World Wide Web (2025) 28:23
https://doi.org/10.1007/s11280-025-01333-3

A DOM-structural cohesion analysis approach for segmentation of modern web pages

Hieu Huynh[1] · Quoc-Tri Le[1] · Vu Nguyen[1,2,3] · Tien Nguyen[1,4]

Received: 9 May 2024 / Revised: 22 January 2025 / Accepted: 15 February 2025 /
Published online: 26 February 2025

Abstract
Web page segmentation is a fundamental technique applied in information retrieval systems to enhance Web crawling tasks and information extraction. Its objectives are to gain deep insights from crawling results and to extract the main content of a Web page by disregarding the irrelevant regions. Over time, several solutions have been proposed to address the segmentation problem using different approaches and learning strategies. Among these, the structural cue, which is a characteristic of the DOM tree, is widely utilized as a primary factor in segmentation models. In this paper, we propose a novel technique for Web page segmentation using DOM-structural cohesion analysis. Our approach involves generating blocks that represent groups of DOM subtrees with similar tag structures. By analyzing the cohesion within each generated block and comparing detailed information such as types, attributes, and visual cues of Web page elements, the approach can effectively maintain or reconstruct the segmentation layout. Additionally, we employ the Canny algorithm to optimize the segmentation result by reducing redundant spaces, resulting in a more accurate segmentation. We evaluate the effectiveness of our approach using a dataset of 1,969 Web pages. The approach achieves 64% on the F_{B^3} score, surpassing existing state-of-the-art methods. The proposed DOM-structural cohesion analysis has the potential to improve Web page segmentation and its various applications.

Keywords Web page segmentation · Web page analysis · Tree edit distance

✉ Vu Nguyen
vu.nguyen@katalon.com; nvu@fit.hcmus.edu.vn

Hieu Huynh
minhhieu2214@gmail.com

Quoc-Tri Le
lqtri691@gmail.com

Tien Nguyen
tien.n.nguyen@utdallas.edu

1 Katalon Inc., Atlanta, GA, USA

2 University of Science, Ho Chi Minh City, Vietnam

3 Vietnam National University, Ho Chi Minh City, Vietnam

4 University of Texas at Dallas, Richardson, TX, USA

1 Introduction

Web page segmentation is the process of dividing a Web page into smaller meaningful regions or segments such as the header, navigation menu, content area, sidebar, footer, and other elements. It is based on a Web page's content, structure, and visibility that humans can perceive.

Web page segmentation has been applied for various purposes, with a prominent application in information retrieval systems to distinguish valuable content from irrelevant content on a Web page. This segmentation process is utilized to enhance crawling (detecting templates [3, 30, 35], duplicates [23, 25], and changes [9, 20]) and for information extraction tasks (indexing [1], main content extraction [32, 36], and entity mining [21]). For example, instead of extracting only an image, we consider extracting the region that contains the image, allowing us to combine additional relevant information and gain more insights from the crawling result [5]. In the field of information extraction, segmentation can be used for extracting main contents by ignoring regions that contain noise, such as advertisements, or unrelated parts like headers, menus, and navigation bars [29]. Furthermore, in other areas, such as test automation, dividing Web pages into regions for page comparison plays a crucial role in generating reliable test oracles [33].

Over time, many solutions have been proposed to address the segmentation problem using different approaches and learning strategies. The most commonly used techniques fall into several categories: ad-hoc approaches [6, 7, 20, 27, 31] (which rely on manually-tuned heuristics and parameter-dependent methods), theoretically-founded approaches [1, 9] (based on graph-theoretic and classical clustering algorithms), computer vision approaches [11, 13], and others (as mentioned in [15]). In general, these approaches share three key elements: visual, textual, and structural cues found on Web pages. Among other features, structural is most widely utilized in various approaches. The structural cue involves approaches that take the DOM (Document Object Model) tree as an input. Due to various information comprised in the DOM tree, several well-known approaches have employed DOM as the main feature of their works, including algorithms such as the VIsion-based Page Segmentation (VIPS) [7], Block-o-Matic (BoM) [27], Jiang et al. [16], and Xiang et al. [31]. However, most of these algorithms rely on hand-crafted parameters [7, 27] or hard-configured rules [7].

These parameter-dependent or rule-based approaches have limitations in adaptability and robustness. They often require domain knowledge and fine-tuning, which is not fully automated and may not yield optimal results when applied on different Web pages. Taking into account these limitations, our approach aims to overcome the reliance on human-tuning parameters by employing flexible techniques that adapt to the characteristics of each website being analyzed.

In this research, a novel DOM-structural analysis using a statistical method is proposed. The proposed method involves the following steps: (1) generating blocks that are DOM subtrees with similar tag structures, (2) analyzing the cohesion in each block to keep the block in segmentation layout or reconstruct it, the detailed information in the Web page elements as type, attributes, and visual cues will be deeply compared to identify the block cohesion, and (3) applying the Canny algorithm [8] to reduce redundant spaces in the block for a more user-friendly layout.

We conduct a comprehensive assessment of our method including both quantitative and qualitative analyses. In the quantitative aspect, we evaluate our approach using a dataset comprising 1,969 Web pages sourced from the Webis-WebSeg-20 dataset [17]. Our results demonstrate a notable performance, with our method achieving a 64% F_{B^3} score, surpassing VIPS and MMDetection by 19% and 10%, respectively, on the same dataset. For qualitative

evaluation, our method outperformed the two compared methods, consistently receiving the highest rankings across the assessments conducted by five independent evaluators.

Our primary contributions to this paper include:

- Introduction of a novel DOM-structural analysis using a statistical method for Web Page Segmentation, this research bridges the gap created by human-tuning parameters in existing approaches.
- Execution of a comprehensive experiment covering both qualitative and quantitative aspects, wherein we compare our proposed approach to other state-of-the-art methods. The results demonstrate a significant improvement in the effectiveness of our approach.

2 Motivating example

Modern websites undergo frequent changes in their display, such as updating information and responsive user interface. This poses a significant challenge for Web page segmentation methods that rely heavily on visual characteristics. The following example shows a limitation of a vision-based method represented by VIPS.

