SoftwareX 23 (2023) 101501

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

A WebExtension framework for experimentation and evaluation of webpage segmentation methods

Geunseong Jung, Jaehyuk Cha *

Department of Computer Science, Hanyang University, Seoul, South Korea

ARTICLE INFO

Article history:
Received 31 January 2023
Received in revised form 26 June 2023
Accepted 7 August 2023

Keywords:
Data mining
Web mining
Web extensions

ABSTRACT

Current webpages contain areas with different functions and contents. Many studies and applications have used webpage segmentation methods to separate these areas or extract only specific areas for their purposes. Examining these methods requires laborious tasks, such as collecting many webpages, inspecting them with human participants, and applying various performance metrics to their results. Therefore, we developed a WebExtension (browser extension) framework to support the examination and analysis of webpage segmentation methods. This framework can build a WebExtension to collect webpages, curate data for labeling web documents, evaluate methods, and measure the results with various performance metrics in a web browser environment. Furthermore, researchers can use preloaded well-known methods and metrics in the framework and add more methods and metrics for their research purposes.

Code metadata

Current code version	1.0.0
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-23-00076
Permanent link to reproducible capsule	–
Legal code license	MIT License.
Code versioning system used	git
Software code languages, tools, and services used	Typescript
Compilation requirements, operating environments, and dependencies	Requirements: Chromium/ Chrome Browser >= 80, node >= 14, Postgresql >= 13, MongoDB >= 4.4 Dependencies: typescript >= 4.3.5, browserlist >= 1.0.1, lodash >= 4.17.21, parcel-bundler = 1.12.4, parcel-plugin-web-extension = 1.6.1, tslib >= 2.0.3, js-yaml >= 4.1.0, diff-sequences >= 27.0.6, auto-bind >= 4.0, jquery >= 3.6.0, has >= 1.0.3
If available, link to developer documentation/manual	–
Support email for questions	aninteger@hanyang.ac.kr

1. Motivation and significance

People share news, information, thoughts, and feelings through social media via the web, which has become the richest source of information in various forms, mainly text, images, and videos [1,2]. Many research fields, such as information retrieval [3–7], natural language processing [8,9], and marketing [10,11], require this type of data.

Currently, a single webpage consists of areas with multiple functions and content instead of a single piece of information [2,5]. For example, web news articles have advertisements, navigation menus, comments, and other website boilerplates. Webpage segmentation, a preprocessing task for separating each area, is essential for building a dataset on these webpages. Although many webpage segmentation techniques exist, creating a webpage dataset still requires the effort and labor of researchers.

One of the challenges is about creating a dataset. Although many existing studies presented datasets containing webpage segmentation, researchers are forced to construct a new dataset. First, due to the rapid changes in web technology and trends, many datasets are old and do not reflect recent webpages [5].

* Corresponding author.
E-mail addresses: aninteger@hanyang.ac.kr (Geunseong Jung), chajh@hanyang.ac.kr (Jaehyuk Cha).

https://doi.org/10.1016/j.softx.2023.101501

Existing datasets also may not meet the research purpose. For example, Leonhardt et al. [5] argued that the dataset is old, but the newly collected data comprise English web pages. Since the web is used worldwide, it is also necessary to build a new dataset to study non-English web pages [12,13].

Examining the collected webpages is another challenge. Datasets must be examined with several processes, such as data cleaning, annotation (labeling), and cross-validation. These processes are often conducted by multiple participants. Although professional tools or services exist for these tasks, researchers may ask for the participation of ordinary people in research subjects like web aesthetics [14] and web accessibility [15]. Thus, collaborative tools for researchers and participants are required in some research scenarios.

Experimenting and evaluating webpage segmentation methods remains challenging. Because of the various purposes of webpage segmentation, webpages can be treated in many forms: a single large text, text sequences, bag-of-words, tree structure (Document Object Model), and an image rendered on a web browser. Similarly, the types of segments to be measured vary according to the segmentation method. For example, Web2Text [4] processes only HTML nodes with text content; therefore, its result has no images, forms, or other non-textual HTML nodes, which can have a visible area when the webpage is rendered. This result can only be measured for performance metrics that calculate the visible area on the displayed webpage with additional post-processing, such as finding a wrapper node that includes all texts in the result.

Therefore, several iterative experiments and evaluations are essential to ensure that the webpage segmentation method is suitable for research purposes. This may involve collecting numerous webpages, comparing several methods with various performance metrics, and controlling many participants. Web automation software, such as web crawlers and annotator tools, presents a partial solution to this problem. However, they are mostly professional and difficult to customize.

This paper presents a JavaScript framework that builds a WebExtension to create data sources and examine webpage segmentation methods on webpages. Using our framework, researchers can build a suitable benchmark WebExtension to evaluate webpage segmentation for their research purposes. Overall, our WebExtension-based framework offers benefits as follows:

- Data collection on web browsers enables the gathering and archiving of webpages based on the appearance displayed in the actual browser, including visual information and user properties.
- In data curation (annotation and labeling) and qualitative analysis, participants can perform labeling tasks in a web browser. Participants can easily perform these tasks on their browsers rather than on hard-to-understand professional software.
- It is extensible for combining web segmentation methods and applying various performance metrics for research purposes.
- The process and results of the segmentation can be displayed on the original webpage via a web browser, which does not require expertise in web techniques and can be understood by ordinary people.

