An Empirical Comparison of Web Content Extraction Algorithms

Janek Bevendorff
Bauhaus-Universität Weimar
Weimar, Germany

Sanket Gupta
Bauhaus-Universität Weimar
Weimar, Germany

Johannes Kiesel
Bauhaus-Universität Weimar
Weimar, Germany

Benno Stein
Bauhaus-Universität Weimar
Weimar, Germany

ABSTRACT

Main content extraction from web pages—sometimes also called boilerplate removal—has been a research topic for over two decades. Yet despite web pages being delivered in a machine-readable markup format, extracting the actual content is still a challenge today. Even with the latest HTML5 standard, which defines many semantic elements to mark content areas, web page authors do not always use semantic markup correctly or to its full potential, making it hard for automated systems to extract the relevant information. A high-precision, high-recall content extraction is crucial for downstream applications such as search engines, AI language tools, distraction-free reader modes in users' browsers, and other general assistive technologies. For such a fundamental task, however, surprisingly few openly available extraction systems or training and benchmarking datasets exist. Even less research has gone into the rigorous evaluation and a true apples-to-apples comparison of the few extraction systems that do exist. To get a better grasp on the current state of the art in the field, we combine and clean eight existing human-labeled web content extraction datasets. On the combined dataset, we evaluate 14 competitive main content extraction systems and five baseline approaches. Finally, we build three ensembles as new state-of-the-art extraction baselines. We find that the performance of existing systems is quite genre-dependent and no single extractor performs best on all types of web pages.

CCS CONCEPTS

• **Applied computing** → **Document analysis**; • **Information systems** → **Data extraction and integration**.

KEYWORDS

Main Content Extraction, Boilerplate Removal, Web Data Extraction

ACM Reference Format:
Janek Bevendorff, Sanket Gupta, Johannes Kiesel, and Benno Stein. 2023. An Empirical Comparison of Web Content Extraction Algorithms. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23), July 23–27, 2023, Taipei, Taiwan.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3539618.3591920

SIGIR '23, July 23–27, 2023, Taipei, Taiwan

ACM ISBN 978-1-4503-9408-6/23/07.
https://doi.org/10.1145/3539618.3591920

1 INTRODUCTION

Since its public release in 1991, the World Wide Web has become humanity's largest open information source with roughly two billion web sites and about 60 billion indexed pages [5, 33]. At the base level, this information is presented as machine-readable markup, but most of it must nevertheless be considered *unstructured*, which poses a big challenge to information extraction and retrieval systems, or general convenience and accessibility tools further down the line. Besides classical IR systems, an accurate extraction of text from web pages has become increasingly crucial also for data-hungry AI language technologies in recent years (although for that specific use case, an individual page's extraction precision may not be too important and some noise may actually be desireable).

Unfortunately, web pages contain not only the primary information which directly satisfies a visitor's information need and which we call the *main content*. Most often, they further contain headers and footers with branding and copyright information, blocks of navigational links, advertisements, and other secondary information. This secondary information, although not always irrelevant, is often referred to as *boilerplate content*. The distinction between main content and boilerplate content is not necessarily clear or may even warrant more than only two classes (e.g., do user comments below an article qualify as main content?). Authors of content extraction systems often use their own intuition of what they regard as boilerplate content. Hence, varying definitions can be found in the literature, such as "ads, hyperlink lists, navigation, previews of other articles, banners, etc." [32], "navigational elements, templates, and advertisements" [23], "irrelevant information such as copyright notices, advertising, links to sponsors, etc." [12], or "elements which constitute noise for the application of the Web data" [29]. For the sake of a search engine indexing a page, the main content can be assumed to be either the central article text of a page (if one exists) or anything else which is not part of the website's repeated template. For information retrieval, one could thus define the main content as the part of the page that most visitors would expect to see if they came from elsewhere, such as a search engine. For this paper, however, we are relying on existing data, so the implementation of this definition is, for the most part, out of our hands and we have to go with what the original annotators saw as main content.

Plenty of main content extraction systems have been developed over the years, but only few are openly available [4] and even fewer have been evaluated under comparable and reproducible conditions from which general conclusions can be drawn. Not only the availability of the systems themselves is an issue, also the availability of (consistently) annotated datasets of adequate size is. Attempts at developing benchmarking suites for extraction systems

have been made, though they also suffer greatly from this lack of data and deeper formal analysis thereof (see Section 2).

In this paper, we (1) combine, clean, and unify existing web content extraction datasets; (2) categorize pages by their potential extraction complexity and propose a simple page-level classification approach for approximating these categories without a ground truth; (3) perform and review a systematic benchmark of existing extraction systems on the combined dataset (taking into account the complexity classes); and (4) develop three ensemble content extractors based on the outputs of the individual systems, which outperform the current state of the art.

The new combined dataset and all code written for this study is publicly available.[1]

2 RELATED WORK

The task of main content extraction has been first described as "Body Text Extraction" by Finn et al. [12] in 2001. Since then, researchers and practitioners have presented several algorithms to tackle the task, of which the most notable are summarized in our review (Section 2.1). Moreover, several other approaches have been developed for closely related tasks (Section 2.2). Despite this attention to the task in the literature, a thorough reproducibility study such as this one has been missing so far (Section 2.3).

