Page-Level Main Content Extraction from Heterogeneous Webpages

JULIÁN ALARTE, Universitat Politècnica de València, Spain
JOSEP SILVA, Universitat Politècnica de València, Spain

The main content of a webpage is often surrounded by other boilerplate elements related to the template, such as menus, advertisements, copyright notices, comments, etc. For crawlers and indexers, isolating the main content from the template and other noisy information is an essential task, because processing and storing noisy information produce a waste of resources such as bandwidth, storage space, computing time, etc. Besides, the detection and extraction of the main content is useful in different areas, such as data mining, web summarization, content adaptation to low resolutions, etc. This work introduces a new technique for main content extraction. In contrast to most techniques, this technique not only extracts text, but also other types of content, such as images, animations, etc. It is a DOM-based page-level technique, thus it only needs to load one single webpage to extract the main content. As a consequence, it is efficient enough as to be used online (in real-time). We have empirically evaluated the technique using a suite of real heterogeneous benchmarks producing very good results compared with other well-known content extraction techniques.

CCS Concepts: • **Information systems → Web mining**; **Document filtering**; **Presentation of retrieval results**.

Additional Key Words and Phrases: Information Retrieval, Content Extraction, Template Extraction, Web Mining, Block Detection

ACM Reference Format:
Julián Alarte and Josep Silva. 2021. Page-Level Main Content Extraction from Heterogeneous Webpages. 1, 1 (February 2021), 20 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The information contained in a webpage can be classified as relevant or irrelevant content according to the user needs. For this reason, extracting information (relevant or irrelevant) from webpages is a productive task for computer systems and humans. The relevant content in a webpage is often referred to as *main content* [3, 5, 8, 24, 38]. It can be formed from text, images, and any other multimedia; and it is usually surrounded by or even mixed with noisy information such as headers, footers, menus, banners, advertisements, etc. Removing this noisy and irrelevant information from a webpage is essential to extract the relevant data for the user (see, e.g., Figure 1).

The focus of this paper are HTML-structured webpages, thus it ignores webpages built with alternative technologies such as Flash. From an engineering point of view, a webpage corresponds to a set of Document Object Model (DOM) nodes [10]. Accordingly, the main content of a webpage can be defined as a subset of those nodes, and it contains the meaningful information of a webpage.

Authors' addresses: Julián Alarte, Universitat Politècnica de València, Departament de Sistemes Informàtics i Computació, Camí de Vera s/n, València, E-46022, Spain, jualal@doctor.upv.es; Josep Silva, Universitat Politècnica de València, Departament de Sistemes Informàtics i Computació, Camí de Vera s/n, València, E-46022, Spain, jsilva@dsic.upv.es.

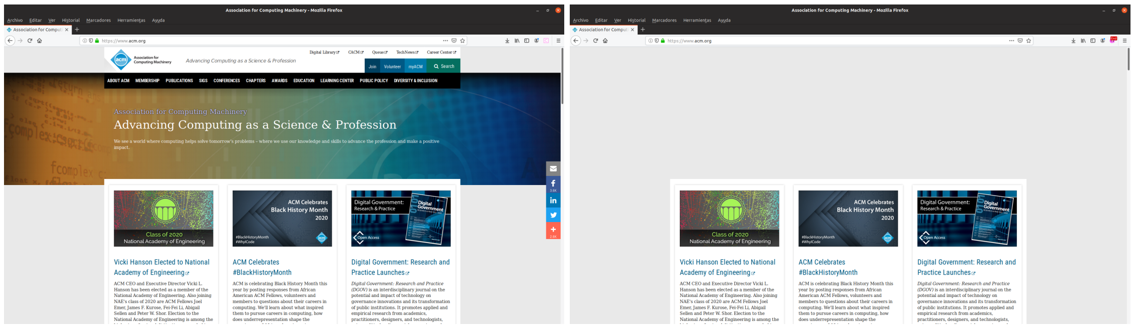
Association for Computing Machinery - Mozilla Firefox
Advancing Computing as a Science & Profession
Vicki Hanson Elected to National Academy of Engineering
ACM Celebrates #BlackHistoryMonth
Digital Government: Research and Practice Launches

Fig. 1. Main webpage of ACM's website and its main content (extracted with our web content extraction tool).

The main content is a key element for indexers and crawlers:

- Gibson et al. [16] determined that template elements comprise almost half of all the data on the Web. This fact justifies the importance of using techniques such as main content extraction, or template extraction [33, 38] as a preprocess method.
- Processing the noisy elements of webpages can lead to a waste of resources like storage, bandwidth, or time. Thus, indexers and crawlers preprocess the webpages to isolate the main content from the noisy content. Due to its relevance, the main content is indexed and stored in another way.

Our approach to main content extraction uses DOM structures for representing webpages. Roughly, given an arbitrary webpage, (1) we first assign weights to several features of its DOM nodes. This allows us to (2) represent DOM nodes as points in a four-dimensional Euclidean space $\mathbb{R}^4$, and then (3) we compute the Euclidean distance between the points and isolate those nodes further away than the median because this set probably contains the main content. Then, (4) the selected DOM nodes are analyzed to identify the part of the web page's DOM tree that contains the main content. The main contributions of this work are: (i) the new four-dimensional Euclidean space proposed and the algorithms used to extract the main content in such a space; (ii) the implementation of our technique as a public and open-source official Firefox add-on. The open-sourced implementation of six other content extraction algorithms that were not public; and (iii) an extensive empirical evaluation of the most important content extraction techniques.

2 THE WEBPAGE'S MAIN CONTENT

In general, it is easy for humans the identification of the main content in a rendered image of a webpage. When we represent a webpage with a DOM tree, however, there usually exist several nodes whose subtree contains the main content. Often, DOM nodes form a complex hierarchy where one node can contain exactly the same text and images as some of its ancestors (e.g., a DIV element with a single child). So, what node should be chosen? The answer to this question should be given according to a design policy. In order to provide a definition of main content in a DOM tree, first, we need to formally define the concepts of website and webpage.

Definition 2.1 (Webpage). A webpage P is a tree (N, A) formed from a finite set of nodes N. Every non-leaf node $n \in N$ contains an HTML tag (including its attributes). Leaf nodes can be text nodes, CDATA section nodes, or comment nodes. The root node corresponds to the body HTML tag. A is a finite set of arcs such that $(n \longrightarrow n') \in A$, with $n, n' \in N$, if and only if the tag or text associated with n' is inside the tag associated with n, and there does not exist an unclosed tag between them.

Given a node n in a webpage P, $ancestors(n)$ is a set of nodes that contains all the nodes that are in the path from the root to n, $descendants(n)$ are those nodes that belong to the subtree of n, $subtree(n) = descendants(n) \cup \{n\}$, $leaves(n)$ is the number of leaves in $descendants(n)$, $childNodes(n)$ is the number of children of n, $depth(n)$ and $maxDepth(P)$ are, respectively, the maximum depth of $subtree(n)$ and P, $words(n)$ is the total number of words in $descendants(n)$ excluding those that belong to hyperlinks, and $distance(n_1, n_2)$ is the length (measured in number of edges) of the path between two nodes n_1 and n_2. Given the domain W of webpages, the set of hyperlinks is a relation $R \subseteq W \times W$. We represent with $hyperlinks(P)$ the set of hyperlinks of a webpage P.

A website consists of webpages sharing a prefix in their URI, and all of them (possibly except for the main webpage) are reachable from another webpage of the website. Hence, all the webpages are linked by at least one hyperlink in another webpage of the same website (i.e., all webpages have an indegree greater than 0).

Definition 2.2 (Website). A website S is a set of webpages such that

- $\exists\, U \ :\ \forall\, P \in S,\ U$ is a non-empty prefix of P's URI.
- $\exists\, P_{root} \in S \ :\ \forall\, P' \in S, P_{root} \neq P' \ :\ (P_{root}, P') \in hyperlinks(S)^*$, where $hyperlinks(S) = \cup_{Q \in S}\ hyperlinks(Q)$, and X^* is the reflexive and transitive closure of X.

We can now provide a definition of main content. Roughly, we can define the main content as the information provided by a webpage excluding template data, side information like comments or advertisements, and meta data like publication date. However, it is important to remark that the main content of a webpage is subjective. A clear example is found in a webpage that displays a news article: the comments of the readers are considered main content by some people (thus, they should be extracted together with the new), while others consider that this part does not belong to the new (and thus they should not be extracted). Therefore, providing a definition of main content is controversial.

We follow an engineering perspective based on the structure of the webpage. To provide a definition that is independent of any method, for a webpage $P = (N, A)$, we assume the existence of a labelling $relevant(n)$ for the leaf nodes that identifies those leaves in the webpage that should belong to the main content. Formally,

Definition 2.3 (Main content). The main content of a webpage $P = (N, A)$ is a set of DOM nodes $M \subset N$ such that:

(1) *All relevant nodes belong to the subtrees of the main content nodes:*
$\forall\, n \in N,\ relevant(n)\ .\ n \in subtree(n' \in M)$.
(2) *All nodes that belong to the subtrees of the main content nodes are relevant:*
$\forall\, n\ \in\ leaves(n' \in M)\ .\ relevant(n)$.
(3) *The set of main content nodes is minimal:*
$\nexists\, M' \subset M\ .\ \forall\, n \in N,\ relevant(n),\ n \in subtree(n' \in M')$.