In particular, this framework was used to build a tool for extracting the main content from webpages. We created a multilingual dataset of eight languages using this framework, including human-labeled HTML documents [16]. We also implemented interfaces for well-known webpage segmentation methods (Readability.js [17], DOM distiller [18], Web2Text [4], and BoilerNet [5]) and performance metrics of text and area similarity (longest common subsequence [19], matched text blocks [4,5], and intersection over union (IoU) [20]). The implementations were preloaded for user convenience. Researchers using this framework can refer to or use implementations to build WebExtensions for their experiments.

Although numerous webpage datasets and benchmarks exist, researchers devote their time to reworking the datasets and methods. Due to the diversity of research purposes and application goals, researchers cannot use existing datasets and collect new ones that meet their goals, resulting in much repetitive labor. For example, Alarte et al. [21] presented a workbench to compare template extractors on a new dataset of their other work [22]. Their workbench includes CleanEval [23], a well-known dataset; however, they could use only 20 webpages because of the out-of-date benchmarks in the dataset. They also mentioned that most existing methods are not open or freely accessible and that they had to reimplement five template extractors. In many studies, ad hoc tools for the experiment have been created for human responses because the human annotation from existing datasets can be misunderstood and is often based on outdated webpages.

Leonhardt et al. [5] used CleanEval and newly collected webpages in their experiments. They said that the web had changed a lot since 2007 when CleanEval was announced. Thus, they collected recent webpages from GoogleTrends-2017 keywords. As such, it is very common to need a new dataset due to the aging of the dataset and its non-conformity with purpose.

Wu et al. [13] enhanced the method of Leonhardt et al. [5] to other languages, such as Chinese, Japanese, and Thai. However, they said their non-English datasets were biased toward news and blog websites, unlike English datasets containing multiple domains. This shows the difficulty of collecting webpages from languages other than English.

Wan et al. [14] presented a webpage layout evaluation model for quantifying webpage layout design. Their methodology involved gathering webpages and showing them to participants using questionnaires about the webpage's aesthetics. Although similar tasks have been performed in previous studies, Wan et al. newly indexed 200,000 pages and refined them to a new dataset of 13,017 webpages for their experiment. This shows that webpages' rapid, dynamic, and heterogeneous nature allows researchers to collect, verify, and evaluate new data.

Li et al. [15] also tried to investigate the Reader View's effects on user experience instead of improving the existing techniques. Because existing datasets could not achieve their objectives, they collected new datasets and experimented on the usability and readability of Firefox's Reader View on 391 experimenters, including 42 dyslexic patients. As in this experiment, webpage datasets can also be used in experiments with many non-experts, and this may be considered when building datasets.

Table 1 summarizes the above works regarding research subjects, purposes, and datasets. The webpage dataset is one of the multidisciplinary datasets, and even people who are not experts in web technology use webpages daily. It leads to the subject of webpage segmentation being wide. Therefore, researchers often face situations where they need to collect datasets for new experiments that meet their research objectives. This paper proposes a framework that allows them to construct the desired benchmarks themselves.

2. Software description

The framework is designed to provide access to webpages, user inputs, and external extensions and store their data on a server. Our framework has two parts: as the front-end, the Chrome Extension API (*WebExtension* in Fig. 1), and as the back-end, the Node.js server provides REST APIs for storing data in the

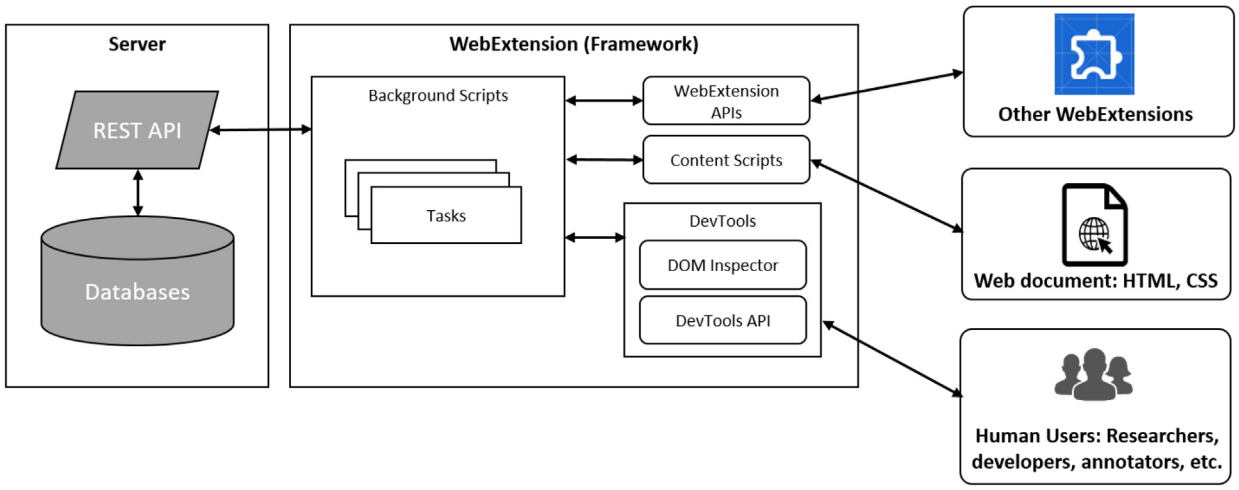

Fig. 1. Architecture diagram of the framework.