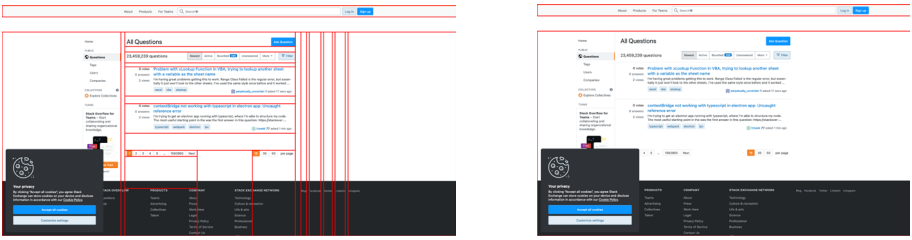
Figure 1 showcases two VIPS segmentation results obtained using different window sizes with the red rectangles representing segments. The segmentation outcome is influenced by two primary factors: the extraction of visual blocks and the identification of horizontal and vertical separators within these blocks. In our investigation, we reduced the window size of the Web page to observe a minor visual change. Surprisingly, the segmentation also changed, even though all other configurations remained the same. This limitation becomes apparent when performing information extraction tasks since the extracted contents are inconsistent, despite the Web page's content remaining unchanged.

Based on this experiment, we have concluded that visual characteristics alone are not sufficient to solve the Web page segmentation problem. Instead, we have shifted our focus toward DOM-structural analysis. Our approach integrates visual cues (text, visibility, etc.) for structural analysis under the DOM node's attributes, disregarding invisible elements on Web pages. By taking this approach, we aim to overcome the limitations observed with purely visual-based methods.

3 Method

Our approach to Web page segmentation relies on identifying regions or segments that have structurally similar elements and highlighting them with corresponding zones on a snapshot of the Web page. The output of our proposed method is a list of DOM nodes (DOM-subtrees) extracted from the Web page, referred to as a *DOM-driven layout*. This output is highly valuable for Web information retrieval as the DOM tree represents the complete information of a Web page, enabling us to extract all the relevant content. Similarly, each segment is represented by a DOM-subtree that contains all the necessary information that can be extracted from that specific segment. For result visualization, in some cases, the DOM node's visual block displays in an unexpected manner on the Web page's screenshot. For example, it does not cover all inside elements' content or contains redundant spaces (i.e., borders, paddings). To tackle this problem, we perform a further task to create an *appearance-driven layout* by employing the Canny algorithm Sect. 3.3.

The segmentation process begins by taking the URL of a Web page (W) and generating a DOM tree (D) through rendering and constructing. By identifying DOM nodes with similar



(a) PDoC = 6, window width = 1960, height = 993

(b) PDoC = 6, window width = 1600, height = 993

Figure 1 VIPS segmentation results with different window sizes

tag structures, we construct a modified DOM tree (D_b) that marks these nodes as blocks. To identify the desired regions within the *DOM-driven layout*, we analyze the cohesion or homogeneity of these blocks. Finally, we aim to optimize the geometry of each region on the Web page's screenshot, aligning it more closely with the human perspective in the *Appearance-driven layout* version.

3.1 Block generation

In this section, we provide a detailed explanation of the *block generation* process mentioned in Fig. 2. The objective of this step is to identify the nodes that have children with similar structural characteristics and mark these nodes in the DOM tree (D_b).

Definition 1 **(A DOM-block tree (D_b))**. is the original DOM tree where each node is marked as a block or not. Figure 3 shows an example of how block nodes are identified. We define the blocks to be a DOM node that contains children with a similar structure, which is illustrated as the red zone in Fig. 3. For employing the breadth-first search traversal, we first initialize a queue with the root node of the tree (e.g., body node). For every element in the queue, we compare its structure to the left and right siblings. If an element has no sibling or its structure does not match, the children of the current element will be added to the queue. Otherwise, the current element is marked as a block node as it contains a similar structure and remove from the queue. This loop will be executed until the queue is empty.

The structure matching we employ here uses only the skeleton structure that considers and includes only distinct elements. For instance, the full structure of Block 1 (Fig. 3) is `div[a[img, div, div], a[img, div, div]]`, and skeleton structure for this block is `div[a[img, div]]`, which excludes duplicated elements. Two structures match if they have the same skeleton structure.

In this stage, we aim to scan the whole tree to get all possible blocks. The output of this step is a *DOM-block tree*. It is possible that some blocks are either too large and contain other

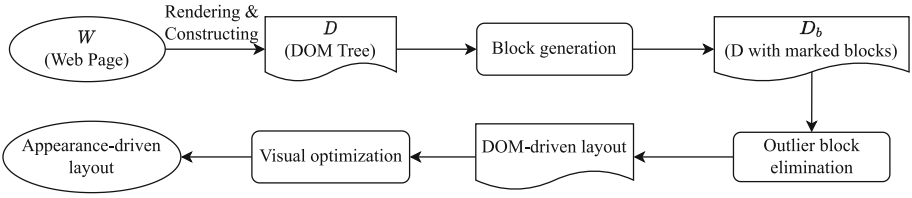


Figure 2 Segmentation model

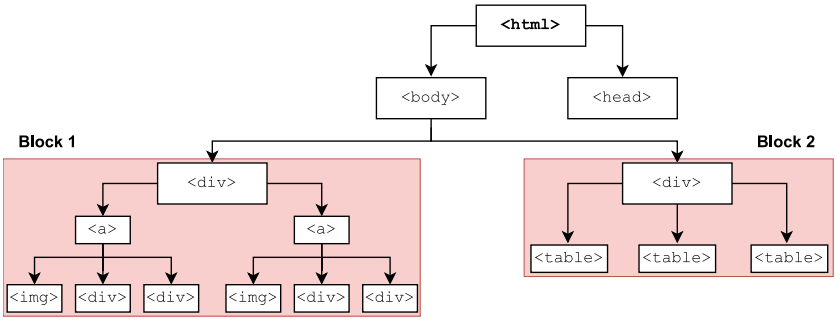


Figure 3 Example of block generation

blocks or too small. The blocks that are of little importance are known as *outlying blocks* and will be dealt with in the upcoming phase called *outlying block elimination.*

3.2 Outlying block elimination

As mentioned earlier, we propose a statistical method to obtain desired blocks from a hierarchical block layout. The method is based on the cohesion of the blocks determined during the previous generation process. A larger block may internally contain more distinct structures, resulting in low homogeneity (high heterogeneity) in structure. In such cases, the corresponding blocks should be removed and replaced by their child blocks. Assuming that we have identified blocks with high heterogeneity in structure (referred to as *outlying blocks*), Algorithm 1 demonstrates how we eliminate outliers based on the result of their heterogeneity analysis.