Definition 2.3 is useful when we have a way to provide the *relevant* labelling. This is the case, for instance, when one is evaluating benchmarks. If one does not have available the *relevant* labelling, this must be approximated. In the following sections, we propose a method to automatically create the *relevant* labelling.

3 MAIN CONTENT EXTRACTION

The main content extraction technique proposed ranks the DOM nodes with features to identify the main content of a webpage. The technique inputs a webpage and it outputs a set of DOM nodes representing its main content. As it is a page-level technique, it only loads and analyzes one single webpage to detect the main content. This is especially important because loading and analyzing one single webpage increases the speed of the algorithm.

Our technique is divided into four phases:

(1) An algorithm selects some DOM nodes of the webpage and, for each one, several weights (word ratio, hyperlink ratio, children ratio, and position ratio) are computed.
(2) For each node with its weights assigned, an algorithm standardizes the value of its weights.
(3) Once the weights are standardized, each node is considered a point in $\mathbb{R}^4$. Then, an algorithm visits all these nodes (points) and computes the centroid. The set of DOM nodes (points) that are farther than the centroid are added to a set of *candidate nodes*.
(4) The nodes in the set of *candidate nodes* are analyzed in the following way:
    - Those nodes that are descendants of other nodes in the set are removed if they have exactly the same text nodes as their ancestors.
    - The algorithm searches for the node with a better ratio between words and tags. This node together with its siblings that belong to the set of *candidate nodes* are selected as the main content.

The following sections describe these four phases.

3.1 Weighting DOM nodes

This section introduces a metric to identify those DOM nodes that can potentially be the root node of the webpage's main content. All the ideas proposed have been empirically validated with a set of 65 webpages taken from the TECO benchmark suite (see section 4).

First, the DOM tree of the webpage is explored in order to compute and assign a weight to each DOM node meeting the following criteria:

(1) It is not a leaf of the DOM tree.
(2) It is a node of type *element* [1] and its tagName is different from the following: "A", "NAV", "UNDEFINED", "HR", "SPAN", "EM", "BODY", "SCRIPT", "HEADER", "H1", "H2", "H3", "H4", "H5", "BR", and "IFRAME". Nodes of type different from *element* (e.g., *text* nodes, comments, etc.) are not considered. Note, however, that the weighted nodes may contain blocks of text.
(3) If the depth of the webpage from the "BODY" DOM node (measured as the number of nodes with the tagNames listed in bullet 2) is less than the number of children of the "BODY" DOM node (measured with the same criterion), the main content of the webpage corresponds to the union of the children of the "BODY" DOM node. This idea has been empirically validated with a set of 65 webpages. Since none of the 65 webpages selected from TECO met the criterion, we selected 65 random webpages from the CleanEval dataset. As a result, 11 of the 65 webpages met the criterion. We measured the percentage of text of the whole webpage that also belong to the main content, obtaining an average value of 91.07%, which means that the main content of that webpages corresponds practically to the whole webpage. It should be highlighted that we observed that this phenomenon only occurs in old webpages.

The rationale behind these criteria is that, in the DOM model, *text* nodes are always leaves, and they are always inside an *element* node. Thus, it is always possible to select a text (or image, etc.) by selecting the element node that contains this text. Moreover, element nodes that are leaves do not contain any visible information (text, images, etc.). Therefore, the main content can be always selected with an element node that is not a leaf. This is why we discard leaves and nodes that are not elements. In addition, the tagNames listed above (bullet 2) cannot be the main content or there are other tags that subsume them and, thus, they are also discarded. Finally, the third criteria only applies to webpages that are very wide and thus the content is distributed between various branches. This criterion was determined empirically, and it avoids selecting a branch with a small concentration of text. Note that this does not mean that the main content node has at most the third of the text of the whole webpage, because the parent node can be selected. It just discards those nodes that contain small amounts of text. Its application increases the precision in 1-5% (see the empirical evaluation in Section 4).

It has been observed (see, e.g., [36]) that the main content of a website usually contains a high density of text and images; but, in general, this density is not enough by itself to detect the main content [19]. In the following, we propose several properties that complement the text and images density and that must be considered to appropriately detect the main content. All these properties are quantified objectively and they are properly combined to form a weighting that can be used to identify the main content DOM nodes.

Definition 3.1 (Node properties). Given a webpage $P = (N, A)$, every node $n \in N$ with $descendants(n) \neq \emptyset$ is rated according to the following properties:

Word ratio: It considers the number of words not included in an hyperlink and their depth. The algorithm assigns a higher value to the words nearer the node:

$$wordRatio(n) = \textstyle\sum_{k \in leaves(n)} words(k)/distance(k,n)$$
$$\text{where } parent(k).tagName \neq \text{“}A\text{”}$$

Hyperlink ratio: It is computed considering the amount of hyperlinks contained in the descendants of a node *n*:

$$hyperlinkRatio(\text{n}) = \begin{cases} 1 & \text{if } links = 0 \\ 1/links & \text{if } links > 0 \end{cases}$$
$$\text{where } links = hyperlinks(subtree(n))$$

Children ratio: It checks whether a node *n* has more than two children:

$$childrenRatio(\text{n}) = \begin{cases} 0 & \text{if } childNodes(n) \leq 2 \\ 1 & \text{if } childNodes(n) > 2 \end{cases}$$

Position ratio: It is computed using the following function:

$$positionRatio(\text{n,P}) = \begin{cases} 1 & \text{if } depth(n) \leq maxDepth(P)/2 \\ \frac{maxDepth(p)}{depth(n)} - 1 & \text{if } depth(n) > maxDepth(P)/2 \end{cases}$$

The *Word ratio* property defines a metrics to account for the amount of text words contained in the descendants of a DOM node. The value is cumulative for all descendants, and it is computed for each descendant as the amount of words it contains divided by the distance from the node to *n*. Therefore, this metrics encourages the DOM nodes with a high amount of text in its nearest descendants. The farther the text is from a DOM node, the lower its *word ratio* is. Our experiments reveal that, as an average, each main content node contains 2.91 words (outside hyperlinks), while non-main content nodes contain 0.68 words (4.29 times less words).

The *Hyperlink ratio* property computes the number of hyperlinks contained in a DOM node and its descendants. The more hyperlinks, the lower the link ratio. The main content of a webpage contains less hyperlinks than other blocks, such as the main menu, the footer, etc. Therefore, this metric penalizes those DOM nodes with many hyperlinks in their descendants. Our experiments reveal that, as an average, 70.43% of the hyperlinks in the webpages did not belong to the main content.

The *Children ratio* property promotes those nodes with more than two children. The main content of a webpage usually contains several element nodes with text nodes between their descendants. Therefore, those nodes with two children or more are assigned a higher weight because they allow for the inclusion of intermediate text nodes in the DOM tree. Our experiments reveal that, as an average, 96.92% of the main content nodes have more than 2 children.

The *Position ratio* property evaluates the depth of a DOM node in the DOM tree. The main content is often located in the first half of the DOM tree. Therefore, this metric progressively penalizes those DOM nodes located at the lower positions in the DOM tree. In our experiments, in 87.69% of the webpages, the main content's root node belong to the first half of the DOM tree.

Once computed, these properties are assigned to each node in the DOM tree. A higher value for these properties indicates that the DOM node is more likely to contain the main content of the webpage. As these properties are dissimilar from each other, there is not a ponderation used to combine them all. However, we developed a novel method to compare the rated DOM nodes in order to determine which ones probably contain the main content. This technique is explained in the following subsections.

3.2 Properties standardization

Our algorithm introduces a novel technique to compute the differences between DOM nodes. The four ratios assigned in Definition 3.1 represent a DOM node as a point in a four-dimensional Euclidean space $\mathbb{R}^4$. Then, the relative Euclidean distance between these points can be used to compare the DOM nodes in the webpage. The four ratios have been designed to distinguish the main content node from the rest, so that most DOM nodes must be relatively close to each other in $\mathbb{R}^4$, but a DOM node that contains the main content should be located farther the rest of nodes. This is validated in the empirical evaluation (see Section 4) and relies on the fact that the value of the node properties is considerably different whether a node contains the main content or not. A node that contains the main content usually contains a large amount of text between its descendants, less hyperlinks between its descendants than other nodes, several children, and a position located in the first half of the DOM tree. Most of the nodes in the DOM tree do not meet these properties, hence the centroid is located with a high probability near them in $\mathbb{R}^4$ and, therefore, far from the nodes that contain the main content. Prior to the DOM node comparison, the four ratios computed in Section 3.1 must be standardized. The standardization process ensures that all the ratios do have the same impact on the distance measurement [24]. The standardization process replaces the value of a ratio with the difference between that value and the average of the values taken by it divided by the standard deviation.

Definition 3.2 (Node standardization). The different ratios of a node r are standardized with the following formula:

$$r_i = (r_i - \overline{r_i})/s_{r_i} \tag{1}$$

where $\overline{r_i}$ is the average of the values taken by r_i, and s_{r_i} is the standard deviation.

In the following we represent DOM nodes as points in $\mathbb{R}^4$.

Definition 3.3 (DOM nodes representation in $\mathbb{R}^4$). A DOM node A is represented in a Euclidean space $\mathbb{R}^4$ with a quadruple (a, b, c, d), where a, b, c, d are the four ratios assigned in Definition 3.1. Two nodes $A = (a, b, c, d)$ and $A' = (a', b', c', d')$ are the same node if and only if $a = a' \wedge b = b' \wedge c = c' \wedge d = d'$.