Table 1
Existing webpage segmentation benchmarks.

	Alarte et al. [21]	Leonhardt et al. [5]	Wu et al. [13]	Wan et al. [14]	Li et al. [15]
Research subject	Webpage template extraction	Webpage boilerplate removal	Webpage boilerplate removal	Web aesthetics	User experience, Accessibility
Purpose	Benchmarking for template extraction methods	Benchmarking for boilerplate removal methods	Benchmarking for boilerplate removal methods	Dataset for user experiments and questionnaires	Dataset for user experiment
Target Dataset Languages	English, German, French, and Spanish	English	English, Chinese, Japanese, Thai	Mixed (Sampled from Alexa Top 500 Sites 2018)	Mixed (Sampled from Alexa Top 500 Sites 2018)
Type of participants involved dataset	Web engineers	Web engineers	Web engineers	Web users	Web users, including the disabled

Table 2
Summary and recommended usage of the Task templates.

	Crawling	Curation	Extraction	Evaluation
Retrieving webpages	Automatic	Manual	Automatic	Automatic/Manual
Target data	HTML elements	HTML elements, browser events, user response	Depends on the extractor	Depends on the evaluator
Best practices	Webpage crawling and archiving	Web segment annotation, dataset review, experiments with user questionnaires	Batch processing for applying the web segment method	Batch processing for evaluation metrics, user experiments on the controlled browser environment

database connections (*Server* in Fig. 1). A task must be defined for the WebExtension built within this framework to work for a specific purpose. Hence, we provide a Task Class and four templates as implementations of the Task Class. They are optional for WebExtension generation, but they are helpful.

2.1. Task templates

Without any implementation, the framework builds a WebExtension that reads URLs from the database and loads them on a new browser tab. We provide several implementations of the Task class as templates, each representing the task frequently used in webpage segmentation: Crawling, Curation, Extraction, and Evaluation. Table 2 presents our recommended usage for the templates.

2.1.1. Crawling template

Crawling template provides a code snippet that automatically traverses the webpage, performs a process when the page is loaded, and then moves to the next page. Our recommendation for this template is to store webpages in files or databases using the preprocesses required for the experiment. For example, it provides functionality to save a webpage as an MHTML file to include information on external content. This template is also helpful when automated preprocessing is required, such as creating a dataset from a webpage, performing a pre-inspection, or filtering irrelevant pages from the URLs.

2.1.2. Curation template

Curation template targets many tasks that require human decision-making, such as labeling web segments from crawled webpages with researchers' insights. This template provides the best practice that uses the DevTools API when users access the DOM inspector to label the webpage segment at the level of the HTML elements. This template helps to review and tag attributes on a webpage with multiple annotators.

Table 3
The data type of extraction results and conversion procedure for the measurements.

	Type of result	Longest common subsequence [19]	Matched text blocks [4,5]	IoU [20]
Readability.js	A single HTML element	Property of elements (trimmed multiline whitespaces)	Visible descendant elements	As is
DOM Distiller	HTML elements	Concatenated strings from attributes of elements	Visible elements	A wrapper of elements
Web2Text	A list of strings	Concatenated strings	Visible elements having a matched string	A wrapper of elements
BoilerNet	A list of element indices	Concatenated strings from attributes of elements	Visible elements	A wrapper of elements

2.1.3. Extraction template

Extraction template supports batch processing for webpages, such as the crawling template. This template also provides information on interacting with external programs such as other WebExtensions of the same web browser. For example, BoilerNet provides its WebExtension version. However, this version processes a single page on the current tab, only activated by the user's action (mouse click). We included a function for message passing to another WebExtension in this template so that we could automate BoilerNet's extraction process on web browsers. The preloaded extraction methods were Firefox Readability.js, Google Chrome DOM Distiller, and BoilerNet.

2.1.4. Evaluation template

Numerous performance metrics can be applied to webpage segments. Most quantitative metrics substitute a webpage for another form of data. For example, as a performance metric, the longest common subsequence (LCS) indicates the similarity of two strings. The F-score is calculated from the length of common subsequence between the text of relevant HTML nodes and the retrieved ones by a method. Thus, we must alter the webpage and our experimental results to pure text for the performance metric. As LCS compares only strings, non-textual contents, such as images and videos, are discarded. Meanwhile, Intersection over Union requires two displayed areas; we must alter our data to the relevant and retrieved areas on the webpage.

Furthermore, some metrics, such as human eye tracking, user impressions, and user behavior, require a controlled experiment and performance evaluation environment. These metrics require semi-automatic or manual webpage iterations and intricate event handling. For example, we might record a user's behavior only a few seconds after loading the webpage to measure the first impression. In this scenario, we should wait for the user to look at the webpage but prevent webpage navigation.

The Evaluation template supports the automation of the evaluation of webpages with the given performance metrics or environments. It can automatically load webpages, extract multiple data to perform calculations and measure data from users or other external sources (e.g., recording the user's behavior 5 s after webpage loading). The preloaded performance metrics are the F-score of the longest common subsequence and matched text blocks, and Intersection over Union values (Table 3) for the results of Firefox Readability.js, Google Chrome DOM Distiller, Web2Text, and BoilerNet.