2.1 Main Content Extraction Algorithms

One can broadly distinguish main content extraction algorithms based on heuristics from those based on machine learning.

Heuristic approaches. Heuristic approaches use either a single measure or a list of heuristic rules, often in the form of a tree, to identify one or more blocks of main content. They are generally more efficient than machine learning approaches, but rely heavily on human expertise in designing the rules. Since efficiency if often key when applying main content extraction algorithms and training data is scarce, heuristic approaches are still in wide use today.

The first category of heuristic approaches is based on the assumption that the main content's markup contains fewer HTML tags than that of boilerplate regions in the HTML code. The first approach, BTE [12], falls in this category. It finds the single region with fewest HTML elements per text using a plateau-finding algorithm. Gottron [14] later extended the algorithm to find multiple plateaus. Following the same core idea but using a different approach, Gottron [15] encodes HTML documents as so-called Code-Content Vectors (CCV) where each token is either "code" or "content." The individual vector components are then successively smoothed ("blurred") to find consecutive areas where most of the tokens are content. Similarly, Weninger et al. [36]'s CETR method detects peaks in the line-wise tag ratio distribution.

The second category of heuristic approaches use several rules to identify regions of main content. Sun et al. [31] still calculate a page's "text density," but also takes the DOM tree into account. Insa et al. [18] calculate the "words-leaves-ratio" (WLR) for each node, i.e., the ratio between a node's content words and the number of child nodes. Other approaches, like jusText by Pomikálek [29], first segment the HTML tree into regions and then employ heuristics to classify each region. Rule-based approaches are in widespread use today, like the Readability extractor,[2] which is implemented and directly available in Mozilla's Firefox web browser to provide distraction-free reading. Readability was originally developed for a (now-defunct) bookmarking service [37] and, despite its focus on article-like pages, is reported to be effective for at least 22 % of the Alexa top-1000 websites [13].

Since heuristic approaches are fast, some approaches are combinations of others. Trafilatura [6] is mainly based on high-precision heuristic rules in the form of XPath expressions, but uses jusText and Readability as fallbacks. News-please [16] is (besides being a crawler) also a meta extractor, implementing rules that specify how different other approaches are employed to extract specific information from news portals.

Machine learning approaches. The second paradigm employs machine learning to classify regions into main content or boilerplate. The earliest approach, Boilerpipe [23] uses structure, text, and text density features to classify individual regions of the HTML code.[3] Newer approaches use sequence labeling methods and deep neural networks. For example, Web2Text [32] uses a hidden Markov model to classify regions depending on the classification of neighboring regions, and a convolutional neural network based on regional features. BoilerNet [24] and SemText [38] both employ LSTMs. However, the training data for main content extraction is scarce. For example, Leonhardt et al. [24] used a dataset of only 180 pages. Moreover, some new approaches render web pages in the browser to extract visual features (e.g., Jung et al. [20]), which requires additional resources (JavaScript and style sheet files, images, …) that are not available for most datasets.

While the above-mentioned algorithms employ only page-level features (which Kohlschütter et al. argue are quite sufficient in most cases), Endrédy and Novák [11]'s GoldMiner algorithm learns global features that repeat on multiple pages under a domain. A similar algorithm was presented later by Alarte et al. [3], who built on previous work on domain-specific HTML template detection. However, the available benchmark datasets for main content extraction are too small to test such algorithms in a fair manner.

2.2 Tasks Related to Main Content Extraction

A closely related task is that of extracting clean—but not necessarily main—text, the prime example being the OSCAR corpus [2], which is created using the project's Ungoliant [1] extractor. Ungoliant uses a FastText [19] language classifier to identify natural-language text in large amounts of web data from the Common Crawl. Although extraction precision is a concern, the focus is clearly on recall.

A more specific task than main content extraction is the task of list web page extraction, which focuses on information extraction from list-like websites, such as online shops. A large resource for this task is PLAtE [30], a dataset of list websites annotated using crowd-sourcing. To distinguish between article-like and list-like web pages and to choose the correct extraction algorithm for each, Nguyen-Hoang et al. [27] developed a CNN-based genre classification using image features from a page rendering.

[1] https://github.com/chatnoir-eu/web-content-extraction-benchmark

[2] https://github.com/masukomi/arc90-readability

[3] https://commoncrawl.org/

Yet more specific is the task of structured information extraction, which is to extract information from tabular or otherwise structured pieces of information. For this task, the largest available resource is the Structured Web Data Extraction (SWDE) dataset [17], a silver standard corpus labeled using regular expressions. State-of-the-art models employ deep neural architectures such as CNNs / LSTMs [39] or Transformers [10, 34]. WebSRC [9] employs semantic embeddings of the page structure.