The use of $\mathbb{R}^4$ provides interesting advantages over other approaches. For instance, the technique MenEx [2] also uses different ratios to identify the DOM node that represents a menu, but they judge the importance of a DOM node with the sum of the different ratios. This means that they cannot distinguish between two DOM nodes with the same values for different ratios, e.g., $(1, 2, 3, 4) = (4, 3, 2, 1)$ because $1 + 2 + 3 + 4 = 4 + 3 + 2 + 1$. Representing DOM nodes as points in the Euclidean space automatically distinguish between any combination of ratios, and it also defines a distance between them: the Euclidean distance.

Definition 3.4 (Euclidean distance). The Euclidean distance between a node A and a node B for n ratios (in $\mathbb{R}^n$) is computed with the following formula:

$$eucli\_distance(A, B) = \sqrt{\sum_{i=0}^{n} (A.ratio[i] - B.ratio[i])^2} \tag{2}$$

This distance also provides interesting properties. For instance, to empirically determine the best ponderation for the ratios used in [2], the authors had to limit the number of experiments they did due to the combinatorial explosion. For instance, they tried with $0.1 * ratio1 + 0.2 * ratio2 + ...$, with $0.2 * ratio1 + 0.2 * ratio2 + ...$, and so on. In contrast, the Euclidean distance automatically allows us to determine how far one point is from another, in spite of them having the same values for different ratios.

3.3 c-SET computation

In this section, the DOM nodes that are more likely to contain the main content of the webpage are added to the set of candidate nodes (c-SET). The c-SET is further analyzed in order to identify the DOM node that contains the main content.

The first step to obtain the c-SET is to compute the centroid of the nodes. The centroid is the arithmetic mean position of all the points in $\mathbb{R}^4$ represented by the rated DOM nodes.

Algorithm 1 computes the centroid of all the rated nodes in a webpage. The centroid is a DOM node surrounded (in $\mathbb{R}^4$) by nodes that very unlikely contain the main content. Note that the value of the properties (see Definition 3.1) assigned to non-content nodes is close to zero, so they must be located near the coordinate axes. Therefore, the set of candidate nodes (c-SET) is built with the DOM nodes located farther from the centroid in $\mathbb{R}^4$, since the value assigned to their properties is significantly high. This is computed with Algorithm 2, which measures the distance from all the rated DOM nodes in the webpage to the centroid, and it selects the c nodes farther from it. In our implementation, the value of c has been determined with an empirical evaluation. It is explained in section 4.

The nodes in the c-SET must be analyzed to select the one that more likely contains the main content.

Algorithm 1 Centroid computation

```
Input: A set of rated DOM nodes $ratedNodes = \{n_1 \dots n_i\}$
Output: The centroid $c$ of $ratedNodes$
begin
    for $prop = 0$ to 3
        $c.ratio[prop] = (\sum_{node=1}^{i} n_{node}.ratio[prop])/i$;
    return $c$;
end
```

Algorithm 2 c-SET computation

```
Input: A set of rated DOM nodes $ratedNodes$, its centroid $cent$, and its size $c$
Output: A set of DOM nodes $c-SET$
begin
    foreach ($n_1$ in $ratedNodes$)
        $sum = 0$;
        for $i = 0$ to $3$
            $sum = sum + (n_1.ratio[i] - cent.ratio[i])^2$;
        $n_1.distance = \sqrt{sum}$;
    $c-SET = \emptyset$
    for $i = 1$ to $c$
        $node = n \in ratedNodes\ .\ \nexists n' \in ratedNodes, n'.distance > n.distance$;
        $ratedNodes = ratedNodes \setminus \{n\}$;
        $c-SET = c-SET \cup \{n\}$;
    return $c-SET$;
end
```

Algorithm 3 c-SET reduction

```
Input: A set of candidate DOM nodes $c-SET$
Output: A set of candidate DOM nodes $c-SET$
begin
    foreach ($n_1$ in $c-SET$)
        foreach ($n_2$ in $c-SET$)
            if $n_1.innerText == n_2.innerText$ and $n_1 \in ancestors(n_2)$
                $c-SET = c-SET \setminus \{n_2\}$;
    return $c-SET$;
end
```

3.4 Selecting the main content nodes

In this section we show how to identify the main content nodes among the nodes in the c-SET .

Firstly, Algorithm 3 reduces the number of elements in the c-SET by checking whether two or more nodes from the c-SET contain the same text nodes. This can only happen if one is a descendant of the other. Given two nodes in the c-SET, if a descendant of a node contains the same text nodes as one of its ancestor, then the descendant is removed from the c-SET. The goal of selecting the ancestor is to avoid missing all the non-textual information, such as images that surround the text nodes.

For this, Algorithm 3 explores the c-SET and removes the nodes with an ancestor in the c-SET that contains exactly the same text nodes. Then, Algorithm 4 explores the remaining DOM nodes in the c-SET to select the main content nodes as follows:

- For all the nodes in the c-SET, it computes the ratio between words and tags.
- The node with the highest ratio is selected as the main content node if it does not have any siblings that belong to the c-SET. In case of a tie between two or more nodes, the algorithm selects them all as the main content nodes.
- If the c-SET contains siblings of the node with the highest ratio, then all of them are selected as the main content nodes.

In Algorithm 4, function $getSiblings(n, set)$ returns the sibling nodes of n that belong to the set of nodes set.

3.5 Final post-process

We observed that, in some webpages (around 5%), the extracted main content contains groups of links that do not belong to the main content (e.g., breadcrumbs, links to other sections of the website, etc.). We can remove those groups

Algorithm 4 Main content selection

```
Input: A set of candidate DOM nodes $c-SET$
Output: A set of DOM nodes $mainContent$
begin
    $maxRatio = 0$;
    $maxNode = null$;
    foreach ($n$ in $c-SET$)
        $n.textPond = |n.innerText| / |n.tags|$;
        if $n.textPond > maxRatio$
            $maxRatio = n.textPond$;
            $maxNode = n$;
    $siblings = getSiblings(maxNode, c-SET)$;
    if $siblings == \emptyset$
        $mainContent = maxNode$;
    else
        $mainContent = \{n\} \cup siblings$;
    return $mainContent$;
end
```

of links, thus improving the precision, with a very cheap post-process. Algorithm 5 explores the main content nodes extracted by Algorithm 4 and it removes the groups of links that do not belong to the main content, if any. The process is done as follows:

- For all the nodes in the mainContent set, it computes a ratio called *textRatio* as the amount of text of a node divided by the amount of text of the node excluding its hyperlinks. It also computes the number of hyperlinks in the descendants of the node.
- The nodes with a *textRatio* higher than a computed threshold *tr*, and with a number of hyperlinks higher than another threshold *hr*, are removed from the mainContent set.
- If all children of a DOM node have the same tagName, and in turn, each one only contains one child node whose tagName is "A", they are removed if there are not images in their descendants.

Algorithm 5 Post-process

```
Input: A set of DOM nodes $mainContent$, a text threshold $tr$, and a hyperlinks threshold $hr$
Output: A set of DOM nodes $mainContent$ excluding some groups of links
begin
    foreach ($n$ in $mainContent$)
        $m = n - \{hyperlinks(n)\}$
        $n.textRatio = |n.innerText| / |m.innerText|$;
        $n.links = |hyperlinks(n)|$;
        if ($n.textRatio > tr$ and $n.links > hr$)
            $mainContent = mainContent - n$;
        $found = true$;
        $siblings = getSiblings(n, mainContent)$;
        foreach ($i$ in $siblings$)
            if ($i.tagName == n.tagName$)
                if ($|i.children| == 1$ and $i.child.tagName ==$ "$A$")
                    foreach ($child$ in $i.child.children$)
                        if ($child.tagName ==$ "$IMG$")
                            $found = false$;
                else
                    $found = false$;
            else
                $found = false$;
        if ($found == false$)
            $mainContent = mainContent - \{n.parentNode\}$
    return $mainContent$;
end
```

4 EMPIRICAL EVALUATION

This technique has been implemented as a WebExtension, which is compatible with several browsers, such as Google Chrome, Mozilla Firefox, Microsoft Edge, Opera, etc. It has been officially published by Firefox.[2] This addon comes in the form of a single button of the browser. When it is pressed, the plugin extracts the main content of the current webpage, which is automatically displayed[3] (and it can be saved). If it is pressed again, the original webpage is displayed.

The evaluation of our method was done using the Template Detection and Content Extraction Benchmark Suite (TECO).[4] TECO is a suite of real and heterogeneous benchmarks with different layouts and page structures. It has been especially designed to evaluate template and content extractors: The HTML elements (textual information, but also pictures, embedded media, etc.) are labelled so that we can detect the main content with our technique and know its exact precision and recall. In the experiments, we used TECO 4.0, which contains 130 benchmarks. We used 65 benchmarks to evaluate the metrics proposed in Section 3.1, 15 bechmarks to train our algorithm, and 50 benchmarks as the evaluation set.

Most content extraction techniques in the literature use the recall, precision, and F1[5] metrics of the retrieved words. This is somehow limited because it assumes that the main content of the webpage is only text. In our evaluation we overcome this limitation by measuring the retrieved DOM nodes. Therefore, we perform an evaluation that considers that the main content can include text, video, images, animations, and any other contents. Moreover, in order to compare our technique with the related work, we have also evaluated the technique using retrieved words as a metrics. Besides retrieved words, a wide range of different metrics (see, e.g., [4, 7, 31, 34]) are used for the evaluation and comparison of content extraction algorithms. Therefore, to perform a proper comparison of our technique with other mainstream algorithms, we also adopted and implemented some of these metrics. The obtained results are detailed in section 5.

