IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 21 November 2022, accepted 5 December 2022, date of publication 14 December 2022,
date of current version 19 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3229080

RESEARCH ARTICLE

Extracting the Main Content of Web Pages Using the First Impression Area

GEUNSEONG JUNG[1], SUNGJAE HAN[2], HANSUNG KIM[3], KWANGUK KIM[1], AND JAEHYUK CHA[1], (Member, IEEE)

[1]Department of Computer Science, Hanyang University, Seoul 04763, South Korea
[2]JEI Group, Seoul 03076, South Korea
[3]Department of Sociology, Hanyang University, Seoul 04763, South Korea

Corresponding author: Jaehyuk Cha (chajh@hanyang.ac.kr)

This work was supported by the National Research Foundation of Korea (NRF) Grant funded by the Korea Government [Ministry of Science and ICT (MSIT)] under Grant 2016R1A2B4016591 and Grant 2018R1A5A7059549.

ABSTRACT Extracting the main content from a web page is essential in various applications such as web crawlers and browser reader modes. Existing extraction methods using text-based algorithms and features for English text can be ineffective for non-English web pages. This study proposes a main content extraction method that obtains visual and structural features from the rendered web page. Our method uses the first impression area (FIA), a part of a web page that users initially view. In this area, websites have applied many techniques that enable users to find the main content easily. Using the non-textual properties in the FIA, our method selects three points with high content area density and expands the area from each point until it meets several structural and visual-based conditions. We evaluated our method, browsers' (Mozilla Firefox and Google Chrome) reader modes, and existing main content extraction methods on multilingual datasets using two measures: Longest Common Subsequences and matched text blocks. The results showed that our method performed better than other methods in both English (up to 46%, matched text blocks $\mathbf{F_{0.5}}$) and non-English (up to 42%, matched text blocks $\mathbf{F_{0.5}}$) web pages.

INDEX TERMS Boilerplate removal, main content extraction, web content extraction, web mining, web segmentation, block detection.

I. INTRODUCTION

Currently, web pages often provide relevant information with irrelevant content, such as advertisements, banners, and other CMS boilerplate. Extracting relevant content on a web page is often referred to as main content extraction [1], [2], [3], [4]. Many methods have been conducted for isolating the main content depending on the user's needs and its use in web crawlers, indexers, and user applications such as browser reader modes [5], [6].

However, most methods do not consider non-English web pages. For instance, we found that the reader modes of two major browsers (Firefox Readability.js and Google Chrome DOM Distiller) did not work on non-English web pages (Fig. 1). Although English and non-English news pages have similar subjects and layouts, the reader modes were not activated. There are several studies tried to resolve this problem by creating a new algorithm or a model for the local language [7], [8], [9]. However, it is difficult to create new methods for all languages owing to the shortage of developers fluent in each language, especially low-resource languages.

The proposed approach excludes linguistic features and determines the location of the main content through visual appearance and HTML structure using Document Object Model (DOM), especially on the first screen. Owing to the human visual system, people do not read a web page sequentially, but look around the screen and identify the information unconsciously [10]. This process of "seeing" takes place in less than one second; therefore, websites designers must ensure their web page has an attractive and familiar view for a good first impressions [11], [12], [13].

The associate editor coordinating the review of this manuscript and approving it for publication was Jerry Chun-Wei Lin.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

FIGURE 1. **Browser reader modes' results on pages with similar layout and subject but different languages.**

For instance, it can be seen that the web pages in Fig. 1 are news pages, even if users do not know the languages.

Several methods have treated web pages as sequences of text [14], [15], [16], [17]. They uses various text-based features such as word count by whitespace, stop words, and punctuations, which are difficult to apply to some non-English languages. For example, Chinese and Japanese do not use whitespace to divide words. Korean uses whitespaces, but not in the same way as English (e.g., Korean postpositions). Chinese and Korean have no letter cases, whereas the Japanese Kana system has Hiragana and Katakana, one sound to two different letters.

Another main content extraction approach is by using rule-based algorithms and heuristics with structural and visual features via HTML and CSS. This approach has been adopted in the reader mode of modern browsers and has been analyzed by many studies [18], [19], [20], [21]. However, most of these studies are based on design patterns or web page characteristics that were popular at the time. For instance, [22] although the proposed heuristics use the resolution of desktop monitors and PDAs at the time, recent PC and mobile screens are significantly larger compared to the past. Therefore, this approach is desirable when to the rules need to be updated continuously (e.g., Reader mode of well-known browsers) or when a topic-specific extraction is needed.

We address a main content extraction method for English and non-English web pages. We assumed that important content would have been deliberately placed in a visible place, regardless of the language. Since humans read with the fovea, a small area at the center of the retina, people begin their visual search in the screen center [23], [24]. In fact, most main contents span the area between the screen and the document center. Therefore, the proposed method can retrieve the main content using new positional features between elements and the screen center.

This study proposes the three steps of the grid, centering, and expanding (GCE) method for extracting the main content from a web page by searching the main content near the first screen rather than seeking all HTML elements on the web page. Furthermore, it defines the first impressions area (FIA) from the first screen and identifies the candidate elements of the main content in the FIA. (1) In the grid step, we overlay a grid on the web page. The top part of the grid represents the FIA. (2) In the centering step, we examine the following three areas: the browser window (equal to the screen area in full-screen mode), web page document, and FIA. Because the centers of these areas are close or overlapped with the main content, the nearest text node is likely to be a descendant leaf node of the main content wrapper. (3) In the expanding step, we expand the DOM tree from the leaf nodes of the centering step to the relevant main content wrapper using the basic

properties of HTML elements and visual features, such as the width, height, density, HTML tag name, HTML attributes, and distance from the centers. The values of these properties can be calculated from HTML and modern web browser API and require lower computing costs compared to machine- and deep-learning approaches.

We focused on our method to help end-users (mainly non-English) who prefer web browsers to access the Web rather than expert users who are familiar with professional tools or programming languages. Therefore, we implemented our method to a web extension and assessed it with representative browser reader modes and well-known main content extraction methods. Furthermore, we created new datasets, including seven non-English regions.

This method has the following contributions:

- New language-independent features to reduce the number of main content candidates from the first screen of the web page.
- A new method for extracting the main content without any language-dependent features.

The remainder of this paper is organized as follows. Section II presents the background of this work. Section III describes the proposed GCE method. Section IV describes the construction new datasets, comparative experiments, and experimental results. Finally, Section V presents the discussion and conclusions.

II. BACKGROUND

A. HUMAN VISUAL PERCEPTION AND THE FIRST IMPRESSIONS IN WEB PAGES

Although the modern web can handle various types of data, it is generally designed for visual interaction [1]. Owing to the expansion of web-based social media, the web has changed into a more interactive space. It has been known that a user's feelings for web pages is deeply related to their behavior on the web page, which is being studied in security, crowdsourcing and marketing [5], [25], [26], [27]. 'First impressions' is one of the most potent factors in human's visual perception of web pages. A user's positive first impressions in terms of aesthetics or visual appeal is known to influence factors such as trust, reliability, and usability [28], [29], [30], [31]. Therefore, websites design their web pages in a more beautiful and familiar layout to attract users.

The process of forming visual perceptions varies depending on the language or region. For example, English users tend to ignore the ads on the right of the web page, whereas Arabic users ignore those on the left [32]. [33] showed different web browsing patterns among Americans, Chinese, and Koreans by analyzing the area of interest (AOI) with eye-tracking. However, this study found that users focused more on the middle AOIs. Although the impact of languages or regions on visual perception needs to be further studied, universal characteristics in first impressions particularly exist [34]. In particular, several eye-tracking studies have shown that users spend more time near the center of a web page [23], [24]. Given the ergonomic aspect of web page design, placing primary information outside of the page is unnatural because people rely heavily on foveal vision when reading [10], [35]. Therefore, the middle area of the first screen is where the user's first gaze naturally falls, and hence, is the best place to deliver the website's main content.