Definition 2 **(The segmentation blocks (S_{blocks})).** is the set of blocks obtained after filtering out the outliers. Before proceeding to the heterogeneity analysis process, we observe the

Algorithm 1 Outlying block elimination.

```
1  Function BlockEliminator(node):
2      S_blocks = ∅
3      if node.isOutlier then
4          for child in node.children do
5              if child.isBlockNode then
6                  S_blocks.insert(BlockEliminator(child))
7              else
8                  S_blocks.insert(child)
9              end
10         end
11     else
12         if node.isBlockNode then
13             S_blocks.insert(node)
14         end
15     end
16     return S_blocks
17 Blocks, Hs ← GetHeterogeneity(root)
18 MarkOutliers(Blocks, Hs)
19 S_blocks ← BlockEliminator(root)
```

Algorithm 2 Heterogeneity Analysis.

```
Function GetHeterogeneity(node):
    Blocks, Hs ← ∅
    if ! node or ! node.isBlockNode then
        return
    if ! node.children then
        Blocks.insert(node)
        Hs.insert(0)
    else
        H ← Heterogeneity (node.children)
        Blocks.insert(node)
        Hs.insert(H)
        for child in node.children do
            childAnalysis ← GetHeterogeneity(child)
            if childAnalysis is not None then
                Blocks.insert(childAnalysis[0])
                Hs.insert(childAnalysis[1])
            end
        end
    end
    return Blocks, Hs
```

hierarchical blocks represented as a flattened list of blocks. The objective of the analysis is to determine the heterogeneity score for each generated block.

Definition 3 **(The heterogeneity score** (H)**).** of a block is derived by evaluating the dissimilarity in the structure of its children. To determine this score, the Tree Edit Distance (TED) algorithm [26] is utilized to compare each pair of children, and the resulting metric is represented by the standard deviation of the edit distance cost for these pairs.

$$\begin{aligned} C_{i,j} &= TED(Child_i, Child_j), \forall i, j \in \mathbb{N}; i, j \leq N \\ H_{block} &= StandardDeviation(C) \end{aligned} \tag{1}$$

- N: Number of children or elements of a block node.
- C: A edit distance cost matrix, where each value in the matrix represents the cost of the best sequence of actions to transform $Child_i$ to $Child_j$.

The tree edit distance refers to the sequence of node edit operations with the lowest cost necessary to convert tree F into G. The typical edit operations include removing a node, adding a node, and changing the label of a node. In this paper, we employed the state-of-the-art TED algorithm called APTED, which has demonstrated its effectiveness not only in various Web research [4, 33, 34] but also in Android studies [28]. APTED allows us to achieve memory-efficient strategy computation, encompassing all paths without limitations, and ensuring optimal strategy computation within memory bounds of distance computation. Additionally, APTED facilitates fast computation for small subtrees, minimizing overhead associated with expensive single-path functions of RTED, thereby enhancing overall algorithm efficiency.

A higher H score for a block indicates a more distinct internal structure within the block, indicating an outlying block. The H score is computed using Algorithm 2.

Figure 4 shows a toy example of how H scores vary in different cases. The elements ○, □, △ belong to a block, and the edit distance cost between each pair of elements is different. In this case, B_1 contains only one type of element, resulting in the minimum H

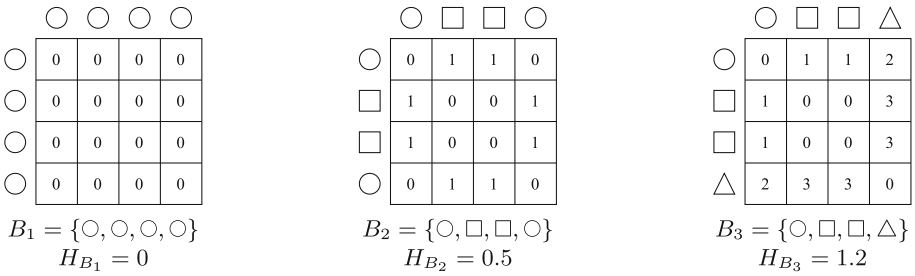


Figure 4 The heterogeneity score (H) reflects the dissimilarity in the structure within each block (B). Assuming, $TED(○, ○) = 0$, $TED(○, □) = 1$, $TED(○, △) = 2$, $TED(□, △) = 3$

score. Conversely, B_3 contains all types of assumed elements, leading to the maximum H score. In more complicated cases, the edit distance cost can be larger than in the example, depending on the differences between each pair of DOM subtrees within a block.

Based on the result of the heterogeneity analysis (Algorithm 2), the question arises: How can we determine the threshold to identify outlying blocks by H scores? To address this research question, we propose a statistical method that is free from predefined thresholds. This method utilizes the Interquartile Range (IQR) to assess the heterogeneity score of each block. By employing the IQR, we can effectively identify outliers within the generated blocks.

The identification of outliers is performed by the `MarkOutliers` function (Line 18, Algorithm 1). The `MarkOutliers` function takes two inputs: the set of block nodes (`blocks`) and their corresponding H scores (`Hs`). This function uses the Interquartile Range (IQR) method to identify outliers in H scores. Blocks with H scores exceeding the IQR threshold are flagged as outliers. Subsequently, the `BlockEliminator` function (Line 19, Algorithm 1) function traverses from the root nodes to obtain the nodes that are block nodes and are not flagged as outliers. This function returns the set of segmentation blocks S_{blocks}.

3.3 Visual optimization

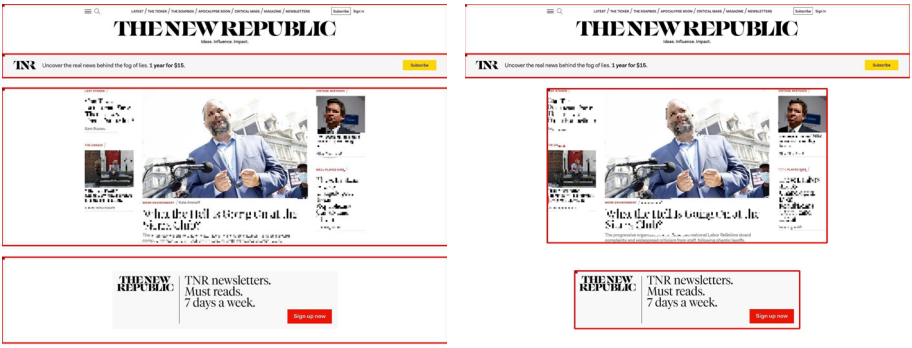
Definition 4 (A segmentation region (S_r)). is a segmentation block whose bounding box has been improved to be more suited for the screenshot of the Web page. The bounding box's empty spaces is removed, and the original bounding box is cropped to match the screenshot's representation of the region's content.

Figure 5a demonstrates that practically all segmentation blocks in the DOM-driven layout version have additional vacant spaces (i.e., borders, paddings) inside, which, in our opinion, lowers the segmentation layout's quality even further. We employ the Canny algorithm [8] to optimize bounding boxes by reducing the redundant space. The bounding box is shrinked down until it reaches edges detected by the Canny algorithm. The optimized segmentation layout is more comprehensible and reduces redundant zones in each section, as shown in Fig. 5b.