A more general task than main content extraction is the task of web page segmentation. Based on their review of the literature, Kiesel et al. [21] define web page segments as a "part of a web page containing those elements that belong together as per agreement among a majority of viewers." Though segmentation does not have a concept of "main" content, content within a segment is semantically coherent and should hence be either all main content or not. Segmentation can therefore be seen as a generalization of main content extraction. In fact, some main content extraction datasets provide more than binary labels for parts of a page, though they are rarely used in training or evaluation. Unlike most of the analyzed approaches to main content extraction, web page segmentation approaches tend to focus more on visual and layout analysis, going back to the classic VIPS algorithm of 2003 [8], which is still competitive today [22].

2.3 Lack of Reproducibility Studies

Despite the many approaches developed for main content extraction over the decades, comparably little effort has been spent on developing resources for reproducible experiments. The CleanEval [7] shared task was held in 2007 to improve the state of HTML page cleaning. The task had five submissions and no follow-up tasks were ever organized. The task's dataset with as few as 738 annotated pages, however, has become its long-lasting legacy and has since been used for evaluating almost all main content extractors to date. A few other datasets have been made available since, but to the best of our knowledge this paper describes the first effort to combine them to a larger dataset.

Perhaps caused by the lack of sufficiently large datasets, there is also a lack of systematic reviews of the literature on main content extraction. Though the CleanEval dataset is frequently used, few authors benchmark their systems against others using comparable metrics. Moreover, besides algorithms from the scientific literature, there are also many software libraries without associated research or white papers, which are rarely considered in the literature at all. The closest we could find to a survey and reproduction study is the one done by Alarte et al. [4], who reimplemented five systems and compare them on the CleanEval dataset. Lopukhin [26] benchmark six open source extraction libraries against their own proprietary extraction service, but use a custom dataset of only 181 pages.[4] The WEE benchmarking tool performs a similar benchmark of eight algorithms on the same small dataset.[5]

In conclusion, we find that previous studies of main content extraction mostly worked on inadequately small datasets. To remedy this problem, the paper at hand presents the largest reproduction study to date. We collect datasets with a combined size more than five times as large as CleanEval and evaluate a total of 14 main content extraction algorithms and five baselines.

[4]https://github.com/scrapinghub/article-extraction-benchmark
[5]https://github.com/Nootka-io/wee-benchmarking-tool

Table 1: Page counts of all datasets in total and split by extraction complexity quartile ranges (see Section 3.3).

	Total	By Extraction Complexity		
Dataset		**Low**	**Medium**	**High**
CETD	700	158 (23 %)	488 (70 %)	54 (8 %)
CleanEval	738	524 (71 %)	171 (23 %)	43 (6 %)
CleanPortalEval	71	0 (0 %)	41 (58 %)	30 (42 %)
Dragnet	1,379	139 (10 %)	694 (50 %)	546 (40 %)
Google-Trends-2017	180	25 (14 %)	95 (53 %)	60 (33 %)
L3S-GN1	621	74 (12 %)	368 (59 %)	179 (29 %)
Readability	115	57 (50 %)	36 (31 %)	22 (19 %)
Scrapinghub	181	24 (13 %)	96 (53 %)	61 (34 %)
Total	3,985	1,001 (25 %)	1,989 (50 %)	995 (25 %)

3 DATASET

We base our analysis on a unified and cleaned combination of eight different annotated datasets for web content extraction. In this section, we describe the individual datasets and steps taken to combine them into a single, larger dataset. Table 1 shows the final page counts of all datasets.

3.1 Source Datasets

The following eight publicly available datasets were used to create a larger combined dataset for web content extraction:

CleanEval. The CleanEval dataset was created for the 2007 CleanEval shared task [7]. Despite its age, it is still the de-facto standard dataset for main content extraction. We merged the development and final evaluation split, resulting in 797 source HTML pages and 738 ground truth text files (with minimal structural markup). The 59 excess HTML files without a ground truth were discarded. Besides the 738 English pages, there is also a Chinese CleanEval set, which we omitted.

CleanPortalEval. CleanPortalEval is an extension to CleanEval created by Endrédy and Novák [11], containing an additional 71 cases in the same format. The pages were taken from the online news portals of the BBC, MSNBC, the Wall Street Journal, and the Washington Post.

CETD. The abbreviation stands for *Content Extraction via Text Density* and the dataset was created for evaluating Sun et al. [31]'s density-based extractor. The dataset is divided into six main genre verticals with content from (1) Ars Technica, (2) the BBC, (3) the New York Times, (4) Wikipedia, (5) Yahoo!, and (6) "Chaos" (i.e., general content from Google News, Wordpress, Blogger). The corpus consists of 700 pairs (100 per vertical, 200 in Chaos) of source HTML pages and plain text files containing the ground truth.