B. FEATURES OF THE MAIN CONTENT EXTRACTION AND RECENT METHODS

Web page is designed based on visual interactions with users rather than delivering text data [1]. It does not simply list the information and is decorated for users to see easily. Owing to the advancement of web technology, modern websites can organize various user interactions, resulting in modern web pages with a mixture of text and chunks of code. These codes depict how the content is delivered to users, but does not contain its meaning. Since web pages do not explicitly specify their main content, we need to speculate it with other information. Generally, text, structural, and vision-based features have been used to separate the main content from the web page, as well as influence the reader mode of modern browsers.

1) FEATURES

Text-based features are characteristics that appear in the letters and words on the web page. In this case, extraction methods treat the web page as a single text or a list of text and calculated text-based values such as the number of words, text density, letter cases, and text frequency [14], [36]. This approach is preferred to machine learning or deep learning considering any web page can be normalized into a word or sentence sequence [15], [16]. However, text-based features can be affected by the character and writing systems of the language.

Structural-based features originate from HTML and Document object model (DOM). Many approaches utilize the natural characteristics of HTML tags and the relationship between nodes that appear in the DOM tree. For example, list tags such as <UL>, <OL>, and <DL> imply that the child elements will be listed in same level. The content can be divided horizontally with <HR> tag. These features were used to distinguish areas on the web page rather than extracting the main content directly [3], [18], [37]. Considering some tags give a implicit information about the content, recent methods use these features to preprocess the web page. However, the HTML tag's semantic and usage can be changed by the trend of web development, such as front-end frameworks.

Vision-based features use both DOM and visual presentation of the page. In general, modern web pages use Cascading style sheets (CSS) for visual representation. Data about the appearance, such as position, font, colors, and size of the content can be obtained from the DOM and CSS. Vision-based extraction methods extract the main content by representing web pages as an abstract set of visible blocks

and identifying visual separators that distinguish each block cluster [18], [38], [39], [40]. Most approaches using visual features focus on style properties between HTML elements. We address new visual features between an element and the browser screen that the user sees.

2) EXISTING METHODS

Numerous methods have been proposed for extracting the main content for different purposes and groups of users [1], such as computer systems such as indexers and web crawlers, and user applications such as browser reader modes. This section briefly describes the various representative extraction methods for the experiment of this paper (Table 1). The browser reader modes are not the latest. However, they are good comparisons about performance on non-English web pages because major browser communities currently maintain them for commercial use to global users.

Readability.js[1] is a typical rule-based method used by the reader mode of Mozilla Firefox. It examines HTML elements by their tag name, text count, and density of links, along with a text pattern that meets the criteria for the main content. Additionally, it uses ad hoc rules to check the custom tags of well-known websites. Readability.js works well on many famous websites.

DOM Distiller,[2] the reader mode of Google Chrome, is a hybrid method that uses the Boilerpipe classifier [14] with additional rules, similar to Readability.js. Boilerpipe uses a decision tree and linear SVM with the shallow text features of continuous text blocks. DOM Distiller and Boilerpipe focus on the changes in the text features, such as the number of words, text-and-link density, uppercase starting letters, and the structural characteristics of HTML, of consecutive text blocks.

BoilerNet [16] is also a DNN-based classifier that uses LSTM to consider the text node of a web page as a sequence of text blocks whose vectors are words and the path from the root of the DOM tree. It provides a Chromium WebExtension that evaluates the trained model on live web pages. However, a deep-learning model inference is still a heavy and unusual task on web browsers, and hence, older or cheaper computers take a very long time to run the model on the browser.

Web2Text [15] is a DNN-based classifier that uses a CNN model with 128 structural and text-based features, such as the word count, the presence of punctuation, and the number of stop words to decide if each text block is part of the main content, from continuous text blocks. However, unlike English, the word count cannot be calculated by spaces in Chinese and Japanese considering they do not use spaces in sentences. Similarly, punctuation marks such as the comma (,) and stop (.) are not used for many languages, including Chinese, Japanese, and Arabic; these languages have own punctuation marks. Moreover, the stop words are entirely language-dependent features.

[1]https://github.com/mozilla/readability

[2]https://chromium.googlesource.com/chromium/dom-distiller/

Considering that many features of Web2Text and BoilerNet are from linguistic characteristics, they may not work correctly in non-English texts. The visual-based approach, which uses features such as layout, size, and color for visual representation, can better deal with these linguistic dependencies. VIPS [18] extracts web content structure with visual features by segmenting a web page into blocks, the semantic part based on human perception, with 13 heuristic rules about the structural and style features of the DOM node. Furthermore, instead of words or context, it uses text features as visual representation such as font size and visibility of text nodes. For the latest web pages, the rules of VIPS are not available owing to changes in the web environment. For example, table nodes such as <TABLE>, <TR>, and <TD>, which VIPS treated as basic web page layout components, are no longer used for web page layout. Nevertheless, the visual-based approach is suitable for extracting main content in the non-English web pages considering its rules for features are not affected by the language of the web page.

III. PROPOSED METHOD

The proposed main content extraction algorithm retrieves DOM nodes in the newly defined virtual area, which is the first impressions area (FIA). This area is used for reproducing the user's first impressions of the web page. We defined FIA that includes the first screen and some areas below it. According to this definition, this area represents a part of web page that users see shortly after the web page has loaded. We selected candidate nodes in the FIA and determined the final main content with rule-based algorithms and heuristics similar to the modern reader modes used.

We provide some definitions for the FIA and the proposed method (Fig. 2):

Definition 1 (Areas of Browser Window and Web Document): The browser window area is the part where the web page is displayed in the browser application on the screen with a window size of $S_w = (w_0, h_0)$. The web document area is the visible part rendered by a browser application with a document size of $S_d = (w_1, h_1)$.

Definition 1.1 (The Midpoint of the Browser Window Center and the Web Document Center): The midpoint is defined as the point between the browser window center $C_w = (\frac{w_0}{2}, \frac{h_0}{2})$ and the web document center $C_d = (\frac{w_1}{2}, \frac{h_1}{2})$. i.e., the midpoint C_m is $(\frac{w_0+w_1}{4}, \frac{h_0+h_1}{4})$.

Definition 2 (First Impressions Area): The First impressions area is the visible part of a web document that the user sees first. If $h_0 < h_1$, its size becomes $(w_0, \min(\alpha h_0, h_1))$ with the scrolling threshold α. Otherwise, its size is equal to S_w. The value α is for a user's instinctive browsing behavior, such as a unconscious mouse wheel scrolling on the first view.

Definition 3 (Extraction Band): The extraction band is presumed to be relatively more noticeable to the user within the web document. It is the rectangular area between two points C_w and C_d as part of the First impressions area.

In our dataset, the midpoint of Definition 1.1 hit the main content in only 80% of web pages, whereas the main content

TABLE 1. Methods comparison.

Name	Methodology	Highlights	Limitations
Readability.js[1]	Rule-based algorithms for HTML elements containing readable content	Fast and simple implementation for generating a browser reader mode. Maintained by major browser community.	Occasionally ignores non-English web pages.
DOM Distiller[2] (Boilerpipe [14])	Text-based features model and rule-based algorithms.	Good performance. Maintained by major browser community.	Using several English-only text features.
BoilerNet [16]	Deep learning (LSTM) with Text and HTML Tag features	Does not rely on any hand-crafted features	Its word embedding approach works only for languages using whitespaces for a word divider.
Web2Text [15]	Hidden Markov model, Convolution neural network	Combinations of the machine-learning approach with DOM-based features	A large number of features (128), including many English-only ones.