4 Quantitative evaluation

4.1 Dataset

We conduct our evaluation on the Webis-WebSeg-20 [17] dataset, which is the largest publicly accessible set of data in terms of Web segmentation. The dataset contains more than 8,000



(a) DOM-driven layout (b) Appearance-driven layout

Figure 5 Example of a visually optimized layout segmentation

Web pages from more than 5,500 different domains. The websites are well distributed from top-ranked to low-ranked websites by Alexa ranking to ensure the diversity in data.

For each data instance, the authors provided an HTML file to store the DOM structure of the Web page and a screenshot of the whole page. Theoretically, the position of every DOM node in the HTML file in the dataset is stored in another metadata file along with the respective absolute XPath. This is to make sure that we do not need to re-render the Web page and guarantee that the position of every node matches the corresponding position in the ground truth.

However, while reviewing the data, we realized that the XPaths in the metadata were constructed incorrectly. As they only count the index of visible siblings while ignoring the invisible ones. Therefore, in an HTML file, if there is a node that is invisible, the following sibling nodes would be mis-indexed. When reconstructing the XPath of nodes in the HTML file without rendering, we are unable to identify whether an HTML element is visible. As a result, it is impossible to map the nodes to the corresponding XPath in the dataset. To resolve this problem, we decided to filter out the data instances that are indexed XPath incorrectly.

Algorithm to filter the data Let X represent the set of provided XPaths, and X^* denote the set of actual XPaths. We define P as the set of tuples $(x,\ p)$ where x belongs to X and p is the rendered position of x on the page, $P = \{(x,\ p) \mid x \in X\}$. Our objective is to create a set $M = \{(x,\ x^*)\}$, which represents the one-to-one mapping between XPaths in the provided set X and their corresponding XPaths in the actual set X^*. Each element in M is a tuple $(x,\ x^*)$, where x is an XPath from X, and x^* is the corresponding XPath from X^*. For each x in X, we aim to map it to the least greater-indexed x^* in X^* that possesses the same skeleton. As the index of x does not count the invisible nodes, it is always smaller than or equal to the actual index. Once all mappings are established, we check for any duplicated x^* values across all mappings. If duplicates are found, we eliminate the corresponding data instance as it fails to comply with the rules.

As a result, there are 1,969 data instances that are indexed correctly, and we use this set to evaluate our approach.

4.2 Metrics

We adopted a comprehensive framework for assessing the performance of Web page segmentation algorithms [17]. This approach allows us to enhance the comparability and consistency

of our method's performance with other evaluated models that have been tested on the same dataset.

Unlike other image segmentation problems, Web segmentation might produce a tree-structured segmentation result with different granularity levels that make a segment bigger or smaller. As a result, a ground truth segment may contain several predicted segments and vice versa. Therefore, this evaluation framework considers Web page segmentation as a clustering task where each Web page element is clustered into a group.

In the context of clustering, defining a similarity function is essential as it allows us to quantify the distance or dissimilarity between elements within a cluster. In such a manner, Kiesel et al. [17] have introduced the *atomic element* concept; they are (1) pixels, (2) DOM nodes, and (3) characters. The authors adapt the extended BCubed measurement from clustering, which examines the P_{B^3} (2), R_{B^3} (3), and F_{B^3} (4) for each data instance [17].

To recapitulate, in the same manner of common precision, P_{B^3} calculates the ratio of true positive segments (segments that are in both ground truth and predicted segmentation) to all of the positive segments (segments that are in the predicted segmentation). Likewise, R_{B^3} calculates the ratio of true positive segments to all of the positive segments in the ground truth. And F_{B^3} is the harmonic mean of P_{B^3} and R_{B^3}. F_{B^3} satisfies all the necessary criteria for a segmentation similarity measure. It effectively handles partial segmentations, overlapping segments, and even nested segments. Additionally, F_{B^3} is resilient against trivial segmentations, such as under-segmentation, which divides the whole page as a segment, or over-segmentation, which consider every atomic element as a segment.

$$P_{B^3}(S, S^*) = \frac{1}{|E^S|} \sum_{e \in E^S} \left(\frac{1}{|E_e^S|} \sum_{e' \in E^S} \frac{\min(|S_e \cap S_{e'}|, |S_e \cap S_{e'}^*|)}{|S_e \cap S_{e'}|} \right) \tag{2}$$

$$R_{B^3}\left(S,\ S^*\right) = P_{B^3}\left(S^*,\ S\right) \tag{3}$$

$$F_{B^3}\left(S,\ S^*\right) = \frac{2 \times R_{B^3}\left(S,\ S^*\right) \times P_{B^3}\left(S,\ S^*\right)}{R_{B^3}\left(S,\ S^*\right) + P_{B^3}\left(S,\ S^*\right)} \tag{4}$$

Let $S,\ S^*$ be the set of prediction segments and ground truth segments respectively, S_e be the set of segments that contains element e, E_S be the set of elements that belong to at least one segment in S. In the mathematical terms, $S_e = \{s \mid s \in S \wedge e \in s\}$, $E_S = \{e \mid e \in E \wedge S_e \neq \emptyset\}$, and $E_e^S = \{e' \mid e' \in E \wedge S_e \cap S_{e'} \neq \emptyset\}$. The set of elements e is denoted as E. At the *pixels* atomic level, each pixel in the page's screenshot is represented by e. There are also two subsets of *pixels* known as *edges-fine* and *edges-coarse*, which utilize the edge image generated by the Canny algorithm [8]. Regarding *nodes*, E pertains to the collection of DOM nodes within the tree structure. As for *chars*, it represents the set of characters shown on the page.

4.3 Experimental setup

We use the evaluation results from [17] to compare with our algorithm. The experiment and parameter setup for each algorithm are summarized as follows.

- **Baseline**: To provide context, the performance of the algorithms is compared to a naive approach of segmenting a Web page into a single segment, which always reaches a maximum recall of 1 but the lowest precision.
- **VIPS**: VIPS [7] creates a hierarchical tree of segments based on the DOM tree of the page. Segments are split based on their degree of coherence, determined by heuristic

rules considering tag names, background colors, DOM node sizes, and visual separators. The single parameter that influences the algorithm is the permitted degree of coherence (PDoC). The higher PDoC is, the lower precision it be. After experimentation, it was determined that the optimal value for PDoC is 6.