Dragnet. The Dragnet dataset is the largest in this list and was created to accompany the Dragnet content extractor.[6] The dataset

[6]https://github.com/dragnet-org/dragnet

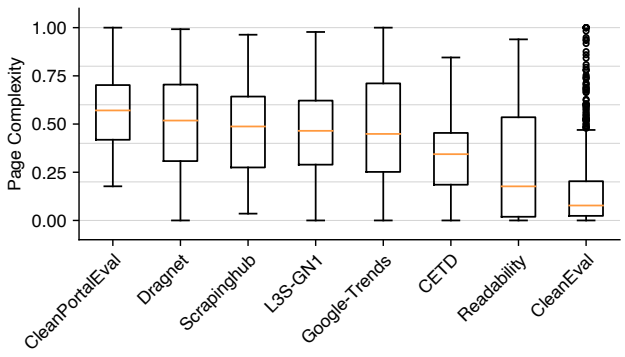

Figure 1: Median page complexity scores per dataset. Complexity is defined as the ratio between HTML text tokens and tokens in the ground truth. Whiskers mark 1.5 × the interquartile range. The CleanEval dataset is clearly an outlier.

consists of 1,381 HTML and text file pairs, similar to CleanEval (but without the structural markup in the ground truth).

L3S-GN1. This corpus was created by Kohlschütter et al. [23], the author of Boilerpipe, and it contains 621 pages. Unlike the other corpora, the ground truth is given in the form of annotated HTML pages. The leaf nodes are wrapped in `span` elements with CSS classes indicating the leaf's class. There are five labels indicating different levels of content relevance from main heading to user comments, as well as one negative label.

Google-Trends-2017. The Google-Trends-2017 dataset was created by the Boilernet authors [24] and contains 180 annotated HTML pages. The labels are specified as CSS classes on the leaf nodes, but unlike in the L3S-GN1 dataset, they are only binary.

Readability. The Readability dataset is a test collection for Mozilla's Readability.js reader mode implementation.[7] It consists of 115 test cases with raw HTML pages and cleaned versions with only the main content and reduced markup.

Scrapinghub. This is the dataset created for the Zyte (formerly: Scrapinghub) article extraction benchmark [26]. It was designed to benchmark a proprietary web extraction service against other open source extractor and comes with 181 zipped HTML pages and their ground-truth text extracts as a single JSON file.

3.2 Data Preparation

For each dataset, we wrote an individual parser, which tries to detect the correct encoding of all pages and converts them into a common JSON format. Since some of the datasets come with only a plaintext truth, we used this as the lowest common denominator and converted the L3S-GN1, Google-Trends-2017, and Readability ground truths to simple plaintext. We also stripped the pseudo markup from the CleanEval and CleanPortalEval text files. To align the L3S-GN1 ground truth with the other datasets, we kept only text annotated with one of the first three positive labels.

Unfortunately, we noticed that many of the CleanEval ground truth files have uncorrectable encoding errors. In the Dragnet dataset, we had to correct several cases of text duplication in the truth files. There was also one such case in the CleanEval dataset, where almost the full file was duplicated, resulting in a cleaned text much longer than the original HTML. We also had to correct a few glaring issues in the raw HTML pages from the L3S-GN1 corpus, stemming from seemingly successful cross-site-scripting. These obscured the `</title>` end tags in some of the documents, so they looked totally empty to a correct parser. The issue was already corrected in the annotated HTML, but not in the raw files. We also noticed that particularly in the CETD set, some elements were annotated as main content that were loaded dynamically via JavaScript and therefore didn't exist in the source HTML (particularly DISQUS comments). As it was too much effort, we did not correct all of these cases (and a browser-based extractor could technically extract such content). We did, however, remove two pages from the Dragnet set, whose HTML was virtually empty.

3.3 Page Complexity

The complexity of a page may inform the choice of the correct extractor and has an influence on its performance. A simple page without boilerplate can be extracted even using a simple XPath expression with high precision and high recall. The more boilerplate there is, however, the more difficult an accurate extraction becomes. We therefore categorize pages based on a simple complexity heuristic. We define the page complexity c as

$$c = 1 - \frac{|\{t \in T : truth(t) = 1\}|}{|T|},$$

where T is a multiset of DOM text tokens and $truth(t)$ returns 1 if token t belongs to the ground truth. Negative scores resulting from a faulty ground truth with duplicated text or text not in the source HTML are clipped to zero. Figure 1 shows the median page complexity per dataset and the distribution. The CleanEval dataset is clearly the least complex of the datasets with outliers towards the top. Upon inspection, we found that the dataset does indeed contain many simplistic pages with little to no boilerplate that are often closer to plaintext documents than to modern web pages. Obviously, such pages need hardly any sophisticated extraction logic. The Readability dataset is also significantly less complex than others, probably due to its focus on article-like pages.

Complexity levels can be defined as quantile ranges on this complexity metric, but since in a real-world scenario there is—unlike in our analysis—no ground truth available to determine the complexity, one has to estimate it. To show that such an estimation is possible, we extracted the relative counts of `h1`–`h6`, `a`, `br`, `strong`, `em`, `div`, `p`, `ul`, and `table` elements, and the overall ratio of HTML element tokens to non-element tokens. We trained a logistic regression classifier on 25 % of the pages to separate page complexities into two balanced classes (split at the median), which achieved an accuracy of 80 % on the remaining pages. With such a high result from this simple model with only basic features, one can expect that more complex models with better features can estimate page complexity levels quite accurately before choosing an appropriate extraction method.