Precision, recall, and F1 evaluation. We built a testing version of our WebExtension to automate the evaluation. For all the webpages of the benchmark suite (TECO), it sequentially executes the content extraction algorithm. For each benchmark, it computes the recall, precision, F1, and the execution time of the retrieved words and the retrieved DOM nodes. First of all, it is necessary to determine the optimal size of the set of candidate nodes (the c value of the c-SET. See Section 3.3). We randomly selected a training subset of 15 benchmarks and computed the recall, precision and F1 of the retrieved DOM nodes and text words for several c-SET sizes.

Table 1 presents the results of the experiments conducted with the training set and with a c-SET size from 1 to 8. Each row shows the average `Recall`, `Precision`, and `F1` of executing the algorithm for all the benchmarks in the training subset with a different value for c in the c-SET. The table contains the average results for both, retrieved text words and retrieved DOM nodes.

The table shows that a n-SET with n=3 produces the best results. 3-SET obtains the best `F1` value in retrieved DOM nodes (82.95%) and the best `F1` value in retrieved words (87.65%). Therefore, even though in our implementation the size of the n-SET is configurable, we use a 3-SET by default. The table also shows that the size of the c-SET can have a considerable impact on the recall and precision. In contrast, it has a very little impact on the performance: the average runtimes are similar for all the tested c-SETs (e.g., the difference between the average runtime of the 1-SET and the average runtime of the 8-SET is only 0.03 seconds).

[2]Firefox runs several rounds of review, after which those addons whose quality is above the Firefox standards become part of the Firefox distribution.
[3]The nodes that do not belong to the main content are properly hidden by changing their *visibility* and *display* attributes to *hidden* or *none*, respectively. Therefore the main content is isolated and it appears in the same place as in the original webpage.
[4]http://personales.upv.es/josilga/retrieval/teco/
[5]Computed as $(2 * P * R)/(P + R)$, where P is the precision and R is the recall.

	DOM nodes			Words			
Size	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**	**Runtime**
1	85,14 %	73,24 %	69,93 %	88,40 %	82,21 %	79,75 %	0,54 s.
2	85,77 %	83,43 %	77,82 %	88,42 %	87,86 %	83,03 %	0,55 s.
3	83,16 %	91,94 %	82,95 %	87,95 %	94,69 %	87,65 %	0,55 s.
4	80,33 %	93,73 %	81,80 %	84,85 %	95,70 %	85,77 %	0,55 s.
5	78,65 %	90,93 %	78,37 %	82,13 %	92,43 %	82,10 %	0,56 s.
6	72,23 %	82,61 %	66,75 %	74,74 %	86,47 %	74,07 %	0,56 s.
7	66,68 %	81,17 %	59,31 %	69,62 %	84,54 %	68,40 %	0,56 s.
8	72,16 %	81,17 %	64,03 %	75,40 %	84,54 %	72,92 %	0,57 s.

Table 1. Determining the optimal size of the c-SET

The table also shows that a bigger c-SET does not necessarily obtain higher F1 values. This happens because increasing the size of the c-SET also increases the probability of selecting one or several nodes that are descendants from the main content root node or nodes. Therefore, as Algorithm 4 has more possible nodes to select, in some benchmarks it can select descendants from the root node or nodes, thus the recall decreases. This can be observed in the dip for $n = 7$. It happens because there are two benchmarks where Algorithm 4 selects different main content nodes for $n = 7$ and for $n = 8$. The selected nodes for $n = 8$ contain more main content nodes than the selected nodes for $n = 7$ in these benchmarks. Therefore, the recall and the F1 are higher for $n = 8$ than for $n = 7$.

We also evaluated the impact of the post-process phase (see Section 3.5). Using a 3-SET, for the training subset of 15 benchmarks, we determined the best values for the `tr` and `hr` thresholds in the following way:

- We tried `tr` values from 1.25 to 3 in steps of 0.25, and `hr` values from 1 to 8.
- For each `tr` value, we selected the best `F1` results for both, DOM nodes and retrieved text words.

Table 2 presents the results of the experiments carried out to determine the `tr` and `hr` thresholds. Each row shows the best combination of results for each `tr` value. We can observe that, for each `tr` value, the best `F1` results are obtained with a `hr` threshold equal to 7 or 8 (both `hr` thresholds obtain the same results). Note that `tr` values higher than 1.5 obtain the same results. The last row of the Table shows the results obtained without applying the post-process phase, which are more than 1% lower than the best results obtained with the application of the post-process.

		DOM nodes			Words		
tr	**hr**	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**
1,25	[7..8]	81,52 %	93,36 %	83,13 %	85,84 %	95,69 %	87,31 %
1,5	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
1,75	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
2	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
2,25	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
2,5	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
2,75	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
3	[7..8]	82,99 %	93,38 %	83,96 %	87,12 %	95,69 %	88,02 %
No post-process		83,16 %	91,94 %	82,95 %	87,95 %	94,69 %	87,65 %

Table 2. Determining tr and hr thresholds

The evaluation subset was formed from 50 benchmarks. For each benchmark, the number of DOM nodes and the number of DOM nodes classified as main content; and the `Recall`, `Precision`, and `F1` of the retrieved DOM nodes and the retrieved words was computed. Additionally, the `Runtime` in seconds was also registered. The average results, computed with a 3-SET, are shown in Table 3.

	Number of nodes		DOM nodes			Words			
Benchmark	**Webpage**	**MainContent**	**Rec.**	**Prec.**	**F1**	**Rec.**	**Prec.**	**F1**	**Runtime**
www.jdi.org.za/	626	199	67,34 %	59,29 %	63,06 %	90,73 %	94,35 %	92,50 %	0,21 s.
www.u-tokyo.ac.jp/	602	97	95,88 %	87,74 %	91,63 %	100,00 %	100,00 %	100,00 %	0,06 s.
www.savethechildren.net/	751	54	68,52 %	100,00 %	81,32 %	99,14 %	100,00 %	99,57 %	0,22 s.
college.harvard.edu/	1090	397	98,74 %	53,99 %	69,81 %	96,04 %	74,05 %	83,62 %	0,31 s.
www.unicef.org/	1052	381	99,21 %	98,95 %	99,08 %	98,33 %	100,00 %	99,16 %	0,22 s.
www.linuxfoundation.org/	588	38	92,11 %	87,50 %	89,75 %	99,40 %	100,00 %	99,70 %	0,30 s.
clinicaltrials.gov/	543	101	88,12 %	76,72 %	82,03 %	91,17 %	96,63 %	93,82 %	0,22 s.
cordis.europa.eu/	959	164	97,56 %	99,38 %	98,46 %	98,62 %	100,00 %	99,31 %	0,31 s.
www.informatik.uni-trier.de/	3085	3021	76,33 %	100,00 %	86,58 %	83,43 %	100,00 %	90,97 %	2,43 s.
parents.berkeley.edu/	283	184	96,20 %	77,29 %	85,71 %	98,19 %	100,00 %	99,09 %	0,10 s.
www.bbc.co.uk/	2991	1360	92,57 %	56,13 %	69,89 %	89,44 %	52,87 %	66,46 %	1,21 s.
techcrunch.com/	2576	586	96,76 %	99,82 %	98,27 %	100,00 %	100,00 %	100,00 %	0,85 s.
www.turfparadise.com/	1057	213	96,24 %	67,66 %	79,46 %	91,24 %	100,00 %	95,642 %	0,39 s.
www.cleanclothes.org/	1335	928	99,68 %	86,94 %	92,88 %	96,83 %	93,30 %	95,03 %	0,31 s.
www.afp.com/	1199	789	99,87 %	99,12 %	99,49 %	100,00 %	100,00 %	100,00 %	0,30 s.
news.discovery.com/	2896	800	20,75 %	100,00 %	34,37 %	78,04 %	100,00 %	87,67 %	1,19 s.
www.history.com/	1246	260	91,15 %	29,37 %	44,43 %	76,92 %	33,51 %	46,68 %	0,44 s.
detroit.cbslocal.com/	1256	98	96,94 %	98,96 %	97,94 %	100,00 %	100,00 %	100,00 %	0,28 s.
www.rocklists.com/	765	184	99,46 %	99,46 %	99,46 %	100,00 %	100,00 %	100,00 %	0,23 s.
www.lashorasperdidas.com/	1822	722	99,86 %	56,86 %	72,46 %	100,00 %	58,36 %	73,71 %	0,87 s.
www.arduino.cc/	830	340	87,06 %	100,00 %	93,08 %	86,20 %	100,00 %	92,59 %	0,14 s.
today.java.net/	698	354	99,15 %	99,72 %	99,43 %	99,96 %	100,00 %	99,98 %	0,22 s.
clotheshor.se/	459	228	45,61 %	100,00 %	62,65 %	44,56 %	100,00 %	61,65 %	0,11 s.
ruzafagallery.com/	439	176	86,36 %	100,00 %	92,68 %	100,00 %	100,00 %	100,00 %	0,10 s.
www.raspberrypi.org/	392	209	96,17 %	99,50 %	97,81 %	94,15 %	100,00 %	96,99 %	0,12 s.
doodle.com/	572	82	56,10 %	100,00 %	71,88 %	96,15 %	100,00 %	98,04 %	0,16 s.
www.newprosoft.com/	832	681	99,85 %	99,85 %	99,85 %	100,00 %	100,00 %	100,00 %	0,29 s.
worryfreelabs.com/	511	190	99,47 %	99,47 %	99,47 %	100,00 %	100,00 %	100,00 %	0,12 s.
www.intelligencetest.com/	592	269	97,40 %	98,50 %	97,95 %	97,35 %	100,00 %	98,66 %	0,21 s.
www.ikea.com/	1545	991	92,18 %	99,90 %	95,88 %	69,87 %	100,00 %	82,26 %	0,55 s.
www.trendencias.com/	2426	1042	82,73 %	99,88 %	90,50 %	71,10 %	100,00 %	83,11 %	1,09 s.
users.dsic.upv.es/~dinsa/	241	160	98,75 %	94,61 %	96,64 %	100,00 %	100,00 %	100,00 %	0,07 s.
googleblog.blogspot.com.es/	5084	1507	99,93 %	99,74 %	99,83 %	100,00 %	100,00 %	100,00 %	2,46 s.
www.robyncarr.com/	292	200	99,50 %	99,50 %	99,50 %	100,00 %	100,00 %	100,00 %	0,10 s.
www.annmalaspina.com/	400	206	6,80 %	100,00 %	12,73 %	71,07 %	100,00 %	83,09 %	0,11 s.
users.dsic.upv.es/~jsilva/	203	34	32,35 %	100,00 %	48,89 %	37,35 %	100,00 %	54,39 %	0,04 s.
foodsense.is/	334	192	95,31 %	82,06 %	88,19 %	96,15 %	89,29 %	92,59 %	0,07 s.
diarium.usal.es/	603	524	99,62 %	99,81 %	99,71 %	99,31 %	100,00 %	99,65 %	0,17 s.
www.folj.com/	559	384	99,74 %	88,66 %	93,87 %	100,00 %	99,46 %	99,73 %	0,05 s.
oneminutelist.com/	490	217	65,44 %	100,00 %	79,11 %	39,00 %	100,00 %	56,12 %	0,12 s.
en.citizendium.org/	1083	633	29,23 %	100,00 %	45,24 %	48,18 %	100,00 %	65,03 %	0,43 s.
www.filmaffinity.com/	1333	976	98,46 %	99,48 %	98,97 %	86,73 %	100,00 %	92,89 %	0,68 s.
stackoverflow.com/	6450	5891	98,47 %	99,97 %	99,21 %	59,62 %	100,00 %	74,70 %	5,67 s.
www.meneame.net/	760	423	96,69 %	100,00 %	98,32 %	97,98 %	100,00 %	98,98 %	0,21 s.
www.strangehorizons.com/	634	403	99,75 %	99,75 %	99,75 %	100,00 %	100,00 %	100,00 %	0,20 s.
www.accountkiller.com/	501	279	48,03 %	100,00 %	64,89 %	74,53 %	100,00 %	85,41 %	0,10 s.
study.com/	7321	5424	99,98 %	99,98 %	99,98 %	100,00 %	100,00 %	100,00 %	8,29 s.
c.mi.com/it/	3490	541	90,94 %	100,00 %	95,26 %	100,00 %	100,00 %	100,00 %	4,09 s.
frances.forosactivos.net/	813	495	99,19 %	78,06 %	87,37 %	97,63 %	69,92 %	81,48 %	0,26 s.
alumni.harvard.edu/	2004	219	69,86 %	100,00 %	82,26 %	92,18 %	100,00 %	95,93 %	1,03 s.
Average	1372,26	676,92	84,87 %	91,47 %	84,54 %	88,93 %	95,23 %	90,32 %	0,75 s.

