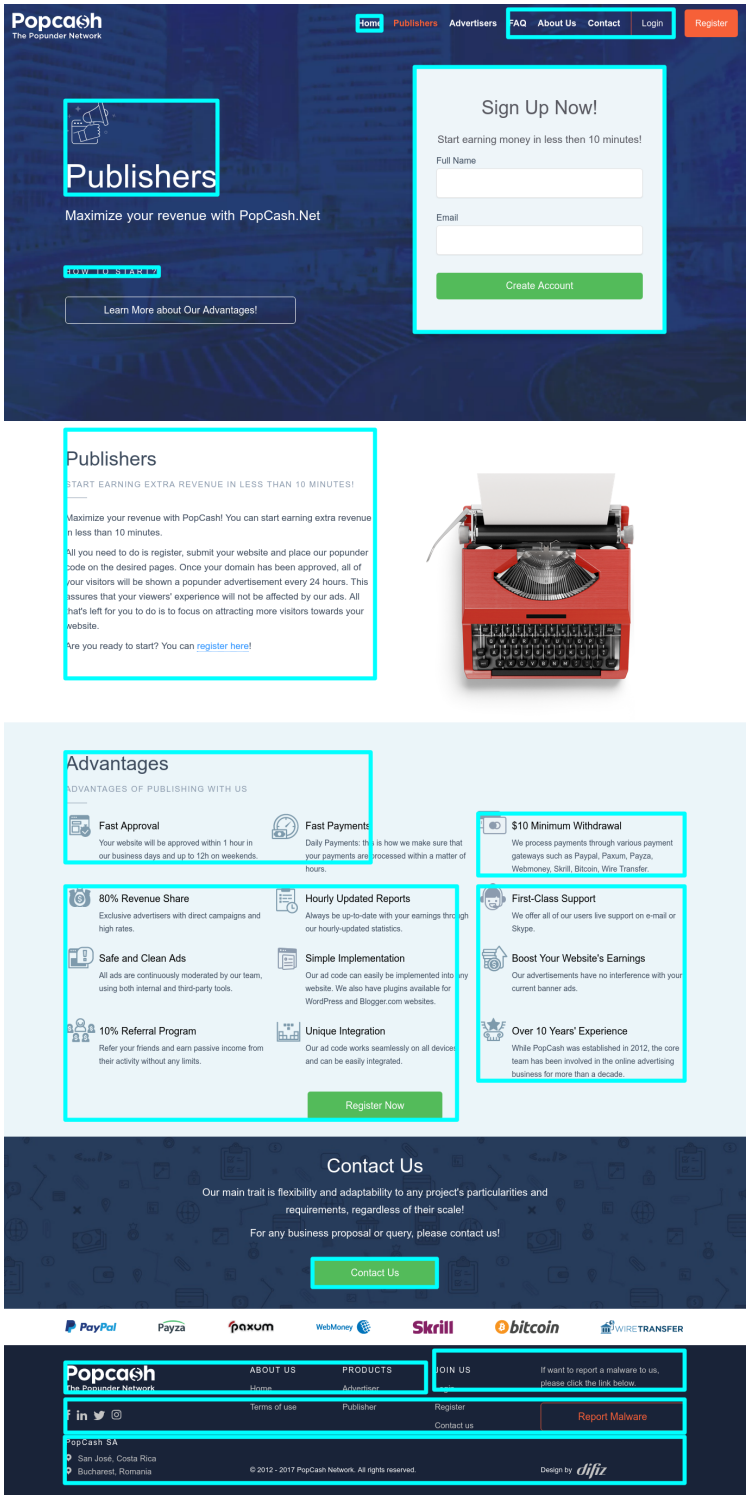


Figure A.11: Example page showing segmentation bounding boxes for MMDetection, **fitted** with $\theta_c = 0.75$