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

Table 2: Overview of the employed extractors.

Extractor	Ref.	Approach
Main content extractors		
BTE	[12]	Heuristic: HTML tag distribution
Goose3		Heuristic: rule-based
jusText	[29]	Heuristic: rule-based
Newspaper3k		Heuristic: rule-based (for news)
Readability		Heuristic: rule-based
Resiliparse		Heuristic: rule-based
Trafilatura	[6]	Heuristic: rule-based
news-please	[16]	Meta heuristic: rule-based (for news)
Boilerpipe	[23]	AI: text node classification
Dragnet	[28]	AI: text node classification
ExtractNet		AI: text node classification
Go DOM Distiller		AI: text node classification
BoilerNet	[24]	AI: sequence labeling (LSTM)
Web2Text	[32]	AI: sequence labeling (HMM+CNN)
Text conversion tools		
BS4		Text node concatenation
lxml Cleaner		Text node concatenation
XPath Text		Text node concatenation
html_text		Text rendering
inscriptis	[35]	Text rendering

4 EXTRACTOR EVALUATION

We evaluated the performance of 14 dedicated main content extraction systems from the literature and the open source community. Further, we tested five HTML-to-text conversion tools without main-content or boilerplate semantics as a baseline comparison. Finally, we compiled three ensemble models from these systems.

4.1 Main Content Extractors

Based on our review of related work (Section 2) and an extensive online search, we selected 14 main content extractors for our comparison. We distinguish heuristic extractors and extractors based on machine learning ("AI" in Table 2).

BTE. The Body Text Extraction algorithm [12] is a heuristic algorithm based on the HTML tag distribution. Specifically, it employs the cumulative tag distribution within a document and finds the largest plateau (region of fewest tags per text). The corresponding text is extracted as main content. We use the Python implementation by Pomikálek [29].

Goose3. This extractor is a heuristic algorithm based on handcrafted rules. Since the original Java (before 2011) and later Scala project was abandoned, we use a Python reimplementation that is still being maintained.[8]

jusText. The justText extractor [29] is a heuristic algorithm based on hand-crafted rules. In a first pass, HTML element blocks (separated using a list of block-level elements) are classified using heuristics on block size, link density, and stopword density. In a second pass, blocks not directly classified as content or boilerplate are heuristically classified based on their surrounding blocks. We use a fork of the original project that is still being maintained.[9]

Newspaper3k. This extractor is a heuristic algorithm (and web scraper) based on hand-crafted rules, specifically targeting news article pages. We use its Python implementation.[10]

Readability. This extractor is another heuristic algorithm based on hand-crafted rules. It is implemented in Firefox for providing a "reader view" that removes distracting elements. On news article pages it can be enabled inside the browser with one click. We use a Python reimplementation.[11]

Resiliparse. The Resiliparse HTML2Text extractor is a heuristic algorithm based on hand-crafted rules that was created by this paper's first author. It is based on tag rules and regular expressions and it focuses on extraction precision and speed.[12]

Trafilatura. The Trafilatura extractor [6] is a heuristic algorithm (and web scraper) that uses hand-crafted rules. It employs a cascade of XPath queries for finding the main content, falling back to jusText and Readability (see above) should the extraction fail.

news-please. This extractor is a heuristic meta extractor (and web scraper) that uses hand-crafted rules to combine the output of other extractors, specifically targeting news article pages [16]. We use the default rules that employ the Readability and Newspaper3k extractors (see above).

Boilerpipe. This extractor by Kohlschütter et al. [23] is a machine learning algorithm that classifies text blocks (sequences of text without tags except links) into content and boilerplate. It is one of the earliest machine-learning-based main content extractors and uses decision trees constructed on structural, shallow text, and text density features. We use a Python wrapper[13] around the original Java implementation.

Dragnet. The Dragnet extractor [28] is a machine learning algorithm that combines the approaches of Boilerpipe and CETR [36]. It uses text (density) features on block level and word frequencies from `class` and `id` attributes. The latest version also includes features from Readability.

ExtractNet. This extractor is based on Dragnet, but uses models trained on different data and also extracts structured information from article pages, such as title, author, date etc.[14]

Go DOM Distiller. This extractor is a machine learning algorithm loosely based on Boilerpipe. Analogously to Readability, the original DOM Distiller is part of Chromium-based browsers for providing a "reader view" that removes distracting elements. The feature can be enabled inside the browser with a hidden setting under `chrome://flags/#enable-reader-mode`. We use a Golang port of the original Java implementation via a command line interface.[15]

[8] https://github.com/goose3/goose3

[9] https://pypi.org/project/jusText/

[10] https://newspaper.readthedocs.io/en/latest/

[11] https://github.com/buriy/python-readability

[12] https://resiliparse.chatnoir.eu/en/stable/api/extract/html2text.html

[13] https://github.com/slaveofcode/boilerpipe3

[14] https://github.com/currentslab/extractnet

[15] https://github.com/markusmobius/go-domdistiller

BoilerNet. This extractor by Leonhardt et al. [24] is a machine learning algorithm that treats extraction as a sequence labeling task. Specifically, it employs an architecture with one LSTM layer. According to the authors, the model outperforms Web2Text (below) in block classification F_1 on CleanEval and Google-Trends-2017. We used the provided code to retrain the model on a split of the Google-Trends-2017 corpus according to the reproduction steps given by the authors. The reference implementation requires Tensorflow 2.1, but we were able to load the trained model with a newer version for the inference step.