- **HEPS**: Kiesel et al. [18] adapt HEPS [22] to the task of Web page segmentation. HEPS employs heuristic rules considering DOM tree positions, tag names, font sizes, and font weights to identify headings and their corresponding segments. Kiesel et al. [18] then merge the extracted headings with segments and extract the bounding box coordinates instead of text segments.
- **Cormer**: Cormer et al. [13] developed a visual algorithm for Web page segmentation based on edge detection. It calculates the probability of significant edges and composes line segments. The algorithm recursively splits segments based on the most semantically significant lines. It depends on two parameters which are tl (maximum line length) and smin (minimum segment size); the best setting (tl = 512, smin = 45) was used for further experiments.
- **Meier**: Meier et al.'s [24] convolutional neural network is cutting-edge in segmenting digitized newspaper pages. Kiesel et al. [18] re-implemented it using text node positions instead of OCR. Uniform height (4096 pixels) and scaling (256x768 pixels) were applied. The training utilized 10-fold cross-validation, stopping after 20.8 epochs on average.
- **MMDetection**: MMDetection [12] is a widely-used computer vision tool originally designed for image segmentation. To adapt it for Web page segmentation, we apply additional post-processing step by mapping each pixel mask to the nearest DOM node. This adaptation is necessary because the pixel masks produced by standard image segmentation models are not directly compatible with the structure of Web pages. In cases where MMDetection fails to detect any segments, the entire page is treated as a single segment to ensure coverage.
- **Segment Anything (SAM)**: Meta's Segment Anything Model [19] is a zero-shot image segmentation algorithm. Trained on a dataset comprising over 11 million images, SAM exhibits generalization capabilities, allowing it to perform zero-shot segmentation across diverse and challenging domains without any additional training. Its effectiveness is demonstrated across various complex datasets, including underwater imagery and medical images such as X-rays. However, SAM has yet to be evaluated using images of Web pages. In this study, we aim to assess SAM's performance in WPS and benchmark it against other state-of-the-art methods. Since both MMDetection and SAM are computer vision-based approaches that generate pixel masks, we apply the same post-processing step (mapping pixel masks into DOM nodes) for SAM to ensure a fair comparison.

For our experiments, we used an NVIDIA T40 GPU. We employed the same configuration and ViT-H checkpoint used in the original SAM paper, ensuring consistency with their experimental setup.

4.4 Results

Table 1 shows the evaluation results of our proposed method and seven other algorithms including the baseline, VIPS, HEPS, Cormer, Meier, MMDetection, and SAM. Since the size of the dataset was reduced to nearly 1/4 as compared to the original one, we also filtered the evaluation data on those selected ones and recalculated the R_{B^3}, P_{B^3}, F_{B^3}, and $F^*_{B^3}$ for each atomic element type. Notably, we observed that the results for each metric exhibited

Table 1 The results of 8 algorithms include 6 algorithms that we reuse the results from Kiesel et al. [18], SAM and ours

Metric		Baseline	VIPS	HEPS	Corm.	MMD.	Meier	SAM	Ours
Pixels	P_{B^3}	15	33	31	31	48	47	48	**60**
	R_{B^3}	100	69	62	**89**	59	55	64	68
	F_{B^3}	23	37	31	32	41	33	44	**51**
	$F^*_{B^3}$	26	45	42	46	53	50	55	**64**
Chars	P_{B^3}	44	81	72	56	80	61	**85**	83
	R_{B^3}	100	73	56	**91**	60	66	63	71
	F_{B^3}	57	**70**	53	61	61	53	67	68
	$F^*_{B^3}$	61	**77**	63	70	68	63	72	76
Nodes	P_{B^3}	32	72	61	47	77	53	**79**	75
	R_{B^3}	100	72	52	**90**	52	63	49	67
	F_{B^3}	45	**66**	46	52	54	46	54	62
	$F^*_{B^3}$	49	**72**	56	62	62	57	60	71
Edges-fine	P_{B^3}	33	67	58	48	73	55	74	**75**
	R_{B^3}	100	71	61	**89**	55	59	56	70
	F_{B^3}	46	61	49	51	55	43	57	**63**
	$F^*_{B^3}$	50	69	59	62	63	57	64	**72**
Edges-coarse	P_{B^3}	33	68	58	49	73	54	75	**75**
	R_{B^3}	100	71	61	**89**	55	59	57	69
	F_{B^3}	46	62	49	52	55	43	58	**63**
	$F^*_{B^3}$	50	70	60	63	63	57	65	**72**

For each kind of atomic element, we provide four metrics, including F-bcubed (F_{B^3}), Precision (P_{B^3}), Recall (R_{B^3}), and the harmonic mean of the averaged P_{B^3} and R_{B^3}. The bold values show the highest values among all, excluding the baseline

only a slight variation, with changes of less than 5% for each of them, while the overall score distribution remained unchanged.

The baseline method provides the most trivial segmentation, which considers a whole Web page as a single segment that covers all of the ground truth segments. Thus, it reaches the maximum R_{B^3} of 1 across 4 types of atomic. Obviously, the P_{B^3} achieves a fairly low score, resulting in the lowest F_{B^3} and $F^*_{B^3}$ scores.

Considering the P_{B^3} score, our algorithm achieves the highest values for every atomic type except for *nodes* and *chars*. The *chars* category attains the highest P_{B^3} score among all categories, reaching 83%. Regarding the F_{B^3} and $F^*_{B^3}$ scores, our approach demonstrates significant improvements, particularly for *pixels*, where our algorithm surpasses MMDetection by 11% and SAM by 9%. SAM and MMDetection are two pixel-centric algorithms that process images as input and produce segmentation in the form of pixel masks. This underscores our algorithm's superiority over computer vision-based approaches in the pixel-level segmentation.

The limitations of computer vision-based approaches become apparent when processing Web pages with complex backgrounds or pop-up advertisements. The multi-layered nature of Web content results in information loss when converted into 2D images, affecting segmenta-

tion accuracy. For instance, if a Web page contains images, vision-based approaches segment these images, which is wrong because each image is itself a Web element and must not be segmented. To address this, we implemented a post-processing step to align segmentation masks with the nearest DOM nodes, thereby discarding segments not represented by any DOM node. Without this matching step, the *pixel* $F^*_{B^3}$ score significantly drops to just 37% for SAM.

In the character-level segmentation, SAM achieves relatively high precision, but its recall is rather low. A close examination of the dataset revealed that in Web instances dominated by text elements, SAM tends to struggle, often segmenting headers and titles and ignoring body text due to the lack of significant differentiation. In contrast, our method effectively segments these elements by grouping them based on tag names.

Despite performing relatively well on character and node elements, VIPS exposes its shortcomings in the *pixels* category. This disadvantage can be explained by the fact that VIPS tends to leave unnecessary background regions in their segmentation.

Figure 6 illustrates the F_{B^3} distribution of our algorithm and the other seven. The F_{B^3} median value obtained by our algorithm is higher than those by all other algorithms in every atomic type except for 'nodes.' Additionally, our upper quartiles for all five atomic levels reach the highest above all, showing that our algorithm generally produces higher accuracy.