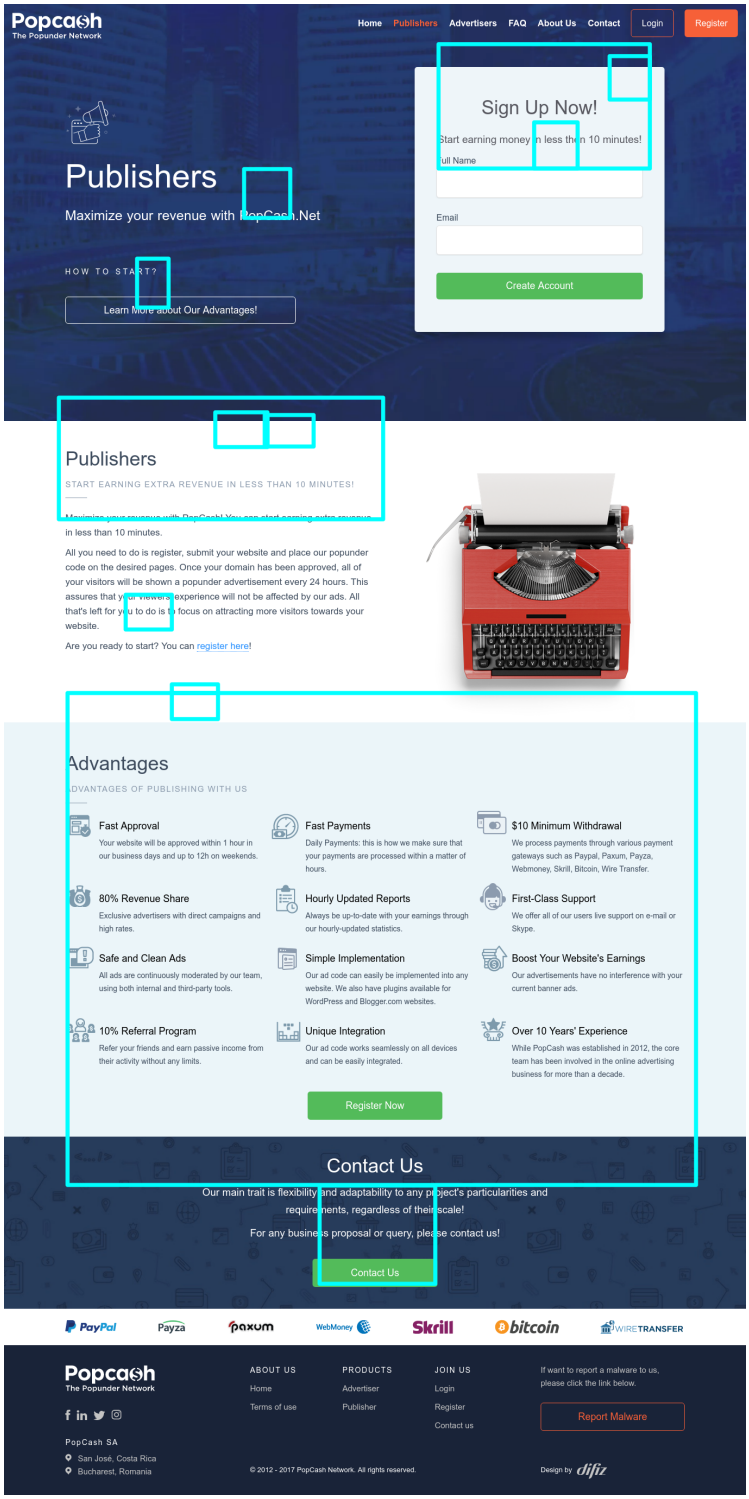


Figure A.12: Example page showing segmentation bounding boxes for the neural network by Meier et al. [2017]