2.2. Sample code snippets analysis

Listing 1 and 2 show an example of creating and assigning a new extraction task to TaskManager using the extraction template. The TaskManager creates a Task that iterates webpages and sends data to the content script whenever a page is loaded. Using *RequestCallback*, developers can add variables or declare a new function to convey the content script (Listing 1). The extraction result is sent using the *sendResponse* function in *onMessageCallback* (Listing 2). Consequently, the extraction result is returned as the response parameter in the response callback. The result is checked, reprocessed, and sent to the databases in this function.

3. Illustrative examples

This section demonstrates the application of our framework. Fig. 2 shows the experimentation process that uses our templates in the order of crawling, curation, extraction, and evaluation.

Preparation The extension options page (Fig. 3) shows the DB interface server and MHTML parsing server URLs (Fig. 3(a)). The target webpages and execution buttons are displayed if the server connection is successful (Fig. 3(b)). Users can execute each template using the configuration shown on this page (Fig. 3(c)). The user ID identifies the annotators for the curation tasks (Fig. 3(d)).

Crawling: This template opens and stores webpages. We used MongoDB for the page metadata and PostgreSQL for the raw data and captured the page in an MHTML file. Before saving the webpage, we numbered the HTML elements according to the order in the DOM tree.

Curation: This template loads crawled webpages on the MHTML parsing server. Annotators can choose an element from the page's main content using the DevTools extension (see Fig. 4). The selected answer with the tag is stored in MongoDB.

Extraction: As Readabilty.js provides a standalone module, it can be directly executed in the content script. The execution results for the JSON objects were stored in MongoDB.

Evaluation: The results of Readabilty.js and the annotated answers are a single HTML element. Hence, we can apply the following measurements: textContent attributes for LCS, visible descendants for matched text blocks, and areas for IoU.

4. Impact

This framework aims to present the process of making a webpage a valuable dataset and making the process visible in a web browser. Webpage collection studies have used programming APIs, command-line tools, or headless browser environments [8,24,25]. Web browsers are less suitable than technical tools or programming APIs for handling many webpages. However, gathering and examining webpages with a web browser is worthwhile because most webpages are created for user interactions on a web browser. Webpages displayed in familiar forms can easily induce users' behavior and responses, allowing researchers to obtain new insights [10–12,14,26,27].

One of the research subjects that is difficult to access with a few experts is the case across multiple regions and languages. The Web is used worldwide, and webpages of their own environment may be created in each region according to language or culture. Because many datasets are mainly English or Western languages, it is difficult to find datasets that contain the characteristics of various regions and languages. Many studies on webpages

```
// Tasks are managed by TaskManager (singleton instance)
const manager = TaskManager.getInstance()

// Request type 1: requesting pre-loaded Mozilla Readability.js
// Send an object instead of callback function to RequestCallback,
// the pre-loaded method runs in current tab and loaded pageId.
const requestMozilla = {
  extractorName: 'mozilla'
}

// Request type 2: requesting customed task with taskName
// Send a request with a callback function requestCallback as a function.
// The task manager assings this callback to the Task instance (In here, Extraction).
const requestCustomedTaskWithCallback = async (tabId: number, pid: string) => {
  return await requestContentScript(tabId, {
    taskName: TASK_NAME_EXTRACTION,
    extractorName: 'mozilla_customed',
    id: pid,
  })
}

const requestCallback = requestMozilla // or can be requestCustomedTaskWithCallback

// Response handling with responseCallback
const responseCallback = async (task: Extraction, response: any) => {
  console.log("Extraction response", response, task)
  const { status, extractorName, data, error, pid } = response
  if (status === 'ok' && !task.config.watch) {
    await POST(LocalConfig.getInstance().host, `extraction/result/${extractorName
      }/${pid}`, data)
  } else if (status !== 'ok') {
    console.error("[EXTRACTION] Error from content script response pid:", pid, error)
  }
}

// Assign Task via TaskManager
const task = await manager.create<Extraction>(
  target.pages, Extraction, config, requestCallback, responseCallback)
task.start()
```

Listing 1. Extraction template code snippet in the background script. Example of creating an Extraction Task in the background context by defining request and response callback

of low-resource languages have had difficulty creating datasets owing to a small number of webpages or a lack of experimental participants [12]. We used this framework to construct a new webpage dataset in multiple languages [16]. In this dataset, the webpages are annotated with recruited participants. They are ordinary people who do not know about complicated web technologies but can surely read webpages and recognize their segments. We gathered webpages from eight languages (English, Korean, Japanese, Chinese, French, Indonesian, Russian, and Arabic from Saudi Arabian webpages) with a browser extension based on the framework (Table 4). They annotated the wrapper HTML elements in the title, menu, and main content.

We also implemented batch processes using the Extraction and Evaluation templates to examine the performance of the well-known methods on the multilingual dataset (Table 5). Each method was measured using three preloaded metrics from the evaluation template. The results demonstrated the performance of the latest reader modes and boilerplate removal methods in various regions and languages. For example, the high recall and low precision of the LCS indicate that it extracted a larger main content area than anticipated.