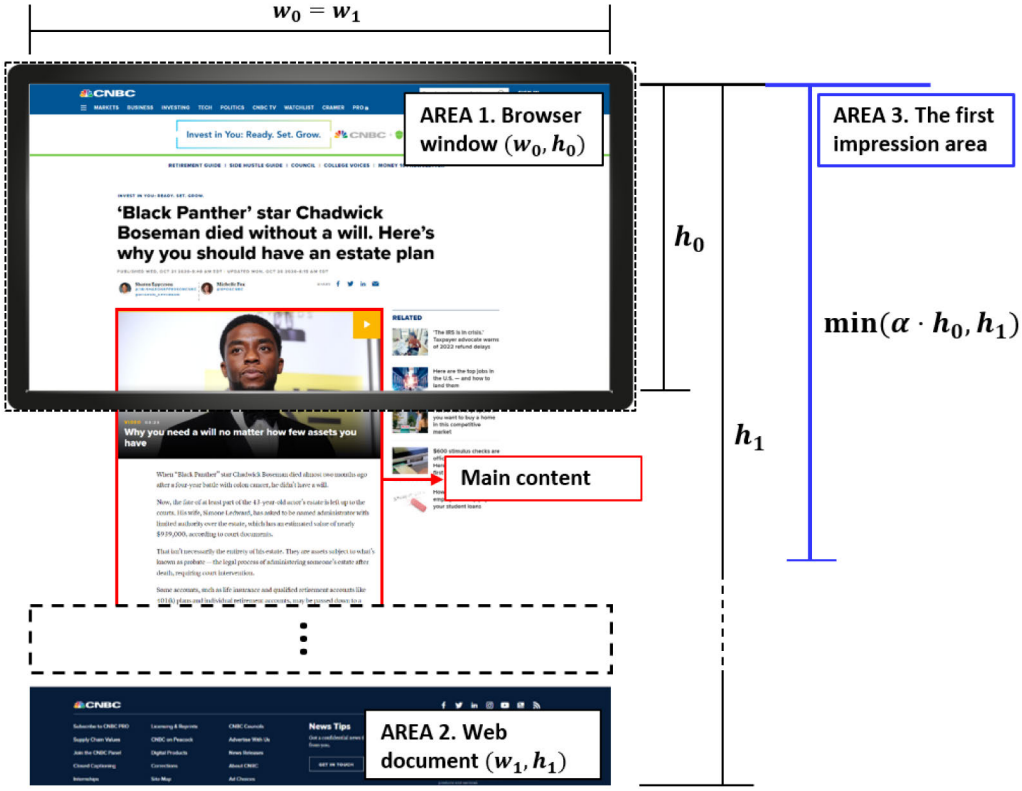

FIGURE 2. Example of the first impressions area in a web page ($h_0 < h_1$).

and extraction band overlapped in 98% of the web pages. We found the more relevant points in the extraction band than the midpoint through the grid and centering steps.

In the grid step, we devised a grid-style First impressions area, called the FIA grid. We overlaid a checkerboard-shape grid on the web document. By the definition of the First impressions area, the initial size of the FIA grid was S_w, which could be extended by adding as many rows as the α value. After the FIA grid size was set, some cells that were expected to be irrelevant from the main content were excluded from the calculation centers in the next step.

In the centering step, we picked points that were likely to be close to the main content with the FIA grid, browser window, and web document area. We determined three center points $C = (C_1, C_2, C_3)$. Considering the FIA grid had cells relatively close to the main content, we assumed its center was one of the most relevant points with the main content. C_1 is the centroid of the FIA grid. C_2 and C_3 are the weighted points of C_1 with the browser window center C_w and the web document center C_d (Fig. 3). We explain how to compute C_2 and C_3 in detail in Section 3.2. In most cases, since these points were in the extraction band and the First impressions area, they were likely to be in or near the main content than the midpoint. In the next step, we retrieved the main content in the DOM tree with the closest leaf node from each point.

In the expanding step, we extracted the main content by traversing the DOM tree with several rules. Because it would be complicated to make a single rule for various languages

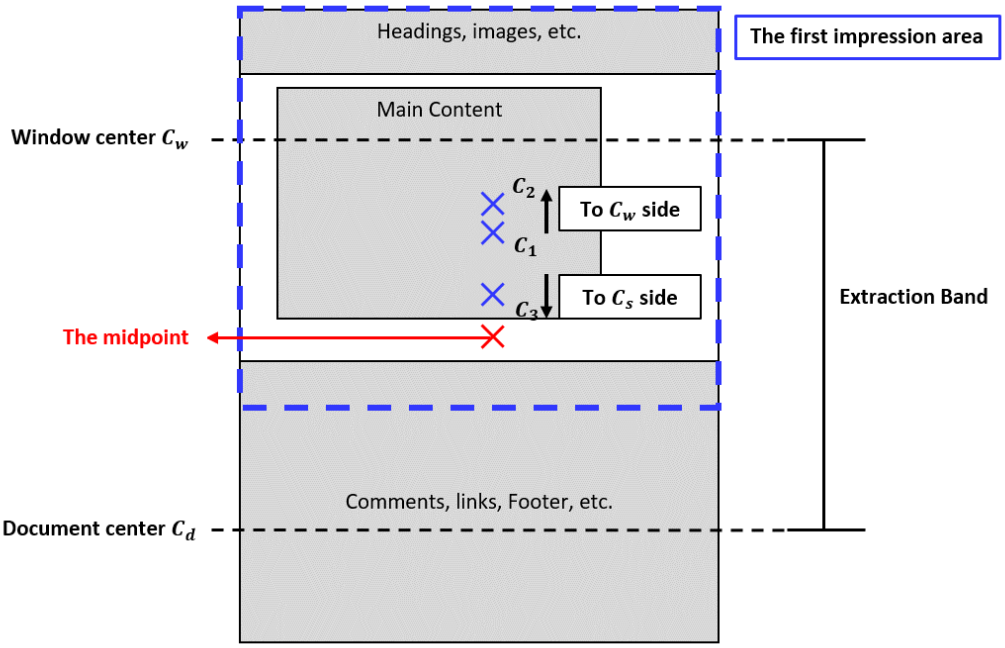

FIGURE 3. Example case of centering step when the midpoint cannot is outside of the main content.

and shapes, it is more effective to use several rules specialized in each. The starting point of the traversal is the closest leaf node from each center point in C. Among the first nodes that correspond with each rule, the node with the highest information density became the main content candidate of the traversing round. This traversal was repeated for each center of C to select the final extraction result among the three candidates. In section 3.3, we explain the three classifiers and text node area density as an information density criterion.

The input of the proposed method and output is a single HTML element such as Reader mode (Readability.js and DOM distiller) with several parameters: N_{col}, α, β, and Δw.

A. STEP 1: GRID

The FIA grid is a checkboard-shaped $N_{row} \times N_{col}$ grid. Its cell size is determined by dividing the browser window size by the number of rows N_{row} and columns N_{col}. For example, the cell size of 4×6 grid is 320×270 in 1920×1080 (Full HD) screen. We determined the number of rows N_{row} with:

$$N_{row} = \text{ceil}(\frac{\min(\alpha h_0, h_1)}{h_c}),$$

where h_0 is the web document height, h_1 is the browser window height, and h_c is the cell height. α is a coefficient that simulates the unconscious browsing behavior, such as a mouse wheel scrolling looking to scan the web page, of a user. This quick behavior helps grow the First impressions area downward. N_{col} is a threshold. In our experiments, the best values were 7×8 grid in 1920×1080 display with $\alpha = 2$.