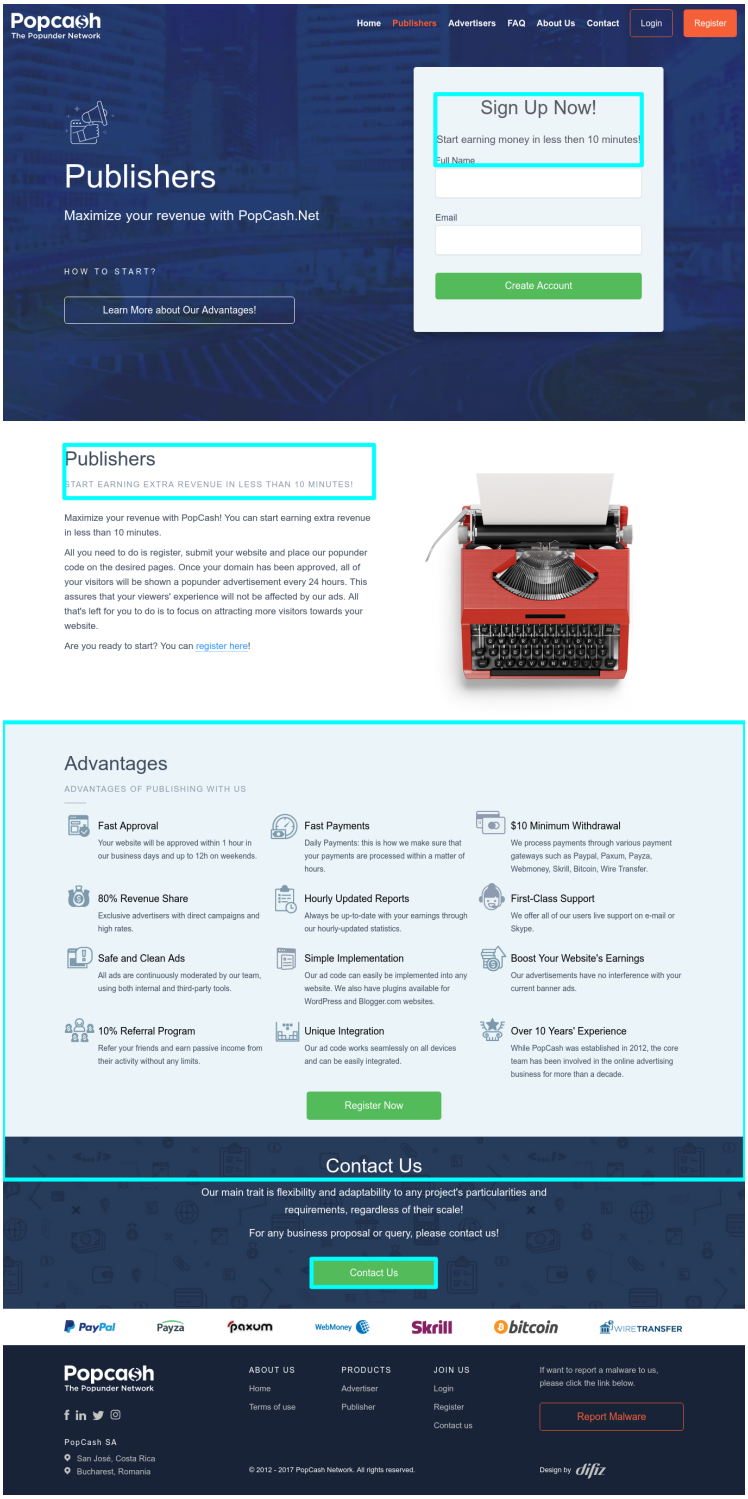


Figure A.13: Example page showing segmentation bounding boxes for the neural network by Meier et al. [2017], **fitted** with $\theta_c = 0.75$

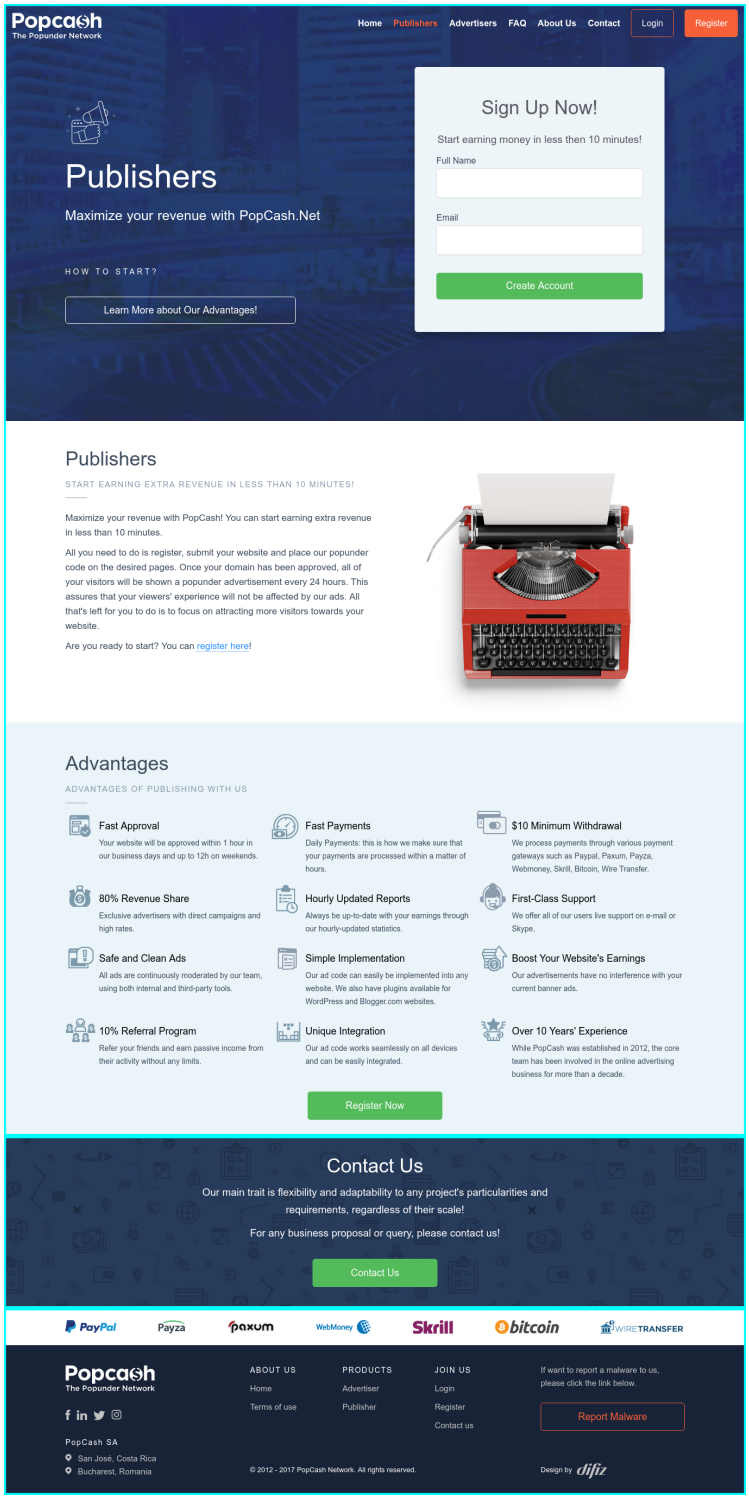


Figure A.14: Example page showing segmentation bounding boxes for Min-vote@1 with **old** parameters