The low recall value in the matched text blocks of the DOM Distiller resulted from the modification of adjacent blocks, that is, ⟨UL⟩, ⟨OL⟩, and ⟨DL⟩. Thus, the matched text blocks are inappropriate for evaluating the methods that create new elements from the original DOM structure. The IoU values indicate how each method can extract a visible area, including images and videos. For example, the low IoU value of Web2Text means it cannot handle non-text HTML elements. A more detailed description of the datasets is provided in the code repository.

5. Conclusions

We presented an open-source WebExtension-based framework for webpage segmentation methods. Researchers and developers can extend this framework to build a tool for crawling

```
chrome.runtime.onMessage.addListener((message, sender, sendResponse) => {
  const { taskName } = message
  // tasks are distinguished by taskName
  if (taskName === TASK_NAME_EXTRACTION) {
    // message contains request from the background script
    const { extractorName, id: pageId } = message

    // perform procedure for corresponding extractor
    if (extractorName === 'mozilla') {
      const result = extractWithMozilla(document)
      sendResponse({
        status: 'ok',
        extractorName,
        data: JSON.stringify(result),
        pid: pageId
      })
    }
    else if (extractorName === 'put_some_method_name') {
      // ...
    }

    return true // Inform the background script that there will be a response.
  }
})
```

Listing 2. Extraction template code snippets in the content script. Example of responding to the Extraction Task request from the background context using Message Passing.

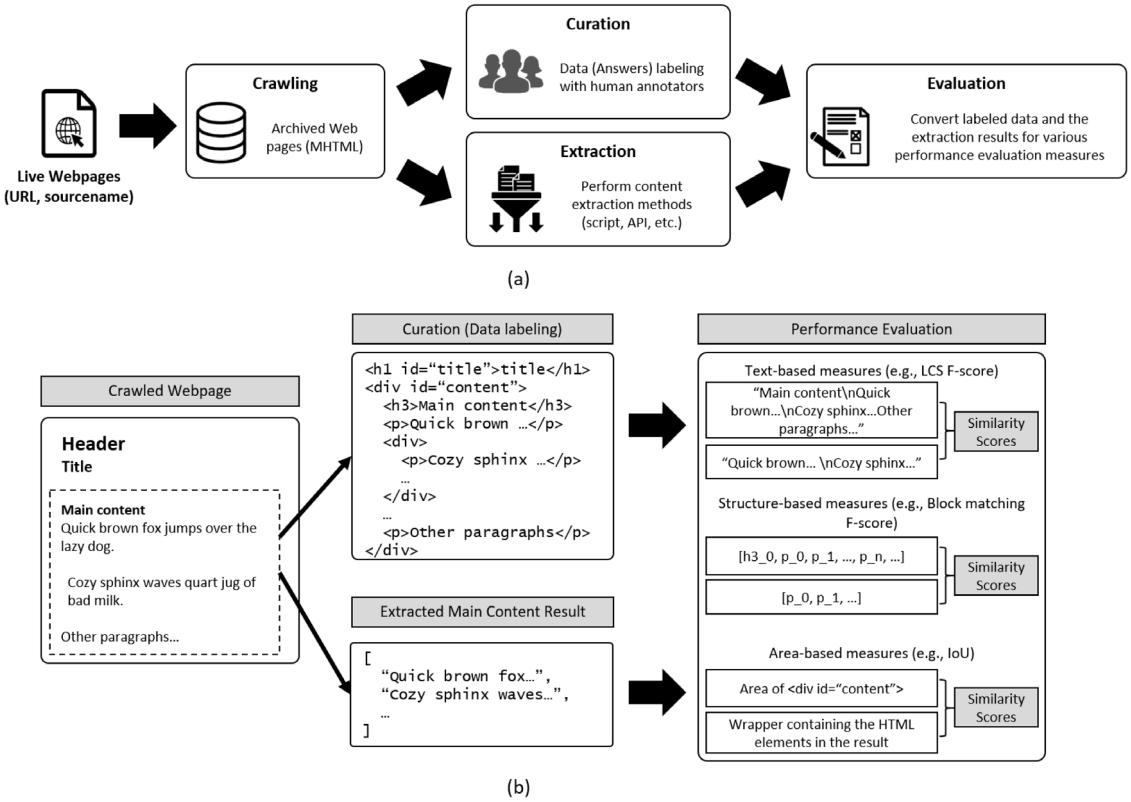

Fig. 2. Framework demonstration: the experimentation process of the webpage segmentation methods for the main content. (a) Overview. (b) Detailed Example.