Table 3. Evaluation of the precision, recall, F1, and runtime

The obtained results show an average F1 of 84.54% for retrieved DOM nodes, and an average F1 of 90.32% for retrieved words.[6] As far as we know, this is the best F1 obtained with these metrics in a page-level heterogeneous technique evaluated with real websites. In particular, other techniques that have been also evaluated using heterogeneous websites obtain the following F1 results for retrieved words: Shanchan et al. 82% [37], Gottron et al. 77% [17], and Insa et al. obtain 74% [19]. There also exist techniques that have been evaluated using prepared datasets (BIG5, MYRIAD40, MSS,

[6]The F1 average is the average of the values in collumn F1, and not the F1 computed with the average precision and the average recall.

etc.), RSS feeds, or prepared websites[7]. Unfortunately, it is not possible to fairly compare precisely different techniques if they use different evaluation datasets or if they know a priori the structure of the website from which they are going to extract the content. For instance, other techniques that evaluated prepared websites reported high F1 values (Adam et al. obtain 93% [1], Zhao et al. 88% [23], Pasternack et al. 95% [26], and Qureshi et al. 94% [28]). But these F1 results are significantly reduced if heterogeneous websites are used.

Runtime evaluation. Figure 2 shows the relationship between the number of rated nodes (see section 3.1) and the time needed to extract the main content from this webpage. Most of the runtime is used by Algorithm 1, which has an asymptotic cost of $O(n^2)$, being n the number of rated nodes.

It can be observed that 82% of the benchmarks took less than 1 second and 74% of them took less than half a second. Only nine benchmarks (all of them very big websites with more than 600 rated nodes) took more than 1 second. Figure 2 shows that, for most webpages, the tool can extract the main content in less than 1 second.

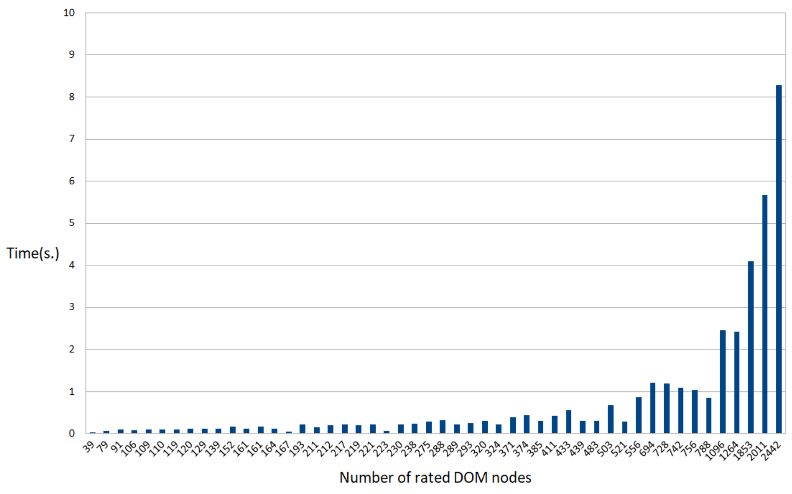

Fig. 2. Relationship between # rated nodes and runtime

5 COMPARISON WITH OTHER TECHNIQUES

We made an extensive evaluation to properly compare our technique with other relevant state of the art techniques. To produce a fair comparative evaluation, we made three different comparisons. Each comparison uses a different dataset (the one originally used by the other techniques). Some of the datasets are composed of heterogeneous websites, while others are sets of webpages collected from the same website. We used the original implementation when it was available, and we implemented it from scratch when the implementation was not public. The benchmarks and all the tools we implemented are publicly available and open-source. They are accesible at http://personales.upv.es/josilga/retrieval/teco/ (benchmarks) and http://personales.upv.es/josilga/retrieval/Web-TemEx/ (tools and experiments). So a side contribution of this work is an open implementation of several state-of-the-art techniques. This collection of techniques implemented in a single workbench can be a reference framework to evaluate and compare any new technique.

[7] Automatically generated webpages that share the same template.

We compared our technique with both page-level and site-level extraction techniques using different specific datasets. The obtained results are explained in the following subsections.

5.1 Comparison with site-level techniques

The comparison of our technique with site-level content extraction algorithms was not possible using the data published by their authors. The dataset used to evaluate each technique was totally different: some of them used real websites, others used artificial benchmarks including randomly generated webpages and webpages with templates known a priori. Some of them used large data sets and others used only a reduced subset of 5 webpages, etc. Moreover, they used different metrics to measure the content retrieved: some of them used characters, others used DOM nodes, and others used words.

The only way to fairly compare our technique with other state-of-the-art techniques is to evaluate all of them with the same dataset, with the same metrics, and in the same context (to properly compare runtimes). We have done it. This part of the work is an important contribution because it provides for the first time a fair comparison (same dataset, same evaluation measures: words and DOM nodes) of various important tools. We selected six well-known site-level algorithms in the literature ([3, 4, 32, 33, 35, 38]). Unfortunately, their implementation was not always available, or it was not public (i.e., proprietary code), so we had to implement some of them from scratch. We have published all of them, so they are open-source and publicly available at: http://personales.upv.es/josilga/retrieval/Web-TemEx/

The results of the accuracy and performance comparison with the evaluation set are shown in Table 4. It can be observed that for both, retrieved DOM nodes and retrieved text words, the best `F1` values are achieved by our algorithm. TemEx is in the second position, with very similar values. Our algorithm also has the best precision of all algorithms. Moreover, because our algorithm is page-level, its performance is quite good compared to most of the other algorithms. For instance, the average runtime of TemEx is 10 times higher than our new algorithm.