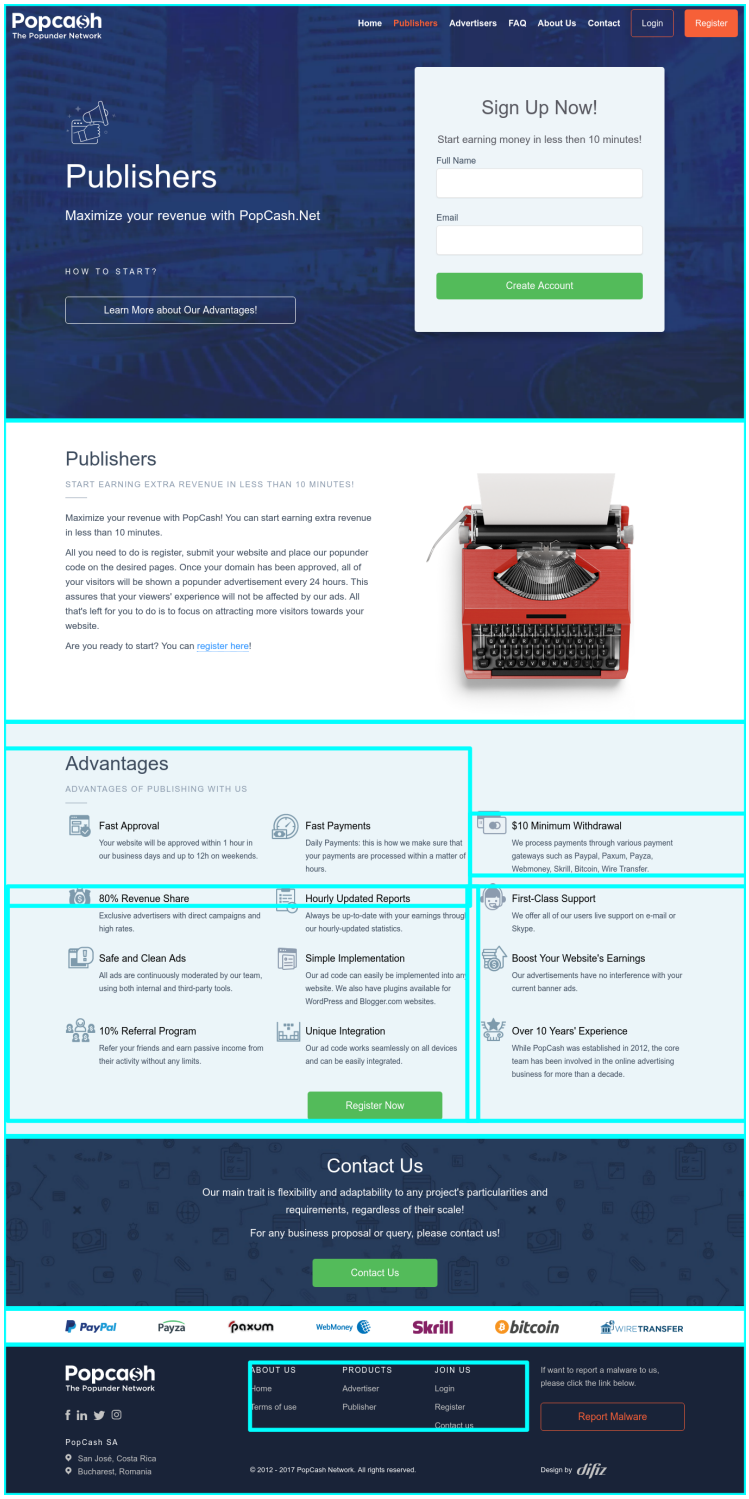


Figure A.15: Example page showing segmentation bounding boxes for Min-vote@2 with **old** parameters

Figure A.16: Example page showing segmentation bounding boxes for Min-vote@3 with **old** parameters

Figure A.17: Example page showing segmentation bounding boxes for Min-vote@4 with **old** parameters