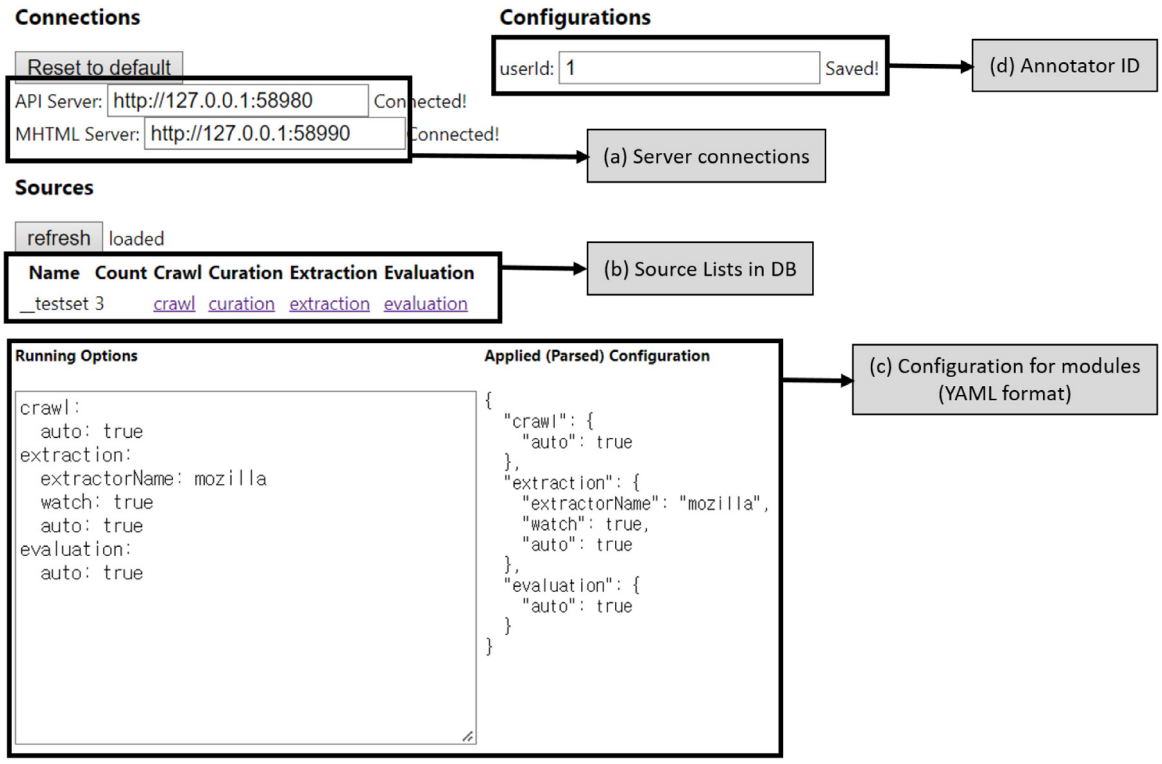

Fig. 3. Options page of the demonstration.

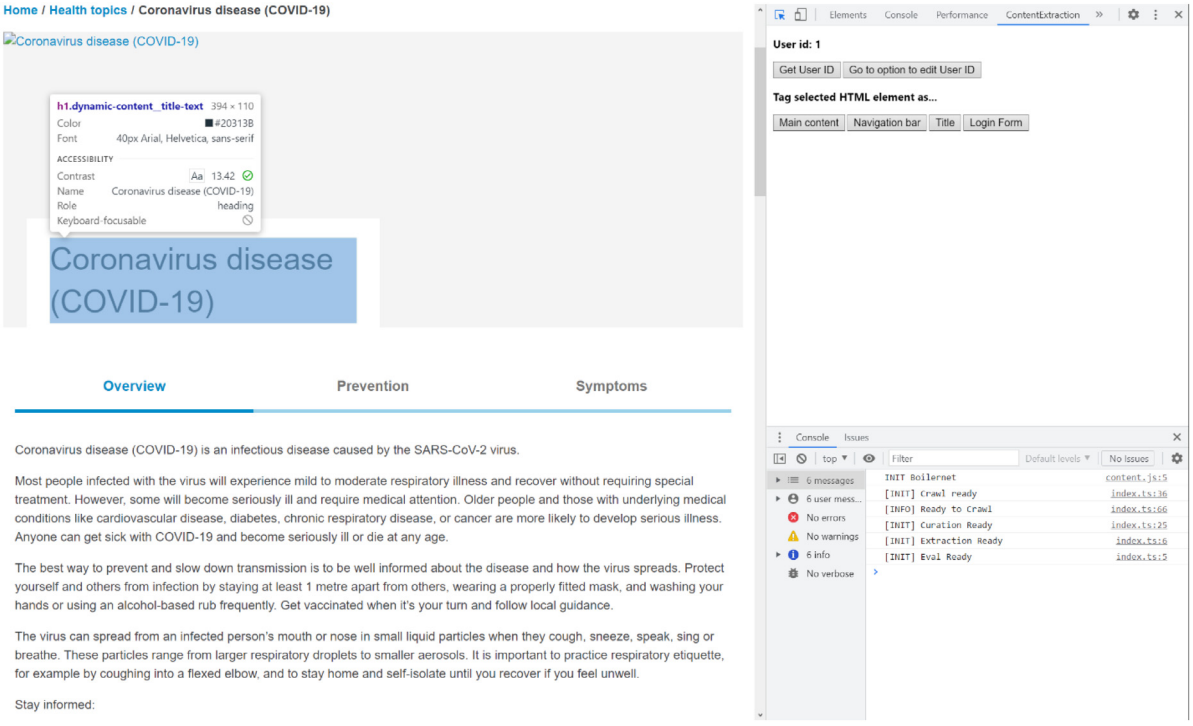

Fig. 4. DevTools in the Curation mode. The selected element (highlighted in blue) is tagged and sent to the server by clicking the buttons on the right panel.

and curating webpages and evaluating methods using various performance metrics in a browser environment.

Owing to WebExtension-based architecture, this framework enables researchers to collect and process webpages in their existing or new dataset.