Kiesel et al. [18] have pointed out a gap in previous works is the low F_{B^3} score on the *pixels* atomic level, where our algorithm makes a significant improvement and remains a competitive F_{B^3} and $F^*_{B^3}$ in other categories. The accompanying figure illustrates the remarkable proximity of our segmentation results to the human-annotated ground truth.

Furthermore, we conduct a statistical analysis to assess the improvement of our approach on WPS over alternative methods in terms of the F_{B^3} metric. Combining results from 8 methods on 1,969 data instances, we utilize the one-sided Wilcoxon Signed-rank test, selected for paired samples. Specifically, for each atomic type, we compare the F_{B^3} values of our method against each comparator. With a predefined p-value threshold of 0.05, p-values below this threshold signify rejection of the null hypothesis, indicating a statistically significant difference in the distributions favoring our method over the compared approach.

A shown in Table 2, across the *pixel*, *edges-fine*, and *edges-coarse* types, our method exhibits significant improvements over all other methods, with p-values < 0.05. Similarly, for 'chars' and 'nodes', our method demonstrates significant enhancements compared to all

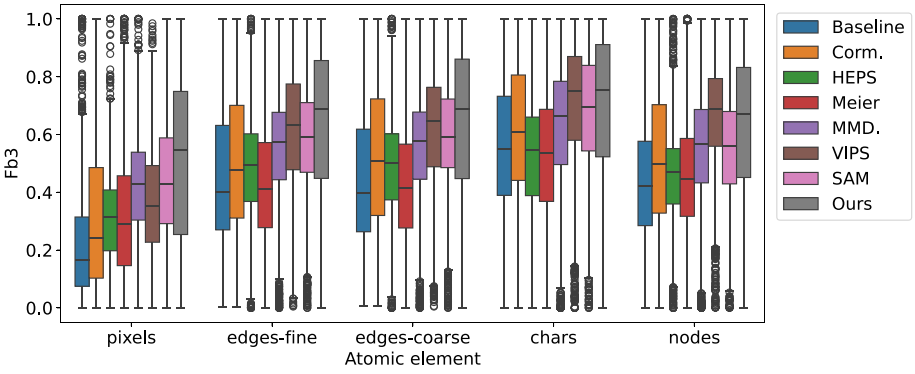


Figure 6 The F_{B^3} score distribution for 1,969 data instances in the Webis-WebSeg-20 dataset [17]. Each box in the representation displays the median with a central mark, while the lower and upper edges of the box represent the 25th and 75th percentiles, respectively. The outliers are represented individually using the symbol ♦ on the plot

Table 2 P-values obtained from one-sided Wilcoxon signed-rank tests comparing our method with other methods on the metric F_{B^3}

Ours vs.	Baseline	VIPS	HEPS	Corm	MMD	Meier	SAM
Pixels	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Chars	**0.0**	0.998	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Nodes	**0.0**	1.0	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Edges-fine	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Edges-coarse	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

The bold values indicate the statistically significant difference between our method and the compared one

methods except VIPS, with p-values of 0.998 and 1, respectively. The test also validates these findings by aligning with the average F_{B^3} results, indicating that our method achieves the highest F_{B^3} results on *pixel*, *edges-fine*, and *edges-coarse*, and obtains slightly lower F_{B^3} scores for 'chars' and 'nodes' compared to VIPS.

Runtime complexity and scalability Considering the runtime, our approach requires 1,240 s to segment all 1,969 Web pages in our dataset, averaging approximately 0.63 s per page. In contrast, the SAM takes about 9.4 s per page on average, making it 15 times slower than our approach. The most time-consuming component of our approach is Algorithm 1, which involves constructing a matrix of tree edit distances. The runtime of this algorithm depends on the number of nodes in each Web page. The largest Web page in our dataset contains 5,100 nodes and requires nearly 60 s to process. However, for 88% of our dataset, this task takes less than 0.5 s.

4.5 Discussion

To have a deeper insight, we conducted an investigation into cases where we achieved the highest and lowest F_{B^3} scores. Our goal was to gain a better understanding of the factors influencing the performance of our algorithm.

For the cases in which our algorithm was poorly performed, we see a common issue that the Web page of these cases does not have a major dissimilarity in structure, such as the case that the whole page is full of a single tag name. As the main idea of our algorithm is to group similarly structured elements, our algorithm will consider the whole page as a single segment.

Conversely, our algorithm demonstrated remarkable performance in cases where Web pages demonstrated easily distinguishable structures, with clear divisions for different functional areas such as the menu, navigation bar, content area, and footer. In these instances, the algorithm effectively identified and grouped elements based on their distinct structural patterns, enabling accurate segmentation and analysis.

In our approach, we specifically address the challenge posed by human-crafted parameters. We achieve this by incorporating an *outlying block elimination* step that is designed to be flexible and adaptive to the characteristics of each individual website. Moreover, our method takes the DOM tree as an input and returns the list of DOM nodes representing the segments; we thus are robust to the change of the relative position of elements as well as variations in element separators.

5 Qualitative evaluation

In addition to the quantitative evaluation, we also conduct a qualitative evaluation for a more comprehensive understanding of the performance and effectiveness of our segmentation approach.

5.1 Procedure

For our qualitative assessment, we have selected three highly competitive approaches—SAM, VIPS, and MMDetection—to compare against our method. These algorithms exemplify the strengths of their respective methodologies. For instance, VIPS utilizes the DOM as input and achieves the highest average F_{B^3} scores for both 'nodes' and 'chars.' In contrast, SAM and MMDetection are purely computer vision-based approaches, securing the second and third-best results for the *pixel* atomic type, which underscores their effectiveness in visual segmentation.

We randomly selected a subset of 500 Web pages from our dataset, which originally consists of 1,969 data instances, to conduct our evaluation. The segmentation results for each page are then be ranked by three evaluators. To ensure an unbiased evaluation, evaluators were blinded to the identities of the segmentation methods.

Each evaluator was tasked with ranking the segmentation results of the four methods for each Web page in order of preference. It was ensured that no two results were assigned the same rank, with 1 denoting the most favorable result and 4 the least. These evaluations were conducted autonomously and independently of one another.

5.2 Criteria

Evaluators follow a set of criteria during their evaluation, which involves compactness, distinctness, and coverage.

- **Coverage:** Segments should cover all visible elements on the Web page without omission. This criterion evaluates the completeness of segmentation across the entire Web page.
- **Compactness:** This refers to the degree to which semantically equivalent elements are grouped together within the same segment. Higher compactness indicates better grouping of related elements.
- **Distinctness:** This criterion assesses how well the segmentation method distinguishes between functionally different elements, ensuring they are placed in separate segments. A higher score indicates clearer separation of distinct elements.