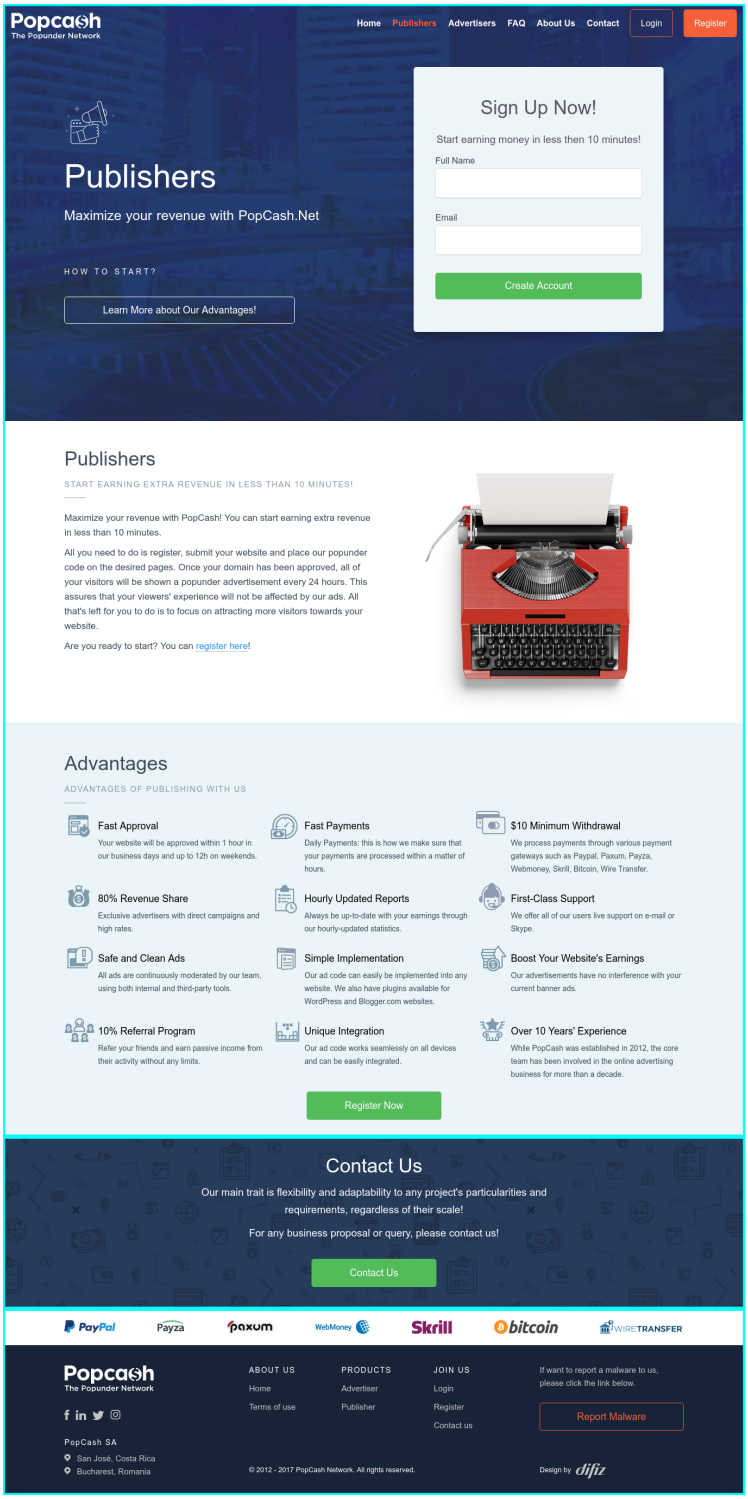


Figure A.18: Example page showing segmentation bounding boxes for Min-vote@1 with **new** parameters