In particular, non-expert participants can easily contribute to the dataset because they are already familiar with rendered webpages on web browsers. Without background knowledge, such as DOM tree and CSS selectors, the WebExtension tool can provide a WYSIWYG-style interface for handling webpage segments.

Table 4
GoogleTrends and Baidu datasets.

Name	URLs	Saved	Readable
GoogleTrends-2017	12720	390	285
GoogleTrends-2020	10560	388	240
GoogleTrends-2020-KR	296	43	21
GoogleTrends-2020-JP	450	47	24
GoogleTrends-2020-ID	900	50	31
GoogleTrends-2020-FR	1580	97	39
GoogleTrends-2020-RU	890	95	48
GoogleTrends-2020-SA	260	97	43
baidu-2020 (CN)	1990	193	53

Table 5
Performance evaluation results of preloaded methods and metrics. Keywords from GoogleTrends-2020 used, except on EN (2017 and 2020) and CN (Baidu 2020 year keywords) datasets.

	Readability.js							DOM distiller						
	LCS-F1			block-F1			IoU	LCS-F1			block-F1			IoU
	Precision	Recall	F1	Precision	Recall	F1		Precision	Recall	F1	Precision	Recall	F1	
EN(2017)	.761	.868	.743	.571	.686	.580	.581	.760	.881	.749	.705	.289	.339	.581
EN(2020)	.796	.905	.776	.646	.766	.634	.567	.762	.886	.751	.747	.369	.399	.569
KR	.783	.905	.784	.514	.511	.511	.527	.831	.813	.753	.848	.427	.492	.667
JP	.585	.988	.662	.429	.625	.448	.512	.753	.921	.742	.697	.353	.365	.556
CN	.757	.939	.787	.525	.540	.519	.606	.862	.962	.876	.869	.613	.653	.719
FR	.736	.911	.729	.531	.673	.534	.475	.793	.879	.757	.730	.239	.294	.517
SA	.803	.919	.775	.630	.751	.641	.602	.769	.917	.748	.760	.414	.420	.523
ID	.830	.914	.817	.640	.636	.593	.545	.878	.909	.837	.820	.263	.316	.737
RU	.843	.887	.773	.721	.729	.687	.664	.886	.882	.812	.849	.384	.433	.655
Avg.	.775	.897	.760	.598	.698	.595	.574	.784	.891	.766	.750	.351	.391	.592
	BoilerNet							Web2Text						
	Precision	Recall	F1	Precision	Recall	F1	IoU	Precision	Recall	F1	Precision	Recall	F1	IoU
EN(2017)	.680	.951	.751	.708	.811	.696	.529	.564	.939	.651	.553	.411	.382	.342
EN(2020)	.684	.952	.748	.699	.817	.680	.500	.600	.959	.677	.636	.810	.628	.337
KR	.846	.952	.887	.898	.807	.834	.770	.672	1.00	.771	.864	.716	.742	.528
JP	.459	.694	.477	.447	.588	.406	.284	.663	.946	.714	.701	.550	.485	.597
CN	.647	.721	.667	.595	.579	.530	.515	.878	.936	.889	.873	.722	.756	.783
FR	.690	.921	.755	.735	.775	.703	.546	.644	.996	.739	.696	.824	.679	.405
SA	.680	.881	.696	.663	.736	.605	.444	.624	.931	.690	.740	.688	.597	.421
ID	.843	.963	.853	.849	.681	.684	.711	.694	.939	.746	.709	.658	.610	.450
RU	.780	.940	.828	.763	.806	.716	.640	.663	.917	.739	.737	.703	.658	.494
Avg.	.689	.922	.745	.702	.779	.671	.528	.621	.948	.695	.647	.629	.547	.404

It is also beneficial in the visual representation of the dataset and result. Most webpages are designed to be rendered on web browsers. Thus, even if researchers do not highly expert in web techniques, they can easily examine their data and segmentation results on webpages rendered on browsers.

Our framework considers the scenario in that researchers conduct collaborative procedures with multiple participants. By distributing web extensions to participants and providing server APIs, researchers can collect and manage data and segmentation results from clients to a server. So, participants can collaboratively experiment according to the web extension's guidance displayed on the browser with minimal pre-training. It relieves the researchers' burden to expand the scale of the experiment.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Jaehyuk Cha reports financial support was provided by Korea Ministry of Science and ICT.

Data availability

The dataset is available in IEEE Dataport openly.

Acknowledgments

Funding

This work was supported by National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) (Nos. 2016R1A2B4016591 and 2018R1A5A7059549).