5.3 Results

Table 3 shows the results obtained from the qualitative assessment of four methods, VIPS, SAM, MMDetection, and ours. The second to forth column represents the results for the ranking from 1 (most favorable) to 4 (least favorable), respectively. The results are shown in the format of M/σ, where M denotes the average number of Web pages receiving the respective ranking and σ the standard deviation. The last column W shows the weighted average, $W = \frac{\sum_{i=1}^{n} w_i x_i}{\sum_{i=1}^{n} w_i}$, and its respective standard deviation.

Our method received 296 out of 500, on average, segmented pages being ranked the most favorable (1), much higher than the other methods. This result also means that most pages

Table 3 Qualitative assessment results

	1	2	3	4	W
VIPS	105/3	**204/11**	110/7	81/6	2.33/0.05
MMD.	48/7	109/3	**176/3**	167/8	2.92/0.05
SAM	50/5	105/7	162/12	**182/1**	2.95/0.03
Ours	**296/9**	82/2	52/7	70/2	**1.79/0.04**

ranked 1, many more than those being ranked 2, 3, and 4. The second best approach is VIPS, which received 105 and 204 segmented pages being ranked 1 and 2, respectively.

These results reveal a clear trend across all evaluators, demonstrating a strong preference for our segmentation method over VIPS, SAM, and MMDetection. Our approach consistently outperformed the other methods, as indicated by its notably lower weighted average rank of 1.79.

This result suggests better adherence to the evaluation criteria, particularly in terms of distinctness, compactness, and coverage. Notably, our method secured the top ranking from all evaluators in nearly 60% of the dataset instances, highlighting its overall effectiveness.

Furthermore, the average rankings among evaluators closely align with the quantitative evaluation metrics. Specifically, our method achieved the highest F_{β^3} score, surpassing the other three methods.

While VIPS remained the best for node and character segmentation in the quantitative evaluation, it ranked second overall, following our method. Despite SAM improving by 10% in pixel segmentation over VIPS and marginally outperforming MMDetection, it was the least favored by evaluators. A closer inspection of SAM's segmentation results across different Web pages reveals a consistent trend: the algorithm tends to segment by color boundaries, which works well in cases where menus or footers have distinct colors. However, when dealing with pages that are primarily text-based, SAM tends to segment some text by size and group all other content into a single segment, making it less preferred by evaluators. The other computer vision-based approach, MMDetection, ranking just 0.03 below SAM, suffers from similar issues.

Finally, the standard deviations of the weighted ranks are all below 0.05, indicating a high level of agreement among evaluators.

6 Related work

6.1 WPS Approaches

There have been many approaches proposed for Web page segmentation over the past two decades. The Gestalt principles of perception are used to group neighboring nodes under the same parent if their similarity measure exceeds 0.7, determined by their edit distance [31]. BoM [27] consists of three phases: building a content structure from the DOM tree, mapping it to a logical structure based on a granularity parameter, and structuring it into a single representation that represents the segmented Web page. VIPS [7] extracts a hierarchical semantic structure by combining the DOM structure with visual cues through block extraction, separator detection, and content structure construction. In addition, the HEPS method [22], mentioned in the Webis-WebSeg-20 dataset [17] for comparison, utilizes text nodes and images to identify potential headings, corresponding blocks, and create a hierarchical segmentation. The DOM structure is also a vital component in other segmentation

models [15, 16], where additional factors like textual and visual cues are integrated to enhance performance.

A purely text-based approach, the Block Fusion (BF) algorithm [20], focuses on comparing the text density of adjacent blocks to determine whether they should be combined.

There are also approaches that treat a Web page as an image and use computer vision techniques for segmentation. One such approach recursively splits segments into two by selecting the vertical and horizontal lines with the clearest edge pixels (detected through edge detection) from the entire page [13]. Another approach in this research direction fine-tuned the hybrid task cascade model from MMDetection for Web page segmentation [11].

6.2 Applications of WPS

WPS has been widely applied in various research. Segmentation on the Web can help to determine informative and non-informative sections within a Web page and can distinguish between various types of information. This distinction is valuable for tasks like Web ranking and data mining on the Web [10].

WPS is also used to automatically adjust the user interface (UI) to accommodate various devices, by reorganizing or redecorating Web pages accordingly [2].

WPS has been employed in Fraggen [33] for Web testing, enhancing state exploration, and test oracle generation. By dividing pages into fine-grained fragments, Fraggen detects 123% more near-duplicates, infers crawl models with 62% higher precision and 70% more recall, and produces robust regression test suites with near-perfect success rates.

Web UI testing has also benefited from the use of WPS. In [14], WPS was successfully applied to test case prioritization. Rather than attempting to cover all possible Web elements, the approach focused on exploring as many segments of the Web page as possible. This method demonstrated a 20% reduction in the average time required to detect an error.

7 Conclusion

This paper proposes a novel method for Web page segmentation using DOM-structural analysis. Our method overcame the limitations of parameter-dependent and rule-based methods by employing flexible techniques that adapt to the characteristics of each website being analyzed. The proposed method involves generating blocks, analyzing their cohesion, and applying the Canny algorithm for a more user-friendly layout. We perform a thorough evaluation of our technique, encompassing both quantitative and qualitative analyses. In the quantitative aspect, we assess our method using a dataset consisting of 1,969 Web pages sourced from the Webis-WebSeg-20 dataset [17]. Our findings exhibit significant efficacy, as our approach achieves a 64% F_{B^3} score, outperforming VIPS and MMDetection by 19% and 10% respectively, on the same dataset. In terms of qualitative assessment, our method surpasses the two comparative methods, consistently earning the highest ratings across evaluations conducted by five independent evaluators.

Future research will focus on improving our segmentation method to better analyze various type of applications. Better segmentation outcomes can be obtained by combining additional variables to provide multimodal cues, such as visual or textual features. Additionally, incorporating computer vision techniques can facilitate our method and produce a model that is more reliable.

Acknowledgements We would like to thank Katalon, University of Science - VNUHCM, and VinIF for supporting this research.

Author Contributions H.H. and T.L. method proposal, implementation, analysis, data collection, and writing. V.N. idea, analysis, and writing. T.N. writing.

Funding not applicable.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Ethical Approval not applicable.

Conflict of interest The authors declare no Conflict of interest.