After the FIA grid's size was set, we excluded cells that were presumably irrelevant from the main content area. The

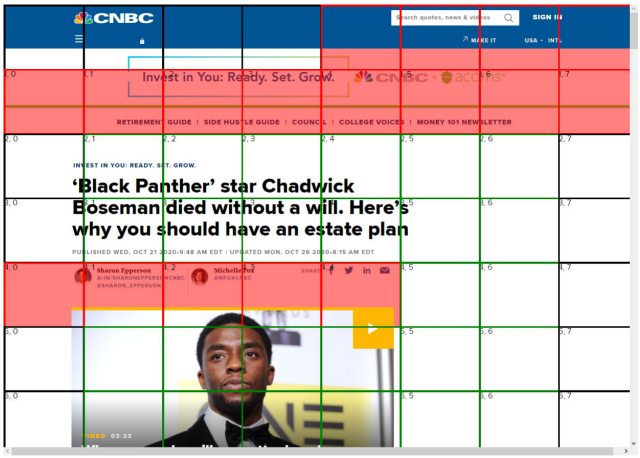

FIGURE 4. FIA grid with excluded cells (red colored cells).

targets were (1) the edge cells or (2) cells overlapped with the navigation menu and ads area. The edge cells were excluded since it was too far from the center and mostly blank (black border cells in Fig. 4). The navigation menu and ads were treated as dense link areas, and many extraction methods were used to filter them out (red-colored cells in Fig. 4). One of the most famous approaches for content extraction was exploiting <A> tags [14], [41]. The most common link density equations, such as *textCount* (or *WordCount*) / *linkTextCount*, had two problems. First, it underestimated images with links, and second, *textCount* and *wordCount* (based on space-separated) values depended on the language. For example, the tendency to use a short word in the menu in Chinese, Japanese, and Korean, the *textCount* or

wordCount approach often missed menu area. Most words in the top-side menu had two or three letters in CJK news websites. Therefore, we used the link node area density $D_l(e)$ of the visible element e with at least one link descendant.

$$D_l(e) = \frac{A_l(e)}{A(e)},$$

where $A(e)$ is the size of e ($width \cdot height$) and $A_l(e)$ is the sum of the areas of the link containers that are descendants of e. If a parent node of any <A> tag or link container element had a single child, the parent became the link container recursively. We labeled a text node and its descendants with $D_l(e) > \beta$ into high-density link text node whereas the rest were labeled into low-density link text node. The best β was 0.5 in our experiment.

B. STEP 2: CENTERING

The centering step was used to determine the centers that were likely to lie on the main content with a high probability. Based on our observation, the better centers compared to the midpoint could exist in the extraction band: between the browser window center and web document center. The centers $C = C_1, C_2, C_3$ were obtained as follows:

$$\begin{aligned} C_1 &= \text{Centroid}(V_a), \\ C_2 &= \text{Centroid}(V_a \cup \{C_w\}), \\ C_3 &= \text{Centroid}(V_a \cup \{C_w, C_d\}), \end{aligned}$$

where the browser window center C_w, web document center C_d, and $V_a = \{v | v \text{ is the center of a cell in the FIA grid}\}$.

C_2 and C_3 are the weighted points of C_1 to the browser center and web document center. C_w is for the starting point when users first look at a web page. C_d is for a document with a large height. Although the points C are expected to be close to the main content, it does not represent the actual HTML element node. Therefore, among the low-density link text nodes, the closest DOM leaf nodes $E = E_1, E_2, E_3$ are derived from each C_1, C_2 and C_3.

C. STEP 3: EXPANDING

Ideally, the leaf nodes E from the centering step were a part of the main content. Hence, a wrapper node that includes the whole main content can be found by expanding the wrapper area from the leaf node E. The wrapper candidate would exist during traversing nodes from the leaf node E to <body> (green route on Fig. 5).

However, the lack of subtree expansion resulted in a low recall while excessive expansion resulted in low precision. Therefore, it was necessary to extend the subtree to include the main content as much as possible while excluding irrelevant elements. We suggested three rules to extract the main content: (1) the <ARTICLE> tag, (2) the words ''article'' and ''content'' in the node attributes, and (3) width difference with the parent node. The first two rules have been widely used in other methods and have worked well in modern websites. The third rule was created based on our observation and can be applied to many old and non-English websites. Although more rules can be included in the while loop of Algorithm 1, we tried to apply as few non-strict rules as possible to reduce bias.

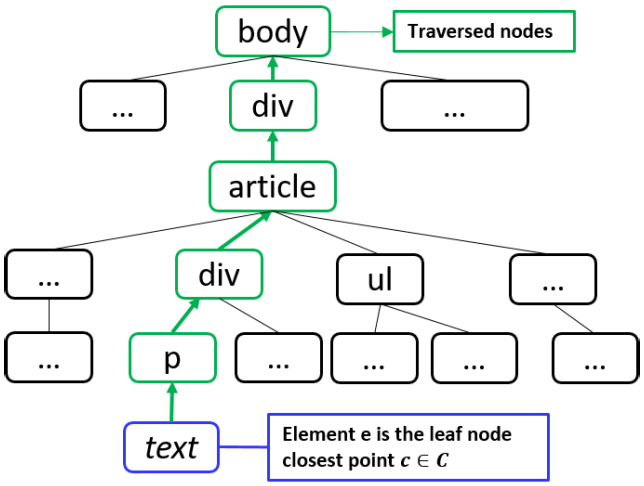

FIGURE 5. DOM tree traversal in Expanding Step.

Algorithm 1 Find Main Content Candidates M by Traversing Nodes From Element E to <body>

```
E ← E1, E2, E3                          ▷ Leaf nodes from each C
M ← ∅
for each En in E do
    N ← En
    P ← parentNode ( N )                ▷ Parent node of N
    Mn_tag, Mn_attr, Mn_diff ← ∅        ▷ Each Mn can be set only once in the below while loop
    while N is not <body> do
        Mn_tag ← P if P is <ARTICLE>
        Mn_attr ← P if "article" or "content" in P's attributes
        Mn_diff ← P if width(P) > Δw · width(N)
    Add Mn_tag, Mn_attr, Mn_diff to set M
Return M
```

1) <ARTICLE> TAG: M_{tag}

<ARTICLE> tag, added in HTML 5, indicates that the content is a self-contained composition of the document, page, app, or site. Therefore, the content of the <ARTICLE> tag is likely to have readable texts. However, not all web pages are HTML 5, while HTML 5 has not compelled the tag to have the main content. Moreover, it has other uses. For example, if a page had a list of multiple articles similar to a magazine stand, the <ARTICLE> tag was recommended for each article. Nonetheless, this tag is appropriate as the main content feature considering the use of this tag to the main content is the best practice in forum posts, news, and blogs.

2) POSITIVE ATTRIBUTES: M_{attr}

The main content area had a specific role in a web page and should be visually separated from other functional areas. Some HTML attributes such as id and class have been commonly used to refer to them. Likewise, web page providers usually assign attributes to handle main content areas. This method is effective and widely used. For example, Readability.js increases the weight of the main content possibility of an HTML element if the element has the following positive words: article, body, content, entry, hentry, h-entry, main, page, pagination, post, text, blog, and story. GCE considers only two words: article and content, because other positive words can imply only a part of the main content.