References

[1] Han H, Tokuda T. A personal web information/knowledge retrieval system. Front Artif Intell Appl 2008;166:338–45.
[2] Yesilada Y. Web page segmentation: A review. eMINE technical report deliverable 0 (D0). Middle East Tech Univ Northern Cyprus Campus 2011;1–39.
[3] Fayzrakhmanov RR, Sallinger E, Spencer B, Furche T, Gottlob G. Browserless web data extraction: challenges and opportunities. In: Proceedings of the world wide web conference. 2018, p. 1095–104. http://dx.doi.org/10.1145/3178876.3186008.
[4] Vogels T, Ganea OE, Eickhoff C. Web2text: deep structured boilerplate removal. In: Proceedings of the 40th European conference on information retrieval. 2018, p. 167–79. http://dx.doi.org/10.1007/978-3-319-76941-7_13.
[5] Leonhardt J, An A, Khosla M. Boilerplate removal using a neural sequence labeling model. In: Companion proceedings of the world wide web conference. 2020, p. 226–9. http://dx.doi.org/10.1145/3366424.3383547.
[6] Cai D, Yu S, Wen JR, Ma WY. VIPS: A vision-based page segmentation algorithm. Beijing microsoft research asia technical report (MSR-TR-2003-79), 2003, p. 1–29, http://dx.doi.org/MSR-TR-2003-79.
[7] Wu G, Li L, Hu X, Wu X. Web news extraction via path ratios. In: Proceedings of the 22nd international conference on information and

knowledge management. 2013, p. 2059–68. http://dx.doi.org/10.1145/2505515.2505558.
[8] Barbaresi A. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In: Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing. 2021, p. 122–31. http://dx.doi.org/10.18653/v1/2021.acl-demo.15.
[9] Štrimaitis R, Stefanovič P, Ramanauskaite S, Slotkiene A. Financial context news sentiment analysis for the Lithuanian language. Appl Sci 2021;11(10). http://dx.doi.org/10.3390/app11104443.
[10] Martínez-González JA, Álvarez-Albelo CD. Influence of site personalization and first impression on young consumers' loyalty to tourism websites. Sustainability 2021;13(3):1–18. http://dx.doi.org/10.3390/su13031425.
[11] Wagner G, Schramm-Klein H, Steinmann S. Online retailing across e-channels and e-channel touchpoints: Empirical studies of consumer behavior in the multichannel e-commerce environment. J Bus Res 2020;107:256–70. http://dx.doi.org/10.1016/j.jbusres.2018.10.048.
[12] Jung G, Han S, Kim H, Kim K, Cha J. Extracting the main content of web pages using the First Impression Area. IEEE Access 2022;10:129958–69. http://dx.doi.org/10.1109/ACCESS.2022.3229080.
[13] Wu YH, Chang CH. Multi-task neural sequence labeling for zero-shot cross-language boilerplate removal. In: Proceedings of the 20th IEEE/WIC/ACM international conference on web intelligence and intelligent agent technology. 2021, p. 326–34. http://dx.doi.org/10.1145/3486622.3493938.
[14] Wan H, Ji W, Wu G, Jia X, Zhan X, Yuan M, et al. A novel webpage layout aesthetic evaluation model for quantifying webpage layout design. Inform Sci 2021;576:589–608. http://dx.doi.org/10.1016/j.ins.2021.06.071.
[15] Li Q, Morris M, Fourney A, Larson K. The impact of web browser reader views on reading speed and user experience. In: Proceedings of the 2019 CHI conference on human factors in computing systems. 2019, https://doi.org/10.1145.
[16] Jung G. Multilingual datasets for main content extraction from web pages, IEEE dataport, 2022. http://dx.doi.org/10.21227/rj0q-t583.
[17] Mozilla Foundation. Github - mozilla/readability: A standalone version of the readability lib. 2023, https://github.com/mozilla/readability, [accessed 31 2023].
[18] Chromiun projects, github - chromium/dom-distiller: distills the DOM. 2023, https://github.com/chromium/dom-distiller, [accessed 31 January 23].
[19] Sun F, Song D, Liao L. DOM based content extraction via text density. In: Proceedings of the 34th international ACM SIGIR conference on research and development in information retrieval. 2011, p. 245–54. http://dx.doi.org/10.1145/2009916.2009952.
[20] Lin TY, Maire M, Belongie S, Hays J, Perona P, Ramanan D, et al. Microsoft COCO: common objects in context. In: Proceedings of the 13th European conference on computer vision. 2014, p. 740–55. http://dx.doi.org/10.1007/978-3-319-10602-1_48.
[21] Alarte J, Silva J, Tamarit S. What web template extractor should I use? A benchmarking and comparison for five template extractors. ACM Trans Web 2019;13(2):1–19. http://dx.doi.org/10.1145/3316810.
[22] Alarte J, Insa D, Silva J, Tamarit S. A collection of website benchmarks labelled for template detection and content extraction. In: Proceedings of las XV Jornadas sobre Programación y Lenguajes. 2015, p. 1–10.
[23] Baroni M, Chantree F, Kilgarriff A, Sharoff S. Cleaneval: A competition for cleaning web pages. In: Proceedings of the 6th international conference on language resources and evaluation. 2008, p. 638–43.
[24] Lejeune G, Brixtel R, Doucet A, Lucas N. DAnIEL: language independent character-based news surveillance. In: Proceedings of 8th international conference on NLP. 2012, p. 64–75. http://dx.doi.org/10.1007/978-3-642-33983-7_7.
[25] Velloso RP, Dorneles CF. Extracting records from the web using a signal processing approach. In: Proceedings of the 26th international conference on information and knowledge management. 2017, p. 197–206. http://dx.doi.org/10.1145/3132847.3132875.
[26] Schenkman BN, Jönsson FU. Aesthetics and preferences of web pages. Behav Inf Technol 2000;19(5):367–77. http://dx.doi.org/10.1080/014492900750000063.
[27] Dong Y, Lee KP. 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 2008;2(2):19–30.