References

1. Alcic, S., Conrad, S.: Page segmentation by Web content clustering. In: Proceedings of the WIMS. pp. 1–9 (2011)
2. Baluja, S.: Browsing on small screens: Recasting Web-page segmentation into an efficient machine learning framework. In: Proceedings of the Fifteenth International World Wide Web Conference. Edinburgh, Scotland (2006). http://www.esprockets.com/papers/www2006-2502-baluja.pdf
3. Bar-Yossef, Z., Rajagopalan, S.: Template detection via data mining and its applications. In: Proceedings of the 11th WWW. pp. 580–591 (2002)
4. Brisset, S., Rouvoy, R., Seinturier, L., Pawlak, R.: Sftm: Fast matching of Web pages using similarity-based flexible tree matching. Inf. Syst. **112**, 102126 (2023)
5. Cai, D., He, X., Li, Z., Ma, W.Y., Wen, J.R.: Hierarchical clustering of www image search results using visual 01 (2004)
6. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: Extracting content structure for Web pages based on visual representation. In: Proceedings of the 5th APWeb. p. 406–417. APWeb'03, Springer-Verlag, Berlin, Heidelberg (2003)
7. Cai, D., Yu, S., Wen, J.R., Ma, W.Y.: Vips: a vision-based page segmentation algorithm 01 (2003)
8. Canny, J.: A computational approach to edge detection. IEEE Trans. Pattern Anal. Mach. Intell. **6**, 679–698 (1986)
9. Chakrabarti, D., Kumar, R., Punera, K.: A graph-theoretic approach to webpage segmentation. In: Proceedings of the 17th WWW. pp. 377–386 (2008)
10. Chakrabarti, D., Kumar, R., Punera, K.: A graph-theoretic approach to webpage segmentation. pp. 377–386 (08 2008). https://doi.org/10.1145/1367497.1367549
11. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: CVPR. pp. 4974–4983 (2019)
12. Chen, K., Wang, J., Pang, J., Cao, Y., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Xu, J., et al.: Mmdetection: Open mmlab detection toolbox and benchmark. arXiv:1906.07155 (2019)
13. Cormer, M., Mann, R., Moffatt, K., Cohen, R.: Towards an improved vision-based Web page segmentation algorithm. In: 2017 14th CRV. pp. 345–352. IEEE (2017)
14. Huynh, H., Pham, N., Nguyen, T.N., Nguyen, V.: Segment-based test case prioritization: A multi-objective approach. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis. pp. 1149–1160 (2024)
15. Jayashree, S.R., Dias, G., Andrew, J.J., Saha, S., Maurel, F., Ferrari, S.: Multimodal Web page segmentation using self-organized multi-objective clustering. ACM Trans. Inf. Syst. **40**(3) (2022).https://doi.org/10.1145/3480966
16. Jiang, Z., Yin, H., Wu, Y., Lyu, Y., Min, G., Zhang, X.: Constructing novel block layouts for webpage analysis. TOIT **19**(3), 1–18 (2019)
17. Kiesel, J., Kneist, F., Meyer, L., Komlossy, K., Stein, B., Potthast, M.: Web page segmentation revisited: Evaluation framework and dataset. In: Proceedings of the 29th ACM CIKM. pp. 3047–3054. CIKM '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3340531.3412782

18. Kiesel, J., Meyer, L., Kneist, F., Stein, B., Potthast, M.: An empirical comparison of Web page segmentation algorithms. In: ECIR 2021, Virtual Event, March 28–April 1, 2021, Proceedings, Part II 43. pp. 62–74. Springer (2021)
19. Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R.: Segment anything. arXiv:2304.02643 (2023)
20. Kohlschütter, C., Nejdl, W.: A densitometric approach to Web page segmentation. In: Proceedings of the 17th ACM CIKM. pp. 1173–1182 (2008)
21. Lu, C., Bing, L., Lam, W.: Structured positional entity language model for enterprise entity retrieval. In: 22nd ACM CIKM. pp. 129–138 (2013)
22. Manabe, T., Tajima, K.: Extracting logical hierarchical structure of html documents based on headings. Proc. VLDB Endow. **8**(12), 1606–1617 (2015). https://doi.org/10.14778/2824032.2824058
23. Manku, G.S., Jain, A., Das Sarma, A.: Detecting near-duplicates for Web crawling. In: Proceedings of the 16th WWW. pp. 141–150 (2007)
24. Meier, B., Stadelmann, T., Stampfli, J., Arnold, M., Cieliebak, M.: Fully convolutional neural networks for newspaper article segmentation. In: 2017 14th ICDAR. vol. 1, pp. 414–419. IEEE (2017)
25. Narayana, V., Premchand, P., Govardhan, A.: A novel and efficient approach for near duplicate page detection in Web crawling. In: 2009 IACC. pp. 1492–1496. IEEE (2009)
26. Pawlik, M., Augsten, N.: Tree edit distance: Robust and memory-efficient. Inf. Syst. **56**, 157–173 (2016). https://doi.org/10.1016/j.is.2015.08.004
27. Sanoja, A., Gançarski, S.: Block-o-matic: A Web page segmentation framework. In: 2014 ICMCS. pp. 595–600 (2014). https://doi.org/10.1109/ICMCS.2014.6911249
28. Su, T., Yan, Y., Wang, J., Sun, J., Xiong, Y., Pu, G., Wang, K., Su, Z.: Fully automated functional fuzzing of android apps for detecting non-crashing logic bugs. Proc. ACM Program. Lang. **5**(OOPSLA), 1–31 (2021)
29. Velloso, R.P., Dorneles, C.F.: Automatic Web page segmentation and noise removal for structured extraction using tag path sequences. JIDM **4**(3), 173–173 (2013)
30. Vieira, K., Da Silva, A.S., Pinto, N., De Moura, E.S., Cavalcanti, J.M., Freire, J.: A fast and robust method for Web page template detection and removal. In: 15th ACM CIKM. pp. 258–267 (2006)
31. Xiang, P., Yang, X., Shi, Y.: Web page segmentation based on gestalt theory. In: 2007 IEEE ICME. pp. 2253–2256. IEEE (2007)
32. Xie, X., Miao, G., Song, R., Wen, J.R., Ma, W.Y.: Efficient browsing of Web search results on mobile devices based on block importance model. In: 3rd IEEE PerCom. pp. 17–26. IEEE (2005)
33. Yandrapally, R.K., Mesbah, A.: Fragment-based test generation for Web apps. IEEE Trans. Softw. Eng. **49**(3), 1086–1101 (2023). https://doi.org/10.1109/TSE.2022.3171295
34. Yandrapally, R., Sinha, S., Tzoref-Brill, R., Mesbah, A.: Carving ui tests to generate api tests and api specification. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE). pp. 1971–1982. IEEE (2023)
35. Yi, L., Liu, B., Li, X.: Eliminating noisy information in Web pages for data mining. In: 9th KDD. pp. 296–305 (2003)
36. Yin, X., Lee, W.S.: Understanding the function of Web elements for mobile content delivery using random walk models. In: Special interest tracks and posters of the 14th WWW. pp. 1150–1151 (2005)