	DOM nodes			Words			
Algorithm	Rec.	Prec.	F1	Rec.	Prec.	F1	Runtime
SST (2003) [38]	65,10 %	47,42 %	51,66 %	72,64 %	59,92 %	61,26 %	18,75 s.
RTDM-TD (2006) [33]	99,39 %	51,53 %	63,10 %	99,98 %	67,35 %	77,57 %	7,27 s.
IWPTD (2008) [35]	70,29 %	61,86 %	61,94 %	79,28 %	64,33 %	68,78 %	3,64 s.
RBM-TD (2009) [32]	**100,00** %	52,45 %	65,10 %	**100,00** %	67,84 %	77,84 %	5,87 s.
TemEx (2015) [3]	90,62 %	81,28 %	83,05 %	94,39 %	88,82 %	90,18 %	4,87 s.
ConEx (2018) [4]	83,91 %	88,07 %	80,93 %	86,13 %	90,57 %	83,06 %	8,47 s.
Our algorithm	84,87 %	**91,47** %	**84,54** %	88,93 %	**95,23** %	**90,32** %	0,47 s.

Table 4. Empirical evaluation and comparison with six site-level web content extraction algorithms

To ensure a fair comparison of runtimes, the experiments were done with the same computer, software configuration, and load. All of them are implemented using the same technology (as WebExtensions). To provide more independence to the experiments the first iteration was always discarded, thus the influence of aspects such as the influence of dynamically loaded libraries persisting in physical memory, data persisting in the disk cache, etc. was avoided. For each benchmark and tool, the experiments were repeated until a window of ten executions in a row produced a standard deviation under 10% of the sample average. The statistic value returned is the average of this window.

Except for TemEx and ConEx, the rest of the algorithms obtain sensibly lower `F1` values for both retrieved DOM nodes (between 51% and 65%) and retrieved words (between 61% and 78%). RBM-TD and RTDM-TD are clearly conservative algorithms as they focus on recall, not on precision (RBM-TD achieved 100% recall in all experiments, and RTDM-TD achieved 99.39% average recall). Therefore, when retrieving the maximum amount of main content is critical, RBM-TD and RTDM-TD are the best choices. RBM-TD and RTDM-TD obtain similar F1 values for retrieved DOM nodes and retrieved words. SST obtained the lowest F1 values for both retrieved DOM nodes and retrieved words. TemEx obtained the best performance for site-level algorithms (our algorithm is excluded because it is page-level).

It is important to point out that the F1 value for retrieved words is higher than the F1 value for retrieved DOM nodes in all algorithms. This is a consequence of the fact that these algorithms are more geared towards text retrieval. In fact, some of them use internal metrics based on the amount of words of their DOM nodes or HTML tags.

On the basis of the results obtained, our algorithm should be used if precision or both recall and precision are important. RBM-TD or RTDM-TD should be used for those applications that need to maximize the recall. For applications that need a high performance our algorithm should also be used.

5.2 Comparison with page-level techniques

In contrast to most site-level techniques (see Section 5.1), many page-level content extraction algorithms have been evaluated with publicly available datasets such as Cleaneval [7]. This has facilitated the comparison of our technique with them. Nevertheless, many algorithms use their own metric to measure the content retrieved. Therefore, to fairly compare our technique with the results reported by other techniques, in each case, we used the datasets and metrics proposed by the authors of those techniques.

Firstly, we used the metrics and datasets proposed in [31]. They use 3 publicly available datasets (CleanEval; Big 5, which contains sets of webpages from New York Times, BBC, Yahoo, Ars Technica, and Wikipedia; and Chaos, which contains webpages from Google News, WordPress, and Blogger). We evaluated our algorithm with their evaluation sets and then, we computed the recall, precision, and F1 using their metrics. We compared our algorithm with the algorithms included in [31] plus WLR and CEHTD-DS (see Table 5). Our experiments, compared to theirs, reveal that, for most of the datasets, CECTD-DS obtains the best F1. It also obtains the best average F1, and the best precision and recall for several datasets. In addition, other CETD variants (CETD-DS, CECTD-S and CEHTD-DS) obtain high average F1 values (over 90%). Our algorithm obtains the best recall for one set from the Big 5 dataset (Yahoo). WLR obtains the best precision in almost all the sets and the best overall average precision (98.79%). The rest of algorithms obtain average F1 values between 68% and 86%, except for FE, which obtains an average F1 around 9%.

It is important to highlight that the CETD algorithms are only based on the text of the DOM nodes. Therefore, in contrast to our algorithm, they ignore the images, video, animations, and other media that belong to the main content.

However, the metrics proposed in [31] uses the longest common subsequence algorithm (LCS). If a is the text extracted and b is the text in the gold standard, the precision is computed as the length of the LCS between a and b divided by the length of a, and the recall is computed as the length of the LCS between a and b divided by the length of b. F1 is computed as in Section 4. This means that this metrics (proposed by themselves and used in Table 5) computes the largest subsequence of text that is common to two strings and, hence, the results of Table 5 favor their algorithms because only content of type text is considered.

We also evaluated our algorithm using the metrics proposed in [34] for the CleanEval dataset. Before they compute their metrics, they use a dynamic programming algorithm to find the optimal alignment between the HTML page and the gold standard from CleanEval. Then, they also align the obtained text content with the gold standard from CleanEval. Finally, they compare both aligned texts and, to decide whether a node should be kept or not, they use a heuristics: if 2/3 or the node's content is present in the gold standard at that location, they mark it as "content". In addition to the CleanEval dataset, they also propose another evaluation dataset which contains 148 webpages selected by them from the CleanEval dataset. Table 6 shows the comparison results. Our algorithm obtain the best F1 value for

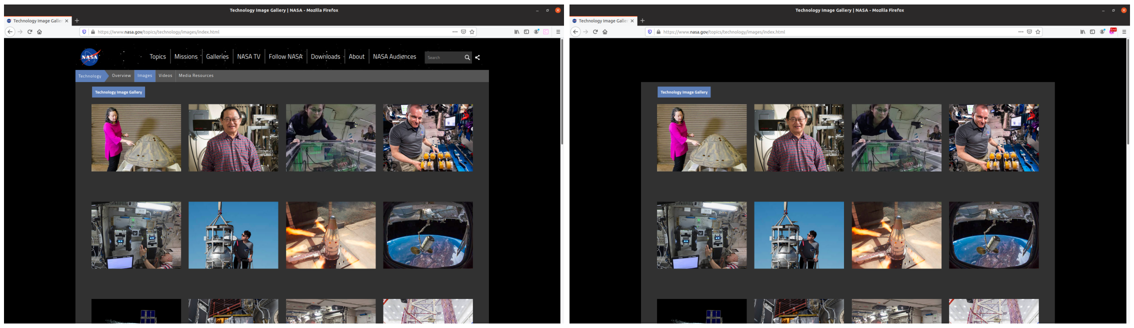

Fig. 3. Image gallery from NASAS's website extracted with our web content extraction tool.

the CleanEval dataset (87%), and the best recall for both datasets, the CleanEval original test dataset and the CleanEval web2text's test dataset (obtaining 90% and 92%, respectively). Web2text obtains the best accuracy for both datasets. Boilerpipe algorithm obtains the best precision for both datasets. Moreover, we downloaded the publicly available implementation of the algorithms proposed in [34] and we measured their performance. Column *Runtime* in table 6 shows significant differences in the runtimes. This is due to the fact that the algorithms are implemented using different technologies, such as Java, Scala, Python, Perl, etc.

It is important to highlight that all the page-level algorithms in this section are focused on text extraction. This fact is evidenced by the metrics used by most researchers to evaluate their algorithms, since they only consider the extracted text. However, as it can be observed in Figure 3, our algorithm extracts the main content of a webpage regardless of its type (it not only extracts text, but also images, animations, etc.). Hence, the metrics used in this section (those proposed by the other techniques) do impair our technique, because they do not consider the non-textual information extracted by our algorithm.

	CleanEval			NYTimes			Yahoo			Wikipedia		
Algorithm	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
BTE (2001) [14]	88,87%	95,83%	92,22%	62,22%	**98,38%**	76,23%	54,94%	95,06%	69,64%	83,91%	81,60%	82,74%
DSC (2002) [27]	91,94%	62,01%	74,07%	98,58%	85,67%	91,68%	96,54%	73,14%	83,23%	81,67%	34,61%	48,62%
FE (2005) [11]	73,87%	9,97%	17,56%	97,51%	3,62%	6,98%	99,08%	4,94%	9,41%	98,79%	1,48%	2,91%
K-FE (2005) [12]	79,28%	69,61%	74,13%	73,82%	71,35%	72,56%	69,49%	56,97%	62,61%	73,76%	44,60%	55,59%
LQF (2005) [25]	88,60%	94,02%	91,23%	90,02%	97,10%	93,42%	64,54%	90,65%	75,40%	83,60%	76,41%	79,85%
CCB (2008) [18]	80,61%	92,71%	86,24%	57,61%	96,09%	72,03%	46,90%	93,45%	62,46%	63,22%	73,14%	67,82%
CETR (2010) [36]	91,26%	86,08%	88,59%	85,19%	90,58%	87,80%	69,36%	77,65%	73,27%	94,69%	72,77%	82,30%
CETD-DS (2011) [31]	92,96%	94,52%	93,73%	98,38%	95,84%	97,09%	83,16%	85,90%	84,51%	98,31%	97,22%	97,77%
CECTD-S (2011) [31]	90,35%	92,60%	91,46%	96,72%	96,56%	96,64%	80,33%	93,34%	86,35%	98,02%	**97,61%**	**97,81%**
CECTD-DS (2011) [31]	95,87%	**97,15%**	**96,51%**	99,69%	98,16%	**98,92%**	84,59%	93,99%	89,04%	98,25%	92,77%	95,43%
WLR (2013) [19]	**96,60%**	65,76%	78,25%	**99,75%**	86,45%	92,62%	**99,66%**	65,28%	78,89%	**99,06%**	82,70%	90,15%
CEHTD-DS (2015) [29]	94,97%	94,07%	94,52%	99,72%	95,96%	97,80%	91,99%	88,59%	**90,26%**	96,58%	90,41%	93,39%
Our algorithm	92,79%	92,35%	92,57%	98,55%	87,68%	92,79%	67,16%	**97,29%**	79,47%	98,90%	94,61%	96,71%

6 RELATED WORK

Web mining is an information retrieval discipline that tries to isolate different functional blocks from a webpage. Therefore, this discipline includes techniques such as content extraction, template extraction, menu detection, etc. There exist different web mining approaches (see, e.g., [9, 18, 19, 36, 37]) and a competition called CleanEval [7] which provided a dataset and a gold standard to score the success of boilerplate detection systems.