3) WIDTH INCREASE: M_{diff}

Traversing the DOM tree from a leaf node to root equals the merging area with sibling nodes. If the main content area has a fixed width, the wrapper of the area should increase downward as the contents (text or image) are merged. However, if horizontal merging occurs, the wrapper merges with other areas such as the headers, menus, and footers. Therefore, merging peripheral areas causes a sudden increase in the width. We stopped the expansion when the width increases over Δw in a single expansion step.

Among the subtree root selected by each rule, the root with the highest text node area density became the candidate M_n of E_n. Several other methods have used text density as a criterion. For example, the text density of [36] counts the number of text fragments (token) and lines. As mentioned earlier, since the token in some languages cannot be counted in the same way as English, we counted the area occupied by the text rather than counting the text itself. The text node area density $\mathrm{D_t}(e)$ of element e is defined as:

$$\mathrm{D_t}(e) = \frac{\sum \mathrm{A}(T_e)}{\mathrm{A}(e)},$$

where T_e is a descendant node of e with a low-link density and $\mathrm{A}(e)$ is the size of e ($width{\cdot}height$). $\mathrm{D_t}(e)$ cannot be calculated with the M_{nobest} conditions: M was <BODY> node, or the height of M was less than half of the browser window height. Otherwise, the M_{best} with the largest text node area density was selected. Consequently, there can be three bests, M_1, M_2, and M_3, from the centers C. We determine the final result in the following order: M_{3_best}, M_{2_best}, M_{1_best}, M_{3_nobest}, M_{2_nobest}, and M_{1_nobest}. This order was better than choosing M_1 or M_2 first in our experiment. If all M are <BODY> node, the extraction was considered as a failure.

IV. EXPERIMENT

A. DATASET

To evaluate the performance of the main content extraction methods, we created new multilingual datasets.[3] For our English dataset, we collected GoogleTrends' global region keywords in 2017,[4] similar to BoilerNet.

[3]Available here: https://dx.doi.org/10.21227/rj0q-t583

[4]https://trends.google.com/trends/yis/2017/GLOBAL/

TABLE 2. Experimental datasets.

Name	Region	URLs	Crawled	Readable
GoogleTrends-2017	Global(English)	12720	390	285
GoogleTrends-2020	Global(English)	10560	388	240
GoogleTrends-2020-KR	South Korea	296	43	21
GoogleTrends-2020-JP	Japan	450	47	24
GoogleTrends-2020-ID	Indonesia	900	50	31
GoogleTrends-2020-FR	France	1580	97	39
GoogleTrends-2020-RU	Russia	890	95	48
GoogleTrends-2020-SA	Saudi Arabia	260	97	43
Baidu-2020 (CN)	China	1990	193	53

Furthermore, we similarly collected the keywords in 2020.[5] The non-English keywords from South Korea, Japan, France, Indonesia, Russia, and Saudi Arabia are provided by each region of GoogleTrends in 2020. Conversely, the Chinese keywords are from Baidu[6] because GoogleTrends has not provided GoogleTrends-CN. As a result, the top 100 URLs were collected by querying the search engines (Google and Baidu) from the nine keyword sets (URLs column in Table 2). We crawled and saved web pages from randomly selected URLs from the collected (Crawled Column in Table 2). These web pages were saved as MHTML files to contain the style data of the web page. Because the crawling targets were randomly selected, some pages (e.g. removed page, pdf, or file link) were discarded by hand.

We labeled ground truth with our browser extension linked to the DOM inspector in the web browser. Our annotators proceeded with labeling based on the following steps: (1) Select a word within the area corresponding to each category ('title,' 'navigation menu,' and 'main content'); (2) using DOM inspector, ascend the DOM tree until the selected node covers all areas of the main content in their mind; and (3) confirm the result and send it to server. If they thought the result was not what they intended in the ascending step, the steps are repeated by selecting another word. We assigned web pages of languages that each annotator can speak. As a result, the web pages including 'main content' label regarded as a readable web page (Readable Column in Table 2).

B. EVALUATION

The thresholds and values of GCE in the experiments were as follows: the browser window size with 1920×1080, 1280×1024, and 2560×1440. The rows and columns of the grid were from 3×3 to 8×8 (up to 24×36 grid in 2560×1440). α and β were [0.25, 0.5, 0.75] and [1.5, 2, 2.5], respectively. The width increase in the expanding step was tested with [130%, 150%, 170%]. We present the best in the result section with these values: 7×8 grid in 1920×1080 window with α=0.5, β=2, and the width increase Δw by 170%. Six methods were used for the experiment with the GCE in the above settings. The four existing methods were Readability.js, DOM Distiller, BoilerNet, and Web2text.

[5]https://trends.google.com/trends/yis/2020/GLOBAL/

[6]https://baijiahao.baidu.com/s?id=1686016936405463174

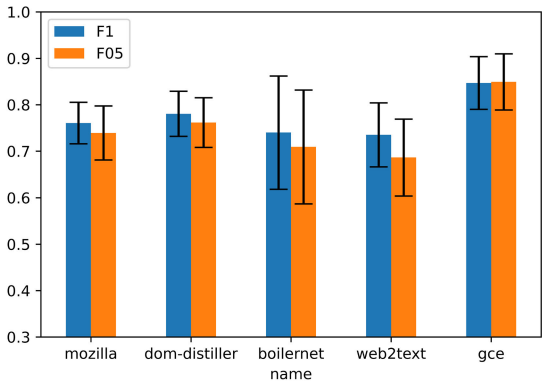

FIGURE 6. LCS F_1 and $F_{0.5}$ score.

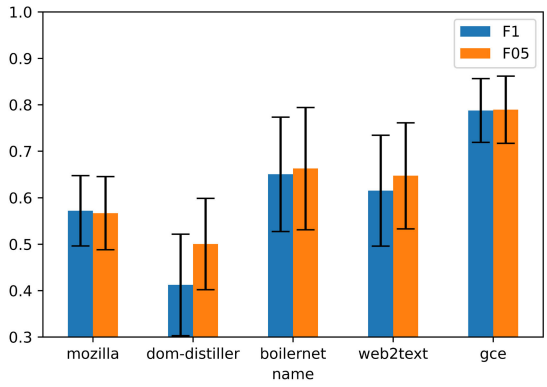

FIGURE 7. Matched text blocks F_1 and $F_{0.5}$ score.

As for performance measures, the longest common subsequence (LCS) [41] and matched text blocks [15] were used. For the LCS, $\mathbf{F_1}$ scores were calculated as the length of the longest common subsequence by comparing the extracted result with the text of the actual main content. Although LCS is intuitive and straightforward, it does not tell how close the result is to the actual main content semantically considering it does not examine the source of the extracted text node.

The matched text blocks calculates the number of extracted nodes matching the text nodes constituting the main content. Furthermore, we used $\mathbf{F_1}$ scores in this measure. Unlike LCS, the same texts belonging to different nodes can be distinguished. Nevertheless, the weight between text nodes cannot be calculated. For example, a <P> tag comprising a long text with rich information and a <SPAN> tag of a single word are weighted equally in this measure.

Additionally, we assessed the $\mathbf{F_{0.5}}$ score of both measures. There are two kinds of failures in extracting the main content. If there was no result, the scores would be 0. However, if the result were the entire web page, the recall would become 1.0, and the F-score would not be 0 although the extraction failed. We considered the $\mathbf{F_{0.5}}$ score to give penalty to high recall.