Web2Text. This extractor by Vogels et al. [32] is a machine learning algorithm that treats extraction as a sequence labeling task. It uses a hidden Markov model on top of a CNN-based representation of blocks (DOM leaf nodes). Structural features from each block and its neighboring blocks are fed to the CNN. The reference implementation comes with a pre- and post-processing tool written in Scala and the CNN part written in Python using Tensorflow 1.15. We use the provided pre-trained model, but since the individual tools are written in different languages and Tensorflow 1.15 requires Python 3.7, we could interface with it only via the command line, making it by far the slowest tool. Due to the repeated loading of the JVM and the trained model, classification of a single example took several seconds, which would be unacceptable for practical real-world applications.

4.2 Text Conversion Tools

Besides these main content extraction systems, we also use five HTML-to-text conversion tools as baselines. We distinguish tools that simply concatenate text nodes and those that "render" a web page trying to retain basic layout information (see Table 2).

BS4. This tool uses the Beautiful Soup (version 4) Python DOM parsing library[16] to extract all text from a web page. We concatenate all text nodes, but exclude `script`, `style`, and `noscript` elements. This baseline aims for full recall but low precision.

lxml Cleaner. This tool is a component of the lxml XML / HTML parser library for Python.[17] It uses a set of built-in and user-supplied rules for stripping "offending" elements and attributes from the DOM to produce an overall cleaner document. We defined a blacklist of common HTML5 elements that are likely to be part of the boilerplate (e.g., noise elements such as `object` or `iframe`, but also `footer` or `aside`) and use it together with lxml's own cleaning rules to render a cleaned text representation of the document. Afterwards, BeautifulSoup is used to extract text nodes from the cleaned DOM.

XPath Text. As the simplest baseline, we use the XPath expression `//body[1]//*[not(name() = "script") and not(name() = "style")]/text()` to extract all DOM text nodes under the `body` element, except for scripts and styles. This baseline aims for a similar result as BS4, but is slightly less robust, as it would be unable to extract pages without an explicit `body` element.

[16] https://pypi.org/project/beautifulsoup4/
[17] https://lxml.de/4.9/apidoc/lxml.html.clean.html

html_text. This tool[18] strips invisible elements, normalizes white space and tries to retain a basic block layout. It is based on lxml Cleaner.

inscriptis. This tool by Weichselbraun [35] tries to retain most of a page's layout information in the output text. It attempts to preserve table layouts and CSS attributes (e.g., margin) without a time-intensive browser-based rendering.

4.3 Performance Measure

Since the ground truth for our combined dataset is unstructured text and not annotated HTML documents, we need to score the model performances with a text-based performance measure. We decided on using ROUGE-L [25], a relaxed measure for the token overlap between two texts that is popular for the evaluation of machine translation. We calculated also other metrics, such as the normalized Levenshtein distance or the Jaccard index, but found them to be highly correlated with ROUGE-L and hence we will omit them for the purpose of this paper.

ROUGE-L looks for the *Longest Common Subsequence* (LCS), which is the longest sequence of tokens that are in agreement between the two texts. The sequence may be interrupted as long as the token order is retained. In particular, we use ROUGE-LSum, which is the summary-level ROUGE-L across multiple sentences. The LSum-LCS precision between a reference text T of n sentences and a candidate text C is defined as

$$P_{lcs} = \frac{\sum_i^n LCS(T_i, , C)}{|C|_{\text{words}}}$$

and the recall, accordingly

$$R_{lcs} = \frac{\sum_i^n LCS(T_i, C)}{|T|_{\text{words}}}.$$

The F_1 measure is the symmetric harmonic mean of both.

We split sentences using NLTK's punkt tokenizer and tokenize the sentences into Unicode word tokens at white-space boundaries, which works well for most texts in the English-centric datasets.

4.4 Results

We find that, expectedly, almost all extractors perform reasonably well on pages of low extraction complexity (for which most tokens are main content). Since it is easy to achieve both high precision and high recall on these pages, this also extends to the simple HTML extractors without main content semantics. In fact, at very low extraction complexity, these extractors perform better than many actual main content extractors due to their virtually perfect recall. On more complex pages, however, they fall off quickly. The median baseline performance over all datasets is relatively high (median baseline $F_1 = 0.738$), meaning that most pages consist primarily of main content and not only boilerplate. Figure 2 shows the F_1 distribution for all extractors at different levels of page extraction complexity. Table 3 lists in detail each model's mean and median performance in terms of ROUGE-LSum precision, recall, and F_1.