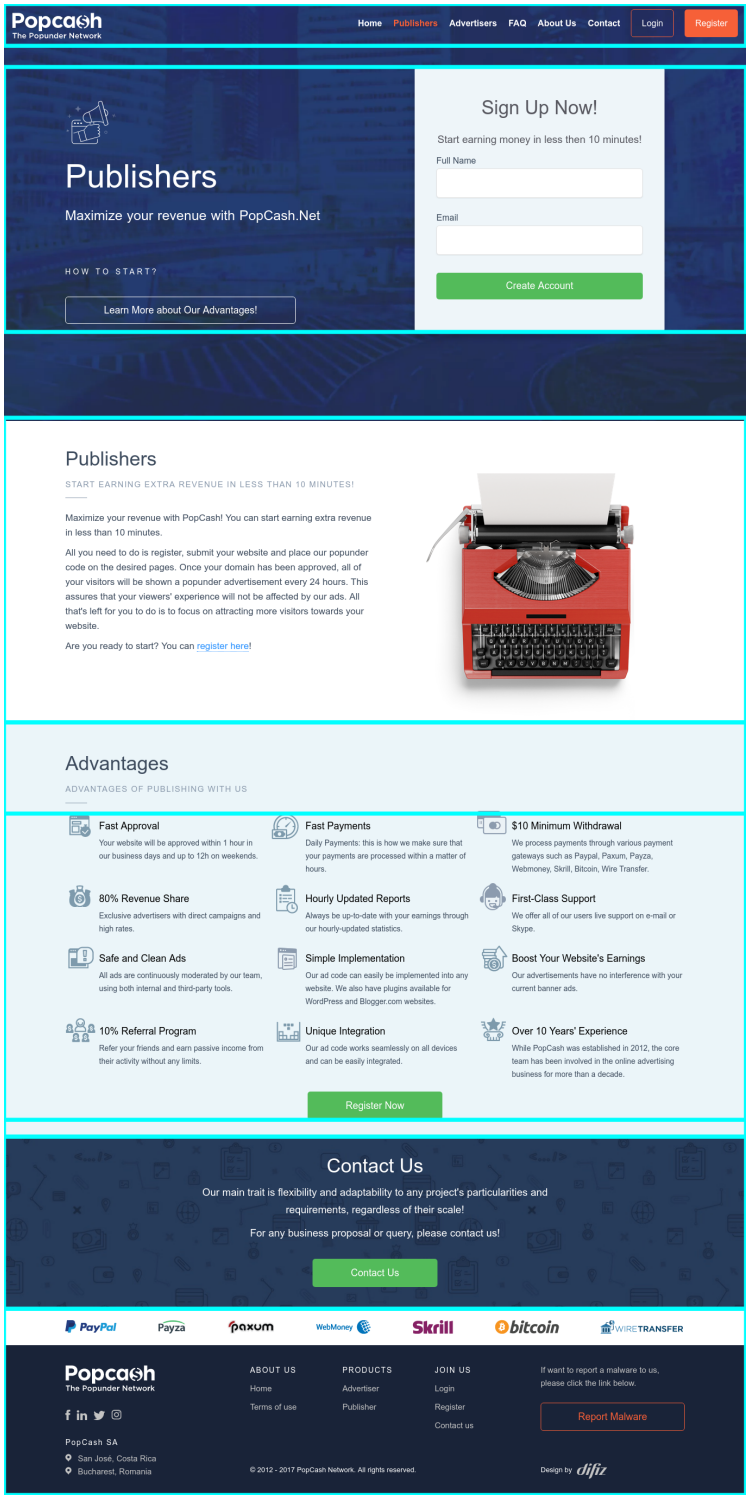


Figure A.19: Example page showing segmentation bounding boxes for Min-vote@2 with **new** parameters

Figure A.20: Example page showing segmentation bounding boxes for Min-vote@3 with **new** parameters

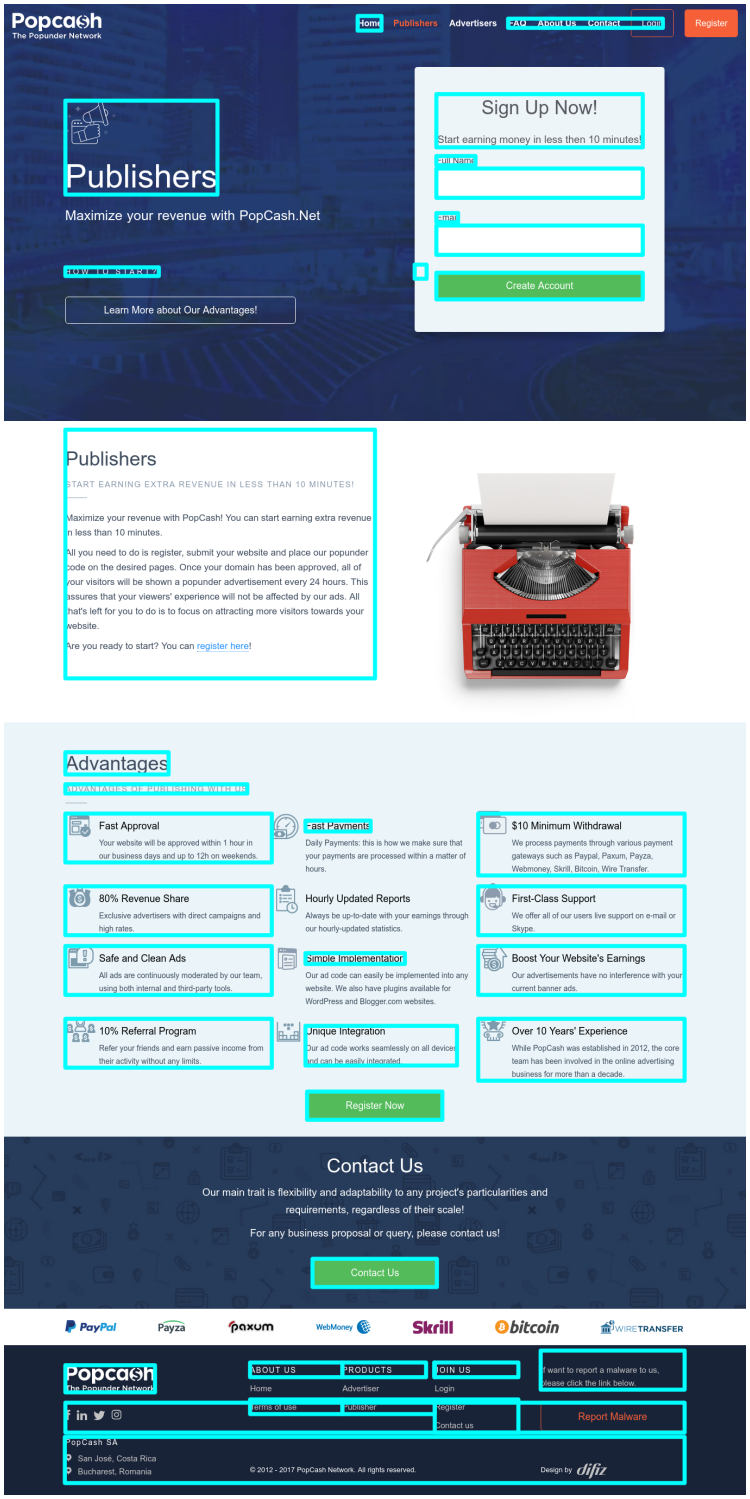


Figure A.21: Example page showing segmentation bounding boxes for Min-vote@4 with **new** parameters

Bibliography

M. Elgin Akpinar and Yeliz Yesilada. Vision based page segmentation algorithm: Extended and perceived success. In *Current Trends in Web Engineering - ICWE 2013 International Workshops ComposableWeb, QWE, MDWE, DMSSW, EMotions, CSE, SSN, and PhD Symposium, Aalborg, Denmark, July 8-12, 2013. Revised Selected Papers*, pages 238–252, 2013. doi: 10.1007/978-3-319-04244-2_22. 6