C. RESULT

Table 3 shows the experiment result, including precision, recall, $\mathbf{F_1}$ and $\mathbf{F_{0.5}}$ scores of the LCS and matched text blocks. The results are summarized in Fig. 6, 7, and 8. Figures 6 and 7 show each method's overall performance and variation by language. Figure 8 shows the performance variation by language in detail. In LCS, $\mathbf{F_1}$ score is higher than the $\mathbf{F_{0.5}}$ score for existing methods, which means they tend to extract slightly larger text area than the main content(Fig. 6). Resulting larger text area is advantageous for extraction methods because it is possible to post-process the remaining result. For example, users can still ignore the noisy area when browser reader mode did not extract the main content thoroughly.

The average scores on the English and non-English datasets were similar, but all methods had language variations(Fig. 8). Most methods scored the worst on the Japanese dataset. The reason is that some algorithms did not work on Japanese websites' original web page layout because Japan has developed its own web environment and the culture and techniques of distinctive web page authoring differ considerably from those used for English websites.

Conversely, regions with a relatively short history in web development, such as China, had less performance degradation because of similarities in templates or underlying frameworks of web pages, even if the language is different from English. For example, most web pages in Chinese datasets are based on well-known platform templates such as Baidu, even though we sampled more web pages than other non-English datasets. All methods, except BoilerNet, performed well on the Chinese web pages (Fig. 8). Moreover, the English dataset (years 2017 and 2020) includes web pages that deal with relatively diverse appearances and topics, even though famous templates, such as WordPress, are widely used in English web pages. This is because of the long history of the standalone personal website in the global (English) pages that have made various web page templates and structures.

The $\mathbf{F_{0.5}}$ scores of GCE's LCS and the matched text blocks were over 0.839 and 0.776, respectively, which were the best scores in the experiment, and showed a 30% improvement over Web2Text's average score (0.641) and 35% improvement over Web2Text's average score (0.574), respectively. Similar to that of other methods, the performance of LCS was worst in the Japanese dataset but it had a relatively better score than the others. Notably, it shows little performance difference between $\mathbf{F_1}$ and $\mathbf{F_{0.5}}$. This result demonstrates that our method could prevent the generation of the <body> as output because the <body> indicates a recall value of 1.00 and is not a helpful output.

Readability.js yielded acceptable performance (LCS $\mathbf{F_{0.5}}$ 0.760, Block $\mathbf{F_{0.5}}$ 0.586), which indicated that high performance was achieved even with traditional rule-based methods through continuous maintenance. As a reader mode, it conservatively removes boilerplate; therefore, the matched text blocks score is relatively low. For example, it remained "Cookie Statement" blocks in several web pages.

The DOM Distiller had the high LCS score ($\mathbf{F_{0.5}}$ 0.744) but scored relatively low on the matched block text(Fig. 7 and Fig. 8). Owing to its text node merging process, we cannot appropriately measure it on the matched block text. For

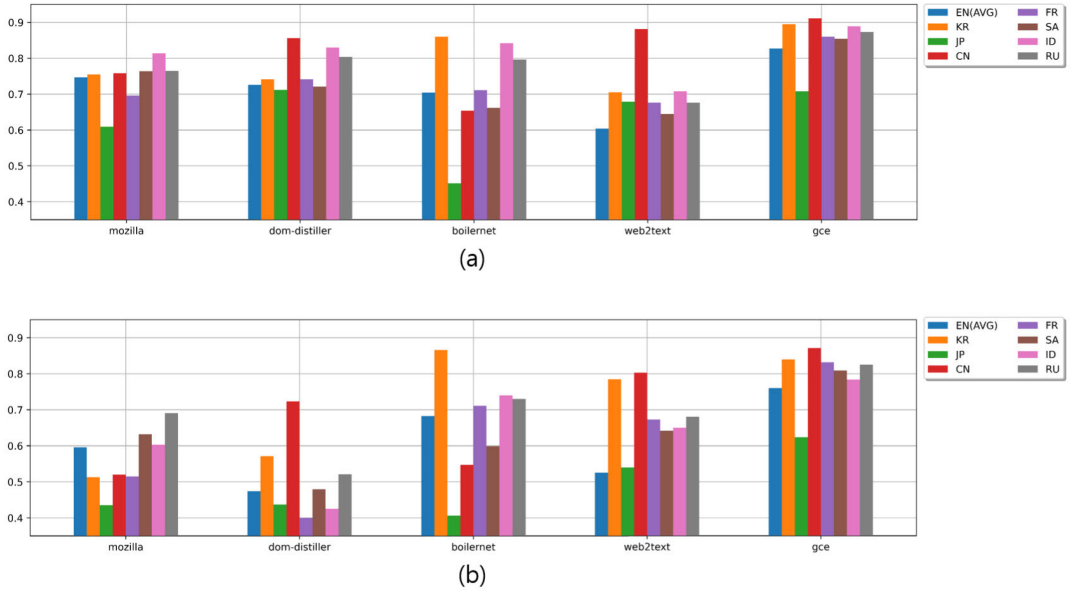

FIGURE 8. $F_{0.5}$ scores by datasets. (a) LCS, (b) Matched text blocks.

TABLE 3. Result of LCS and the matched text blocks by methods.

	Readability.js								DOM distiller							
	LCS				Matched Text Block				LCS				Matched Text Block			
	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5
2017	.761	.868	.743	.728	.571	.686	.580	.567	.760	.881	.749	.722	.705	.289	.339	.454
2020	.796	.905	.776	.765	.646	.766	.634	.625	.762	.886	.751	.729	.747	.369	.399	.494
KR	.783	.905	.784	.755	.514	.511	.511	.513	.831	.813	.753	.741	.848	.427	.492	.571
JP	.585	.988	.662	.609	.429	.625	.448	.435	.753	.921	.742	.712	.697	.353	.365	.437
CN	.757	.939	.787	.758	.525	.540	.519	.520	.862	.962	.876	.856	.869	.613	.653	.723
FR	.736	.911	.729	.696	.531	.673	.534	.515	.793	.879	.757	.741	.730	.239	.294	.400
SA	.803	.919	.775	.764	.630	.751	.641	.632	.769	.917	.748	.721	.760	.414	.420	.479
ID	.830	.914	.817	.814	.640	.636	.593	.603	.878	.909	.837	.830	.820	.263	.316	.425
RU	.843	.887	.773	.765	.721	.729	.687	.691	.886	.882	.812	.804	.849	.384	.433	.521
Avg.	.775	.897	.760	.744	.598	.698	.595	.586	.784	.891	.766	.743	.750	.351	.391	.482
	BoilerNet								Web2Text							
	LCS				Matched Text Block				LCS				Matched Text Block			
	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5
2017	.680	.951	.751	.703	.708	.811	.696	.690	.564	.939	.651	.589	.553	.411	.382	.431
2020	.684	.952	.748	.705	.699	.817	.680	.675	.600	.959	.677	.619	.636	.810	.628	.620
KR	.846	.952	.887	.860	.898	.807	.834	.866	.672	1.000	.771	.705	.864	.716	.742	.785
JP	.459	.694	.477	.451	.447	.588	.406	.406	.663	.946	.714	.679	.701	.550	.485	.540
CN	.647	.721	.667	.654	.595	.579	.530	.547	.878	.936	.889	.881	.873	.722	.756	.803
FR	.690	.921	.755	.711	.735	.775	.703	.711	.644	.996	.739	.676	.696	.824	.679	.673
SA	.680	.881	.696	.662	.663	.736	.605	.599	.624	.931	.690	.645	.740	.688	.597	.642
ID	.843	.963	.853	.842	.849	.681	.684	.740	.694	.939	.746	.708	.709	.658	.610	.650
RU	.780	.940	.828	.796	.763	.806	.716	.730	.663	.917	.739	.676	.737	.703	.658	.681
Avg.	.689	.922	.745	.706	.702	.779	.671	.672	.621	.948	.695	.641	.647	.629	.547	.574
	GCE															
	LCS				Matched Text Block											
	Prec.	Recall	F1	F0.5	Prec.	Recall	F1	F0.5								
2017	.848	.909	.830	.821	.769	.859	.764	.749								
2020	.863	.920	.838	.833	.795	.890	.777	.771								
KR	.961	.898	.892	.895	.895	.867	.840	.840								
JP	.762	.827	.711	.708	.654	.797	.627	.624								
CN	.934	.930	.906	.911	.901	.880	.860	.871								
FR	.932	.860	.843	.860	.875	.830	.808	.832								
SA	.860	.960	.858	.854	.814	.918	.812	.809								
ID	.926	.865	.870	.889	.850	.811	.769	.784								
RU	.894	.912	.875	.873	.837	.896	.833	.825								
Avg.	.886	.898	.842	.839	.821	.861	.788	.776								

example, if adjacent text nodes existed, such as continuous <li>, it merges them and creates a new single text node. Therefore, we did not restore the modified nodes by each method to measure the result as is. In other case, it showed good performance on all datasets, regardless of language.