As can be seen, no single extractor performs best on all datasets at all complexity levels, but there are a few candidates tied for

[18] https://pypi.org/project/html-text/

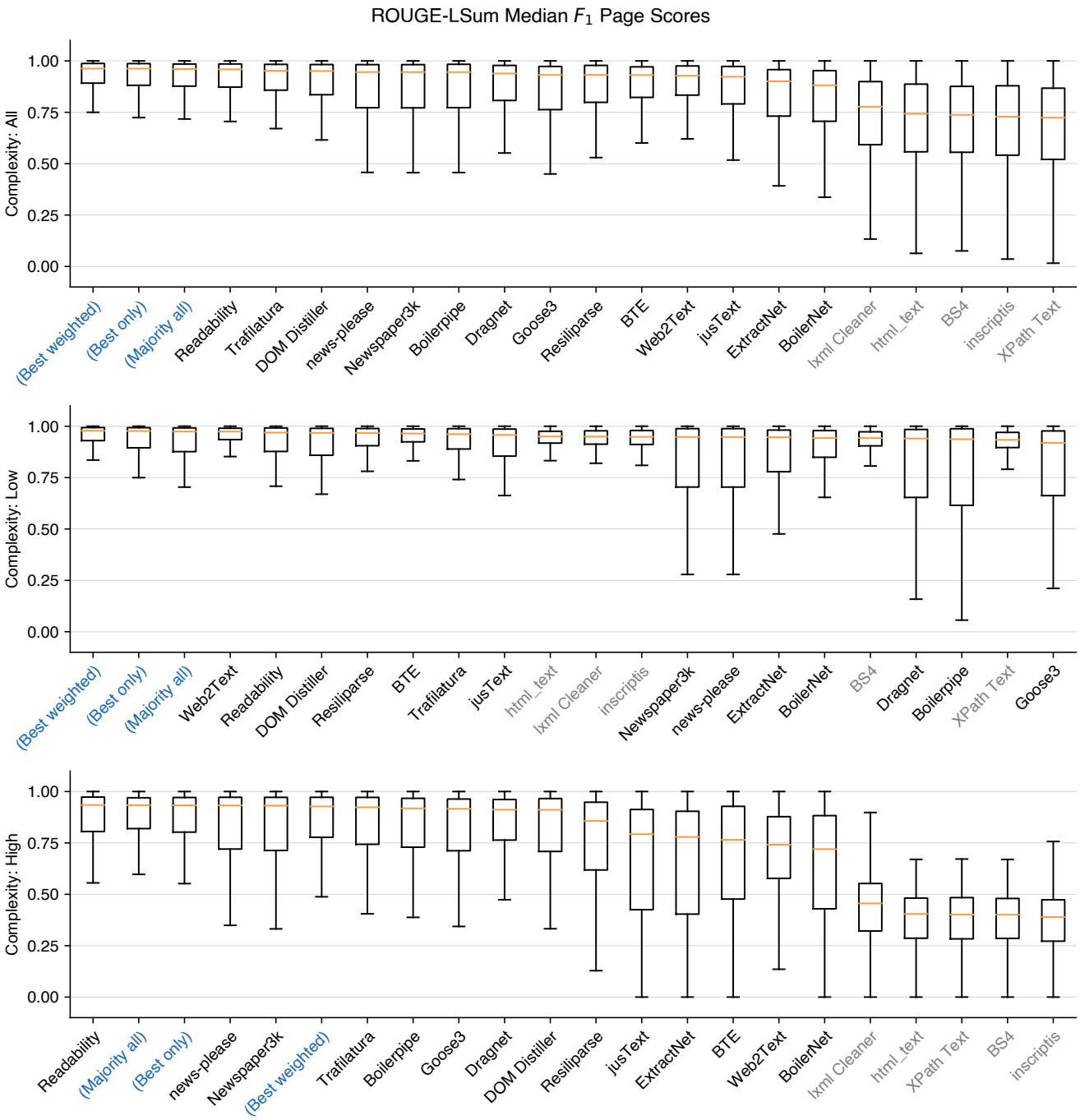

Figure 2: ROUGE-LSum F_1 scores of the extractors on (1) all pages and (2) pages with low (< 25th percentile) or (3) high extraction complexity (> 75th percentile). Boxes indicate the interquartile range (IQR or: "middle 50 %"), the whiskers the 1.5 × IQR spread. Ensembles are in parentheses and highlighted in blue, baselines in gray. Almost all models perform well on pages of low extraction complexity, including the baselines. At higher complexity, the baseline models fall off substantially, whereas the heuristic models in particular keep performing well. Although Web2Text also performs quite well on pages of low complexity, the performance of the neural models is in general surprisingly weak. All three majority ensembles beat the individual extractors, though differences among the top systems are very minor.

Table 3: Macro mean and median ROUGE-LSum precision, recall, and F_1 across all datasets and page complexity levels. The table is sorted in descending order of mean F_1. Ensemble extractors are written in parentheses and italics.