Enrique Amigó, Julio Gonzalo, Javier Artiles, and Felisa Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486, 2009. doi: 10.1007/s10791-008-9066-8. 33

Shumeet Baluja. Browsing on small screens: recasting web-page segmentation into an efficient machine learning framework. In *Proceedings of the 15th international conference on World Wide Web, WWW 2006, Edinburgh, Scotland, UK, May 23-26, 2006*, pages 33–42, 2006. doi: 10.1145/1135777.1135788. 6

Lidong Bing, Rui Guo, Wai Lam, Zheng-Yu Niu, and Haifeng Wang. Web page segmentation with structured prediction and its application in web page classification. In *The 37th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '14, Gold Coast , QLD, Australia - July 06 - 11, 2014*, pages 767–776, 2014. doi: 10.1145/2600428.2609630. 7

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 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*, pages 406–417, 2003a. doi: 10.1007/3-540-36901-5_42. 7, 8, 10, 16, 19, 32, 42, 51

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. VIPS: a vision-based page segmentation algorithm. Technical Report MSR-TR-2003-79, Microsoft Research, 2003b. 7, 8, 10, 12, 15, 16, 19, 32, 42, 51

John Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851. 25, 34

Jiuxin Cao, Bo Mao, and Junzhou Luo. A segmentation method for web page analysis using shrinking and dividing. *IJPEDS*, 25(2):93–104, 2010. doi: 10.1080/17445760802429585. 7

Deepayan Chakrabarti, Ravi Kumar, and Kunal Punera. A graph-theoretic approach to webpage segmentation. In *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, pages 377–386, 2008. doi: 10.1145/1367497.1367549. 7

Kai Chen, Jiaqi Wang an Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *CoRR*, abs/1906.07155, 2019a. URL `http://arxiv.org/abs/1906.07155`. 3, 19, 30, 59

Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4974–4983, 2019b. 31

Michael Cormier, Karyn Moffatt, Robin Cohen, and Richard Mann. Purely vision-based segmentation of web pages for assistive technology. *Computer Vision and Image Understanding*, 148:46–66, 2016. doi: 10.1016/j.cviu.2016.02.007. 14, 15, 16, 24, 29

Michael Cormier, Richard Mann, Karyn Moffatt, and Robin Cohen. Towards an improved vision-based web page segmentation algorithm. In *14th Conference on Computer and Robot Vision, CRV 2017*, pages 345–352, 2017. doi: 10.1109/CRV.2017.38. i, iii, 3, 6, 14, 15, 19, 24, 25, 41, 43, 44, 45, 46, 47, 49, 50, 55, 59, 60, 61, 62, 63, 64, 66, 67, 68, 70, 75, 76, 77

Steve Faulkner. ARIA in HTML. W3C working draft, W3C, December 2018. https://www.w3.org/TR/2018/WD-html-aria-20181214/. 4

Hanyang Feng, Wenzhe Zhang, He-Sheng Wu, and Chong-Jun Wang. Web page segmentation and its application for web information crawling. In *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI*, pages 598–605, 2016. doi: 10.1109/ICTAI.2016.0097. 7

David Fernandes, Edleno Silva de Moura, Altigran Soares da Silva, Berthier A. Ribeiro-Neto, and Edisson Braga Araújo. A site oriented method for segmenting web pages. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 215–224, 2011. doi: 10.1145/2009916.2009949. 7

Gen Hattori, Keiichiro Hoashi, Kazunori Matsumoto, and Fumiaki Sugaya. Robust web page segmentation for mobile terminal using content-distances and page layout information. In *Proceedings of the 16th International Conference on World Wide Web, WWW 2007, Banff, Alberta, Canada, May 8-12, 2007*, pages 361–370, 2007. doi: 10.1145/1242572.1242622. 6

Johannes Kiesel, Florian Kneist, Milad Alshomary, Benno Stein, Matthias Hagen, and Martin Potthast. Reproducible Web Corpora: Interactive Archiving with Automatic Quality Assessment. *Journal of Data and Information Quality (JDIQ)*, 10(4):17:1–17:25, October 2018. doi: 10.1145/3239574. URL `https://dl.acm.org/authorize?N676358`. 38

Johannes Kiesel, Martin Potthast, Benno Stein, Kristof Komlossy, Florian Kneist, and Lars Meyer. Web page segmentation from first principles. 2019. iii, 1, 4, 8, 9, 14, 16, 18, 19, 21, 32, 33, 38, 39, 41, 47, 62, 63, 64, 65, 66

Florian Kneist. Neural networks for web page segmentation. Master's thesis, 2019. iii, 28, 29

Christian Kohlschütter and Wolfgang Nejdl. A densitometric approach to web page segmentation. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 1173–1182, 2008. doi: 10.1145/1458082.1458237. 8, 10

Milos Kovacevic, Michelangelo Diligenti, Marco Gori, and Veljko M. Milutinovic. Recognition of common areas in a web page using visual information: a possible application in a page classification. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM)*, pages 250–257, 2002. doi: 10.1109/ICDM.2002.1183910. 7