BoilerNet was as good as two reader modes (LCS $\mathbf{F_{0.5}}$ 0.706, Block $\mathbf{F_{0.5}}$ 0.672). It shows the highest score degradation in the Japanese (LCS $\mathbf{F_{0.5}}$ 0.451, Block $\mathbf{F_{0.5}}$ 0.406), the Saidu Arabian (Arabic) (LCS $\mathbf{F_{0.5}}$ 0.662, Block $\mathbf{F_{0.5}}$ 0.599), and the Chinese (LCS $\mathbf{F_{0.5}}$ 0.654, Block $\mathbf{F_{0.5}}$ 0.547) datasets. Thus, the approach of BoilerNet can be highly affected by languages that have different linguistic characteristics from English.

Web2Text had the lowest performance score ($\mathbf{F_{0.5}}$ 0.695). However, it was slightly better for the non-English datasets compared to the English datasets, which indicated that the WebText's structural-based features were more appropriate than its text-based features for modern web pages.

V. CONCLUSION AND DISCUSSION

We expected the performance to degrade owing to the characters, writing systems, and local web page design trend. Most methods in our experiment have a low score in the Japanese dataset, which has a very different character and writing system from English, and hence, has developed its own web environment due to the early introduction of the Web.

However, the experimented methods, except BoilerNet, showed no significant performance drop in the Chinese datasets possibly due to lack of diversity in our Chinese dataset. Most web pages had similar templates from Baidu considering we had to collect web pages from the Baidu search result. Therefore, more studies are needed on which text-based features are affected by languages through other Chinese or non-English language datasets.

This study examined why some content extraction methods work poorly on non-English web pages. The proposed method is based on universal principles than the languages of web pages: how people perceive and how web pages attract users. In 98% of web pages in our datasets, the main content area spans the extraction band, below the screen center, and above the document center. We can also reduce the number of main content candidates using the concept of FIA to outperform the well-known methods with a simple rule-based algorithm on fewer candidates. Additionally, considering these rules are independent, other rules or classifiers can be appended to improve the performance or create a content extractor for a specific purpose.

We have experienced cases where multiple applications and technologies do not work or perform poorly owing to the language. Since the Web is global in scale, the technology barriers to language should be reduced. Therefore, we tried to reduce the web experience of end-users caused by linguistic differences. Most main content extraction methods have been mono-lingual, which works only on English web pages. We have shown that the visual and structural features related to user behavior could create a better main content extraction method for multiple languages. Because we focused on the desktop environment for end users, the propsoed method may need adjustment for the mobile environment (e.g., narrow and long web document shape in portrait mode). Moreover, the concepts of our method, utilizing the visual properties of web pages for users, can be applied in any softwares and applications dealing with web content for visual properties.

REFERENCES

[1] Y. Yesilada, "Web page segmentation: A review," eMINE Tech. Rep. Deliverable 0 (D0), Middle East Tech. Univ. Northern Cyprus Campus, 2011, pp. 1–39.
[2] E. Ferrara, P. De Meo, G. Fiumara, and R. Baumgartner, "Web data extraction, applications and techniques: A survey," *Knowl. Based Syst.*, vol. 70, pp. 301–323, Nov. 2014.
[3] J. Alarte and J. Silva, "Page-level main content extraction from heterogeneous webpages," *ACM Trans. Knowl. Discovery Data*, vol. 15, no. 6, pp. 1–105, Jun. 2021.
[4] J. Alarte, J. Silva, and S. Tamarit, "What web template extractor should i use? A benchmarking and comparison for five template extractors," *ACM Trans. Web*, vol. 13, no. 2, pp. 1–19, May 2019.
[5] T. van den Hout, T. Wabeke, G. C. M. Moura, and C. Hesselman, "LogoMotive: Detecting logos on websites to identify online scams—A TLD case study," in *Proc. Int. Conf. Passive Act. Netw. Meas.*, 2022, pp. 3–29.
[6] R. Štrimaitis, P. Stefanovič, S. Ramanauskaitė, and A. Slotkienė, "Financial context news sentiment analysis for the Lithuanian language," *Appl. Sci.*, vol. 11, no. 10, p. 4443, May 2021.
[7] W. Thanadechteemapat and C. C. Fung, "Automatic content extraction and visualization of Thai websites for improved information representation," in *Proc. IEEE Int. Conf. Syst., Man, Cybern. (SMC)*, Oct. 2012, pp. 2229–2234.
[8] Z. S. Zubi, "Using some web content mining techniques for Arabic text classification," in *Proc. 8th WSEAS Int. Conf. Data Netw., Commun., Comput. (DNCOCO)*, 2009, pp. 73–84.
[9] H. M. Alghamdi and A. Selamat, "Arabic web page clustering: A review," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 31, no. 1, pp. 1–14, Jan. 2019.
[10] S. Djamasbi, "Eye tracking and web experience," *AIS Trans. Hum.-Comput. Interact.*, vol. 6, no. 2, pp. 37–54, Jun. 2014.
[11] G. Lindgaard, G. Fernandes, C. Dudek, and J. Brown, "Attention web designers: You have 50 milliseconds to make a good first impression!" *Behav. Inf. Technol.*, vol. 25, no. 2, pp. 115–126, 2006.
[12] J. A. Martínez-González and D. C. Álvarez-Albelo, "Influence of site personalization and first impression on young consumers' loyalty to tourism websites," *Sustainability*, vol. 13, no. 3, pp. 1–18, 2021.
[13] W. Liu, Y. Cao, and W. R. Proctor, "The roles of visual complexity and order in first impressions of webpages: An ERP study of webpage rapid evaluation," *Int. J. Hum.-Comput. Interact.*, vol. 38, pp. 1–14, Aug. 2021.
[14] C. Kohlschütter, P. Fankhauser, and W. Nejdl, "Boilerplate detection using shallow text features," in *Proc. 3rd ACM Int. Conf. Web Search Data Mining (WSDM)*, 2010, pp. 441–450.
[15] T. Vogels, O. E. Ganea, and C. Eickhoff, "Web2Text: Deep structured boilerplate removal," in *Proc. Eur. Conf. Inf. Retr.*, 2018, pp. 167–179.
[16] J. Leonhardt, A. Anand, and M. Khosla, "Boilerplate removal using a neural sequence labeling model," in *Proc. Companion Proc. Web Conf.*, Apr. 2020, pp. 226–229.
[17] D. Song, F. Sun, and L. Liao, "A hybrid approach for content extraction with text density and visual importance of DOM nodes," *Knowl. Inf. Syst.*, vol. 42, no. 1, pp. 75–96, Jan. 2015.
[18] D. Cai, S. Yu, J.-R. Wen, and W.-Y. Ma, "VIPS: A vision-based page segmentation algorithm," Tech. Rep. MSR-TR-2003-79, 2003.
[19] S. Alcic and S. Conrad, "Page segmentation by web content clustering," in *Proc. Int. Conf. Web Intell.*, 2011, pp. 1–9.
[20] A. Barbaresi, "Trafilatura: A web scraping library and command-line tool for text discovery and extraction," in *Proc. 59th Annu. Meeting Assoc. Comput. Linguistics 11th Int. Joint Conf. Natural Lang. Process., Syst. Demonstrations*, 2021, pp. 122–131.
[21] J. Hernandez, H. M. Marin-Castro, and M. Morales-Sandoval, "A semantic focused web crawler based on a knowledge representation schema," *Appl. Sci.*, vol. 10, no. 11, p. 3837, May 2020.