	BBC			Ars Technica			Chaos			Average		
Algorithm	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
BTE (2001) [14]	69,09%	97,09%	80,73%	68,25%	97,91%	80,44%	76,36%	92,80%	83,78%	71,95%	94,10%	80,83%
DSC (2002) [27]	89,27%	78,89%	83,76%	95,82%	90,52%	93,09%	94,45%	80,27%	86,79%	92,61%	72,16%	80,17%
FE (2005) [11]	**98,95%**	3,71%	7,15%	0,01%	0,00%	0,00%	72,59%	6,22%	11,46%	77,26%	4,28%	8,97%
K-FE (2005) [12]	63,84%	65,02%	64,43%	81,35%	82,73%	82,03%	73,97%	66,04%	69,78%	73,64%	65,19%	68,73%
LQF (2005) [25]	77,03%	92,17%	83,93%	88,40%	98,43%	93,15%	82,76%	93,98%	88,01%	82,14%	91,82%	86,42%
CCB (2008) [18]	53,52%	92,19%	67,72%	64,05%	96,27%	76,92%	64,45%	91,05%	75,47%	61,48%	90,70%	72,66%
CETR (2010) [36]	68,93%	86,58%	76,76%	83,06%	93,93%	88,16%	78,75%	86,92%	82,63%	81,61%	84,93%	82,78%
CETD-DS (2011) [31]	84,39%	95,21%	89,48%	97,81%	98,85%	98,33%	93,59%	94,99%	93,59%	92,66%	94,65%	93,50%
CECTD-S (2011) [31]	82,55%	93,77%	87,80%	94,61%	93,56%	94,08%	89,64%	91,22%	91,22%	90,33%	93,97%	92,24%
CECTD-DS (2011) [31]	86,15%	**97,95%**	91,67%	98,04%	**99,51%**	**98,77%**	96,21%	96,10%	**96,15%**	94,11%	**96,52%**	**95,21%**
WLR (2013) [19]	98,45%	66,97%	79,71%	**99,98%**	94,40%	97,12%	**98,03%**	70,05%	81,71%	**98,79%**	75,94%	85,49%
CEHTD-DS (2015) [29]	95,55%	96,46%	**96,00%**	98,12%	98,83%	98,48%	94,74%	**96,42%**	95,57%	95,95%	94,39%	95,15%
Our algorithm	93,15%	91,42%	92,28%	97,81%	97,03%	97,42%	95,01%	91,88%	93,42%	91,91%	93,18%	92,09%

Table 5. Empirical evaluation with CETD's metrics

	Cleaneval test				Web2text's test				
Method	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Runtime
BTE (2001) [14]	79%	79%	89%	83%	75%	76%	84%	80%	0,08 s.
CRF (2008) [30]	82%	87%	81%	84%	82%	88%	81%	84%	0,13 s.
Default-ext (2010) [21]	80%	89%	75%	81%	79%	89%	74%	81%	0,05 s.
Article-ext (2010) [21]	72%	**91%**	59%	71%	67%	89%	50%	64%	0,05 s.
Largest-ext (2010) [21]	60%	83%	36%	52%	59%	**93%**	33%	48%	0,05 s.
Unfluff (2014) [15]	71%	90%	57%	70%	68%	90%	51%	65%	0,52 s.
Web2text (2018) [34]	**84%**	88%	85%	86%	**86%**	87%	90%	**88%**	0,05 s.
Our algorithm	83%	83%	**90%**	**87%**	84%	83%	**92%**	87%	0,27 s.

Table 6. Empirical evaluation with Web2text's metrics

Bar-Youssef et al. [6] defined a pagelet as *a self-contained logical region within a page that has a well-defined topic or functionality*. While *content extraction* deals with the detection and isolation of the main content pagelets of the webpage, *template extraction* deals with the isolation of the template. Therefore, both techniques are closely related because they are almost complementary.

Block detection techniques can be further classified depending on the way in which they internally represent the webpages:

(i) HTML-based approaches use the textual information. Many of them assume that the main content on a webpage contains a high text density and a low tag density. Namely, Ferraresi et al. [13] analyze the HTML code and define the main content as the largest continuous text area with the fewer amount of HTML tags. Kohlschütter et al. [20] examine a small set of shallow text features to classify the text elements of a webpage. Weninger et al. [36] developed the CETR algorithm (Content Extraction via Tag Ratios), which computes the CETR ratio by analyzing the HTML code and, for each tag, counting its number of characters and tags.

(ii) Another approach is to use a rendered image of the webpage on the browser. These techniques (e.g., Burget et al. [8]) are based on the assumption that the main content of a webpage is usually placed in its mid-section, and all or part of it is visible to the user. The main drawback of this kind of techniques is that rendering webpages for classification is a computational expensive operation (Kohlschütter et al. [22]).

(iii) Currently, the most extended approach is to use the representation of a webpage as a DOM tree. This third approach is where our technique falls. In 2002, Bar-Yossef et al. [6] introduced a method that infers information from the webpage's DOM tree and computes the frequent pagelet sets. Yi et al. [38], Vieira et al. [33] and Alarte et al. [3]

also proposed template detection techniques that use the DOM tree representation of the webpage. Roughly, these techniques identify the template by finding common DOM subtrees in different webpages of a website.
In particular, Yi et al. [38] proposed a new data structure that summarizes information from various DOM trees, called Site Style Tree (SST). Authors assume that the most repeated nodes in the SST are template nodes. Sun et al. [31] proposed a general method for extracting content from diverse webpages. It introduces two metrics that measure the importance of nodes: Text Density and Composite Text Density. Insa et al. [19] used a similar notion of density that, for each DOM node, computes a ratio between the number of words and leaves in its subtree. Then, among the nodes with higher density of text, they identify the main content. In [31], the approach is based on computing the ratio between the amount of chars and tags in the subtree of a node.

7 CONCLUSIONS

This paper presents a novel technique for content extraction from heterogeneous webpages. The main novelty of our technique is the new metric proposed, the representation of the DOM node features as points in a 4-dimensional Euclidean space $\mathbb{R}^4$, and the way in which the main content is detected in such a space. The features considered in the metrics and their ranked values have proven to be useful to isolate the containers of the main content nodes. For this reason, those DOM nodes whose features are clearly different to most of the nodes probably contain the main content information. The representation of the DOM node features as points in $\mathbb{R}^4$ is helpful to easily and quickly identify the candidate nodes by means of the standard Euclidean distance between the points.

We have carried out an extensive evaluation with over 10000 experiments and with different data sets used in other techniques. The evaluation and comparison of our tool with other state-of-the-art tools revealed that it achieves the best results in various datasets. We first compared it with well-known site-level techniques, which we had to implement from scratch (they are now open-source and publicly available). Then, we compared it with other well-known page-level techniques using different datasets and metrics. As a result of the comparison (from the best of our knowledge, the one with more datasets, metrics, and tools), we have provided a general view of the weak and strong points of each technique, identifying the best tool for different possible scenarios. One strong point of our technique is that, contrarily to most techniques, which are only focused on text extraction, it is able to extract the main content regardless of its type. That is, it not only extracts text, but also animations, images, etc. Our implementation is open and free (it is being distributed as an official Firefox addon).

8 ACKNOWLEDGEMENTS

This work has been partially supported by the EU (FEDER) and the Spanish MCI/AEI under grants TIN2016-76843-C4-1-R and PID2019-104735RB-C41, by the *Generalitat Valenciana* under grant Prometeo/2019/098 (DeepTrust), and by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under GA No 952215.

REFERENCES

[1] G. Adam, C. Bouras, and V. Poulopoulos. 2009. CUTER: An Efficient Useful Text Extraction Mechanism. In *2009 International Conference on Advanced Information Networking and Applications Workshops*. 703–708. https://doi.org/10.1109/WAINA.2009.60

[2] Julian Alarte, David Insa, and Josep Silva. 2017. Webpage Menu Detection Based on DOM. In *Proceedings of the International Conference on Current Trends in Theory and Practice of Informatics(SOFSEM 2017) (Lecture Notes in Computer Science (LNCS), Vol. 10139)*. Springer, 411–422.