Robert Kreuzer, Jurriaan Hage, and Ad Feelders. A quantitative comparison of semantic web page segmentation approaches. In *Engineering the Web in the Big Data Era - 15th International Conference, ICWE 2015, Rotterdam, The Netherlands, June 23-26, 2015, Proceedings*, pages 374–391, 2015. doi: 10.1007/978-3-319-19890-3_24. 3, 8, 9, 10, 11, 12, 33, 38, 39

K. S. Kuppusamy and G. Aghila. A model for web page usage mining based on segmentation. *CoRR*, abs/1202.2622, 2012a. 7

K. S. Kuppusamy and G. Aghila. Segmentation based approach to dynamic page construction from search engine results. *CoRR*, abs/1202.2617, 2012b. 7

Marc Teva Law, Carlos Sureda Gutierrez, Nicolas Thome, and Stéphane Gançarski. Structural and visual similarity learning for web page archiving. In *10th International Workshop on Content-Based Multimedia Indexing, CBMI 2012, Annecy, France, June 27-29, 2012*, pages 1–6, 2012. doi: 10.1109/CBMI.2012.6269849. 7

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference*, pages 740–755, 2014. doi: 10.1007/978-3-319-10602-1_48. `http://cocodataset.org/`. 30

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 27

Tomohiro Manabe and Keishi Tajima. Extracting logical hierarchical structure of HTML documents based on headings. *PVLDB*, 8(12):1606–1617, 2015. 7, 13, 19, 21, 32, 39, 44, 49

Benjamin Meier, Thilo Stadelmann, Jan Stampfli, Marek Arnold, and Mark Cieliebak. Fully convolutional neural networks for newspaper article segmentation. In *Document Analysis and Recognition (ICDAR), 2017 14th IAPR International Conference on*, volume 1, pages 414–419. IEEE, 2017. i, iii, 19, 27, 28, 30, 43, 46, 47, 58, 59, 60, 61, 62, 80, 81

Scott O'Hara, Patricia Aas, Shwetank Dixit, Bruce Lawson, Xiaoqian Wu, Terence Eden, and Sangwhan Moon. HTML 5.3. W3C working draft, W3C, October 2018. https://www.w3.org/TR/2018/WD-html53-20181018/. 4, 23

Chitra Pasupathi, Baskaran Ramachandran, and Sarukesi Karunakaran. Web document segmentation using frequent term sets for summarization. *Journal of Computer Science*, 8:2053–2061, 2012. doi: 10.3844/jcssp.2012.2053.2061. 7

Tomáš Popela. Implementace algoritmu pro vizuální segmentaci www stránek. Master's thesis, Brno University of Technology, Faculty of Information Technology, 2012. URL `https://www.fit.vut.cz/study/thesis/14163/`. iii, 9, 10, 21, 22, 44, 51, 52

Myriam Ben Saad and Stéphane Gançarski. Using visual pages analysis for optimizing web archiving. In *Proceedings of the 2010 EDBT/ICDT Workshops, Lausanne, Switzerland, March 22-26, 2010*, 2010. doi: 10.1145/1754239.1754287. 7

Andrés Sanoja and Stéphane Gançarski. Block-o-matic: a web page segmentation tool and its evaluation. In *29ème journées "Base de données avancées", BDA'13*, 2013. 10

Andrés Sanoja and Stéphane Gançarski. Web page segmentation evaluation. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13-17, 2015*, pages 753–760, 2015. doi: 10.1145/2695664.2695786. 8, 10, 12, 33, 35, 44

Andrés Sanoja and Stéphane Gançarski. Migrating web archives from HTML4 to HTML5: A block-based approach and its evaluation. In *Advances in Databases and Information Systems - 21st European Conference, ADBIS 2017, Nicosia, Cyprus, September 24-27, 2017, Proceedings*, pages 375–393, 2017. doi: 10.1007/978-3-319-66917-5_25. 6

Irwin Sobel and Gary Feldman. A 3x3 isotropic gradient operator for image processing. *Talk at the Stanford Artificial Project*, pages 271–272, 1968. 25

Srinivas Vadrevu, Fatih Gelgi, and Hasan Davulcu. Semantic partitioning of web pages. In *Web Information Systems Engineering - WISE 2005, 6th International Conference on Web Information Systems Engineering, New York, NY, USA, November 20-22, 2005, Proceedings*, pages 107–118, 2005. doi: 10.1007/11581062_9. 8, 9

Ou Wu, Yunfei Chen, Bing Li, and Weiming Hu. Evaluating the visual quality of web pages using a computational aesthetic approach. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 337–346. ACM, 2011. 7

Ou Wu, Weiming Hu, and Lei Shi. Measuring the visual complexities of web pages. *ACM Transactions on the Web (TWEB)*, 7(1):1, 2013. 7

Xin Yang and Yuanchun Shi. Learning web page block functions using roles of images. In *2008 Third International Conference on Pervasive Computing and Applications*, volume 1, pages 151–156. IEEE, 2008. 7

Xin Yang and Yuanchun Shi. Enhanced gestalt theory guided web page segmentation for mobile browsing. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 3, pages 46–49. IEEE, 2009. 6

Yelíz Yesilada. Web page segmentation: A review. Technical report, Middle East Technical University Northern Cyprus Campus, 2011. 7