[22] P. Xiang, X. Yang, and Y. Shi, "Effective page segmentation combining pattern analysis and visual separators for browsing on small screens," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI Main Conf. Proceedings)(WI)*, Dec. 2006, pp. 831–837.
[23] D. Shaw, "An eye-tracking evaluation of multicultural interface designs," M.S. thesis, Boston College Electron., 2005, pp. 1–77.
[24] L.-M.-U. Munich, M. Maurer, J.-U. Mainz, and C. Oschatz, "What you see is what you know: The influence of involvement and eye movement on online users' knowledge acquisition," *Int. J. Commun.*, vol. 13, p. 25, Aug. 2019.
[25] D. J. Wells, S. J. Valacich, and J. T. Hess, "What signal are you sending? How website quality influences perceptions of product quality and purchase intentions," *MIS Quart., Manage. Inf. Syst.*, vol. 35, no. 2, pp. 373–396, 2011.
[26] G. Wagner, H. Schramm-Klein, and S. Steinmann, "Online retailing across e-channels and e-channel touchpoints: Empirical studies of consumer behavior in the multichannel e-commerce environment," *J. Bus. Res.*, vol. 107, pp. 256–270, Feb. 2020.
[27] F. Guo, J. Chen, M. Li, W. Lyu, and J. Zhang, "Effects of visual complexity on user search behavior and satisfaction: An eye-tracking study of mobile news apps," *Universal Access Inf. Soc.*, vol. 21, no. 4, pp. 795–808, Nov. 2022.
[28] K. Karvonen, "The beauty of simplicity," in *Proc. Conf. Universal Usability (CUU)*, 2000, pp. 85–90.
[29] S. Djamasbi, M. Siegel, T. Tullis, and R. Dai, "Efficiency, trust, and visual appeal: Usability testing through eye tracking," in *Proc. 43rd Hawaii Int. Conf. Syst. Sci.*, 2010, pp. 1–10.
[30] J. E. Lee, S. Hur, and B. Watkins, "Visual communication of luxury fashion brands on social media: Effects of visual complexity and brand familiarity," *J. Brand Manage.*, vol. 25, no. 5, pp. 449–462, Sep. 2018.
[31] H. Wan, W. Ji, G. Wu, X. Jia, X. Zhan, M. Yuan, and R. Wang, "A novel webpage layout aesthetic evaluation model for quantifying webpage layout design," *Inf. Sci.*, vol. 576, pp. 589–608, Oct. 2021.
[32] M. Alsaffar, L. Pemberton, K. R. Echavarria, and M. Sathiyanarayanan, "Visual behaviour in searching information: A preliminary eye tracking study," in *Proc. 11th Int. Conf. Res. Challenges Inf. Sci. (RCIS)*, May 2017, pp. 365–370.
[33] Y. Dong and K. P. Lee, "A cross-cultural comparative study of users' perceptions of a webpage: With a focus on the cognitive styles of Chinese, Koreans and Americans," *Int. J. Des.*, vol. 2, no. 2, pp. 19–30, 2008.
[34] G. Lindgaard, J. Litwinska, and C. Dudek, "Judging web page visual appeal: Do east and west really differ?" in *Proc. IADIS Multi Conf. Comput. Sci. Inf. Syst.*, 2008, pp. 157–164.
[35] J. M. Henderson and F. Ferreira, "Effects of foveal processing difficulty on the perceptual span in reading: Implications for attention and eye movement control," *J. Experim. Psychol., Learn., Memory, Cognition*, vol. 16, no. 3, pp. 417–429, 1990.
[36] C. Kohlschütter and W. Nejdl, "A densitometric approach to web page segmentation," in *Proc. 17th ACM Conf. Inf. Knowl. Mining (CIKM)*, 2008, pp. 1173–1182.
[37] J. J. Andrew, S. Ferrari, F. Maurel, G. Dias, and A. Giguet, "Web page segmentation for non visual skimming," *Proc. 33rd Pacific Asia Conf. Lang., Inf. Comput. (PACLIC)*, 2019, pp. 423–431.
[38] T. Wei, Y. Lu, X. Li, and J. Liu, "Web page segmentation based on the Hough transform and vision cues," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA)*, Dec. 2015, pp. 865–872.
[39] M. Cormier, K. Moffatt, R. Cohen, and R. Mann, "Purely vision-based segmentation of web pages for assistive technology," *Comput. Vis. Image Understand.*, vol. 148, pp. 46–66, Jul. 2015.
[40] J. Zeleny, R. Burget, and J. Zendulka, "Box clustering segmentation: A new method for vision-based web page preprocessing," *Inf. Process. Manage.*, vol. 53, no. 3, pp. 735–750, May 2017.
[41] F. Sun, D. Song, and L. Liao, "DOM based content extraction via text density," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. (SIGIR)*, 2011, pp. 245–254.

GEUNSEONG JUNG received the B.S. degree in computer science from Hanyang University, Seoul, South Korea, in 2014, where he is currently pursuing the Ph.D. degree. His research interests include web content extraction, web data mining, web crawlers, and web automation.

SUNGJAE HAN received the Ph.D. degree in computer science from Hanyang University, Seoul, South Korea, in 2015.

From 2015 to 2021, he was the Principal Research Engineer with KT, Seoul. He is currently the Director of the AI Research Laboratory, JEI Group, Seoul. His research interests include web content analysis and e-learning.

HANSUNG KIM received the Ph.D. degree in social welfare from the School of Social Work, University of Southern California, Los Angeles, CA, USA, in 2008.

From 2008 to 2011, he worked as an Assistant Professor with the Department of Social Work, California State University, Fullerton, CA, USA. He is currently a Professor with the Department of Sociology, Hanyang University, Seoul, South Korea. His research interests include social welfare, social policy, poverty, inequality, and underprivileged.

KWANGUK KIM received the Ph.D. degree in computer science from Hanyang University, Seoul, South Korea, in 2009. From 2008 to 2010, he was a Fellow Researcher with Duke University, Durham, NC, USA. From 2010 to 2013, he was a Fellow Researcher with the University of California at Davis, Davis, CA, USA. He is currently an Associate Professor with the Department of Computer Science, Hanyang University. His research interests include human–computer interaction, virtual reality, and human–robot interaction.

JAEHYUK CHA (Member, IEEE) received the B.S., M.S., and Ph.D. degrees in computer science from Seoul National University, South Korea, in 1987, 1991, and 1997, respectively. He was associated with the Korea Research Information Center, from 1997 to 1998. He is currently a Professor with the Department of Computer Science and the Dean of Paiknam Library, Hanyang University, Seoul, South Korea. His research interests include DBMS, flash storage systems, and multimedia content adaptation.