[3] Julián Alarte, David Insa, Josep Silva, and Salvador Tamarit. 2015. Site-Level Web Template Extraction Based on DOM Analysis. In *Perspectives of System Informatics - 10th International Andrei Ershov Informatics Conference, PSI 2015, in Memory of Helmut Veith, Kazan and Innopolis, Russia, August 24-27, 2015, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 9609)*, Manuel Mazzara and Andrei Voronkov (Eds.). Springer, 36–49. https://doi.org/10.1007/978-3-319-41579-6_4

[4] Julian Alarte, David Insa, Josep Silva, and Salvador Tamarit. 2018. Main Content Extraction from Heterogeneous Webpages. In *Web Information Systems Engineering – WISE 2018*, Hakim Hacid, Wojciech Cellary, Hua Wang, Hye-Young Paik, and Rui Zhou (Eds.). Springer International Publishing, Cham, 393–407.

[5] Mohsen Asfia, Mir Mohsen Pedram, and Amir Masoud Rahmani. 2010. Main content extraction from detailed web pages. *International Journal of Computer Applications* 4, 11 (2010), 18–21.

[6] Ziv Bar-Yossef and Sridhar Rajagopalan. 2002. Template detection via data mining and its applications. In *Proceedings of the 11th International Conference on World Wide Web (WWW'02)* (Honolulu, Hawaii, USA). ACM, New York, NY, USA, 580–591. https://doi.org/10.1145/511446.511522

[7] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. 2008. Cleaneval: a Competition for Cleaning Web Pages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC'08)* (Marrakech, Morocco). European Language Resources Association, 638–643.

[8] Radek Burget and Ivana Rudolfova. 2009. Web Page Element Classification Based on Visual Features. In *Proceedings of the 1st Asian Conference on Intelligent Information and Database Systems (ACIIDS'09)*. IEEE Computer Society, Washington, DC, USA, 67–72. https://doi.org/10.1109/ACIIDS.2009.71

[9] Eduardo Cardoso, Iam Jabour, Eduardo Laber, Rogério Rodrigues, and Pedro Cardoso. 2011. An efficient language-independent method to extract content from news webpages. In *Proceedings of the 11th ACM symposium on Document Engineering (DocEng'11)* (Mountain View, California, USA). ACM, New York, NY, USA, 121–128. https://doi.org/10.1145/2034691.2034720

[10] W3C Consortium. 1997. Document Object Model (DOM). Available from URL: http://www.w3.org/DOM/.

[11] Sandip Debnath, Prasenjit Mitra, and C Lee Giles. 2005. Automatic extraction of informative blocks from webpages. In *Proceedings of the 2005 ACM symposium on Applied computing*. ACM, 1722–1726.

[12] Sandip Debnath, Prasenjit Mitra, and C Lee Giles. 2005. Identifying content blocks from web documents. In *International Symposium on Methodologies for Intelligent Systems*. Springer, 285–293.

[13] Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of english. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4)*. 47–54.

[14] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. 2001. Fact or Fiction: Content Classification for Digital Libraries. In *DELOS Workshop: Personalisation and Recommender Systems in Digital Libraries* (Dublin (Ireland)). 1–6. http://www.ercim.org/publication/ws-proceedings/DelNoe02/AidanFinn.pdf

[15] Albert Geitgey. 2014. Unfluff - an automatic web page content extractor for node.js! (2014).

[16] David Gibson, Kunal Punera, and Andrew Tomkins. 2005. The volume and evolution of web page templates. In *Proceedings of the 14th International Conference on World Wide Web (WWW'05)* (Chiba (Japan)), Allan Ellis and Tatsuya Hagino (Eds.). ACM, 830–839. https://doi.org/10.1145/1062745.1062763

[17] Thomas Gottron. 2008. Content Code Blurring: A New Approach to Content Extraction. In *Proceedings of the 2008 19th International Conference on Database and Expert Systems Application (DEXA '08)*. IEEE Computer Society, Washington, DC, USA, 29–33. https://doi.org/10.1109/DEXA.2008.43

[18] Thomas Gottron. 2008. Content Code Blurring: A New Approach to Content Extraction. In *Proceedings of the 19th International Workshop on Database and Expert Systems Applications (DEXA'08)* (Turin (Italy)), A. Min Tjoa and Roland R. Wagner (Eds.). IEEE Computer Society, 29–33. https://doi.org/10.1109/DEXA.2008.43

[19] David Insa, Josep Silva, and Salvador Tamarit. 2013. Using the words/leafs ratio in the DOM tree for content extraction. *The Journal of Logic and Algebraic Programming* 82, 8 (2013), 311–325. https://doi.org/10.1016/j.jlap.2013.01.002

[20] Christian Kohlschütter. 2009. A densitometric analysis of web template content. In *Proceedings of the 18th International Conference on World Wide Web (WWW'09)* (Madrid (Spain)), Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl (Eds.). ACM, 1165–1166. https://doi.org/10.1145/1526709.1526909

[21] Christian Kohlschütter et al. 2010. Boilerpipe–boilerplate removal and fulltext extraction from HTML pages. *Google Code* (2010).

[22] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the 3th International Conference on Web Search and Web Data Mining (WSDM'10)* (New York (New York / USA)), Brian D. Davison, Torsten Suel, Nick Craswell, and Bing Liu (Eds.). ACM, 441–450. https://doi.org/10.1145/1718487.1718542

[23] Zhao Li, Wee Keong Ng, and Aixin Sun. 2005. Web data extraction based on structural similarity. *Knowledge and Information Systems* 8, 4 (01 Nov 2005), 438–461. https://doi.org/10.1007/s10115-004-0188-z

[24] Bing Liu. 2006. *Web Data Mining: Exploring Hyperlinks, Contents, and Usage Data (Data-Centric Systems and Applications)*. Springer-Verlag, Berlin, Heidelberg.

[25] Constantine Mantratzis, Mehmet Orgun, and Steve Cassidy. 2005. Separating XHTML content from navigation clutter using DOM-structure block analysis. In *Proceedings of the sixteenth ACM conference on Hypertext and hypermedia*. ACM, 145–147.

[26] Jeff Pasternack and Dan Roth. 2009. Extracting Article Text from the Web with Maximum Subsequence Segmentation. In *Proceedings of the 18th International Conference on World Wide Web* (Madrid, Spain) *(WWW '09)*. ACM, New York, NY, USA, 971–980. https://doi.org/10.1145/1526709.1526840

[27] David Pinto, Michael Branstein, Ryan Coleman, W Bruce Croft, Matthew King, Wei Li, and Xing Wei. 2002. QuASM: a system for question answering using semi-structured data. In *Proceedings of the 2nd ACM/IEEE-CS joint conference on Digital libraries*. ACM, 46–55.

[28] Pir Abdul Rasool Qureshi and Nasrullah Memon. 2012. Hybrid Model of Content Extraction. *J. Comput. Syst. Sci.* 78, 4 (July 2012), 1248–1257. https://doi.org/10.1016/j.jcss.2011.10.012

[29] Dandan Song, Fei Sun, and Lejian Liao. 2015. A hybrid approach for content extraction with text density and visual importance of DOM nodes. *Knowledge and Information Systems* 42, 1 (01 Jan 2015), 75–96. https://doi.org/10.1007/s10115-013-0687-x

[30] Miroslav Spousta, Michal Marek, and Pavel Pecina. 2008. Victor: the web-page cleaning tool. In *4th Web as Corpus Workshop (WAC4)-Can we beat Google*. 12–17.

[31] Fei Sun, Dandan Song, and Lejian Liao. 2011. DOM Based Content Extraction via Text Density. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. ACM, New York, NY, USA, 245–254. https://doi.org/10.1145/2009916.2009952

[32] Karane Vieira, André Luiz da Costa Carvalho, Klessius Berlt, Edleno S. de Moura, Altigran S. da Silva, and Juliana Freire. 2009. On Finding Templates on Web Collections. *World Wide Web* 12, 2 (2009), 171–211. https://doi.org/10.1007/s11280-009-0059-3

[33] Karane Vieira, Altigran S. da Silva, Nick Pinto, Edleno S. de Moura, João M. B. Cavalcanti, and Juliana Freire. 2006. A fast and robust method for web page template detection and removal. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM'06)* (Arlington, Virginia, USA). ACM, New York, NY, USA, 258–267. https://doi.org/10.1145/1183614.1183654

[34] Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. 2018. Web2Text: Deep Structured Boilerplate Removal. In *European Conference on Information Retrieval*. Springer International Publishing, 167–179.

[35] Yu Wang, Bingxing Fang, Xueqi Cheng, Li Guo, and Hongbo Xu. 2008. Incremental Web Page Template Detection. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)* (Beijing, China). ACM, New York, NY, USA, 1247–1248. https://doi.org/10.1145/1367497.1367749

[36] Tim Weninger, William Henry Hsu, and Jiawei Han. 2010. CETR: Content Extraction via Tag Ratios. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)* (Raleigh (North Carolina / USA)), Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti (Eds.). ACM, 971–980. https://doi.org/10.1145/1772690.1772789

[37] Shanchan Wu, Jerry Liu, and Jian Fan. 2015. Automatic Web Content Extraction by Combination of Learning and Grouping. In *Proceedings of the 24th International Conference on World Wide Web* (Florence, Italy) *(WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 1264–1274. https://doi.org/10.1145/2736277.2741659

[38] Lan Yi, Bing Liu, and Xiaoli Li. 2003. Eliminating noisy information in Web pages for data mining. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data mining (KDD'03)* (Washington, D.C.). ACM, New York, NY, USA, 296–305. https://doi.org/10.1145/956750.956785