	Mean			Median		
Model	**Prec.**	**Recall**	**F_1**	**Prec.**	**Recall**	**F_1**
(Best weighted)	0.922	0.912	**0.899**	0.986	0.981	0.970
(Best only)	0.926	0.892	0.889	0.992	0.976	0.973
(Majority all)	**0.930**	0.879	0.885	0.996	0.971	**0.974**
Trafilatura	0.913	0.895	0.883	0.989	0.965	0.957
Readability	0.921	0.856	0.861	0.991	0.972	0.970
Resiliparse	0.863	0.901	0.859	0.940	0.993	0.942
DOM Distiller	0.894	0.864	0.858	0.983	0.970	0.959
Web2Text	0.797	0.944	0.841	0.885	0.984	0.917
Boilerpipe	0.908	0.825	0.834	0.973	0.966	0.946
Dragnet	0.901	0.810	0.823	0.980	0.950	0.943
BTE	0.796	0.897	0.817	0.927	0.965	0.936
Newspaper3k	0.896	0.803	0.816	0.994	0.961	0.958
news-please	0.895	0.802	0.815	0.994	0.961	0.958
Goose3	0.899	0.779	0.810	**0.999**	0.919	0.940
BoilerNet	0.840	0.816	0.798	0.944	0.938	0.895
ExtractNet	0.858	0.773	0.791	0.963	0.915	0.911
jusText	0.794	0.769	0.759	0.949	0.921	0.904
lxml Cleaner	0.615	0.964	0.717	0.670	0.995	0.798
html_text	0.567	**0.992**	0.683	0.506	**1.000**	0.667
BS4	0.563	0.990	0.680	0.506	0.997	0.669
inscriptis	0.557	0.990	0.673	0.483	**1.000**	0.649
XPath Text	0.550	0.965	0.664	0.510	0.997	0.674

first place: Readability has the highest median score ($F_1 = 0.970$, macro-median over datasets) and the lowest IQR spread. Trafilatura achieves the best overall mean performance (macro $F_1 = 0.883$, micro $F_1 = 0.867$). Web2Text is a direct runner-up in *micro*-average mean performance ($F_1 = 0.861$; score slightly inflated, see below). Goose3 achieves a very impressive median precision of 0.999. Unfortunately, it suffers badly from severe recall outliers, costing it the top spot in the ranking and making it the worst on pages of low complexity. In fact, most models have similar strengths and weaknesses which make for a relatively high variance across datasets and complexity levels. The only exception to the rule is Readability, which seems like a true jack of all trades, being the most robust model at all complexities.

Some models, like Goose3, suffer from heavy recall outliers with many answers that are either too short or entirely empty. The problem is particularly impactful at low page complexity, where many models are outperformed even by the baselines. We found that this can be attributed in some part to the Dragnet and CETD datasets. Excluding them from the evaluation alleviates these outliers to a degree. But since not all models suffer from this to the same extent, we cannot attribute the problem to the data alone. We did, however, notice problems with content duplication or passages in the ground truth not being present in the source HTML particularly in the Dragnet dataset earlier (see Section 3.2). This would explain some of the lower-than-expected recall scores, but not the (relatively) frequent zero scores.

An important observation to be made in this regard is that comparing the mean and median performances of all models reveals a very skewed, non-normal distribution of the scores. In fact, the frequencies of the scores of all models (apart from the baselines) follow roughly a power distribution with over-represented zero scores from empty answers. For visual comparison to the median score distribution in Figure 2, we show the global mean F_1 performance in Figure 3 and overlay the score bars with errors bars indicating the Q_2–Q_3 interquartile range. For many models (particularly at the top), the global mean falls only barely within this range. Hence one should be very careful when reporting the mean performance of a content extractor without showing the actual distribution of the scores (as is done often in the literature). With a distribution like this one, the median represents quite an optimistic measure of central tendency, whereas the arithmetic mean is much more conservative. For better understanding, we therefore report both.

It seems that overall, heuristic content extractors perform the best and are the most robust models across the board, whereas the large neural models perform surprisingly badly. Web2Text does indeed achieve the best mean performance, but this can be attributed mostly to a very good and consistent performance on pages of low complexity and in (small) part to being trained on CleanEval. It is worth mentioning that Web2Text and BoilerNet where trained on splits of the CleanEval and Google-Trends-2017 datasets, respectively, which are both part of the test collection. The inclusion of CleanEval in the evaluation does indeed inflate Web2Text's mean performance slightly (micro mean $F_1 = 0.861$) and excluding it moves it down one place in the ranking ($F_1 = 0.849$). However, since (a) the effect is very small, (b) CleanEval and Google-Trends-2017 make for only a small part of the whole dataset, and (c) neither model showed a particularly outstanding performance with or without a leak of training data, we decided to not make an exception in the evaluation procedure to keep things simple. Moreover, we were at all unable to reproduce the supposedly superior performance of BoilerNet from the original paper, although this doesn't necessarily mean that the model's performance itself is worse than previously reported. (Part of) this discrepancy could well be a result of the different evaluation method, where instead of comparing individual DOM elements, we measure performance on the final text output. In general, the neural models apparently do not deliver on the most complex pages, for which they were primarily designed (in fact, they both perform quite poorly as can be seen in Figure 2). Considering their computational complexity, it begs the question whether their application is a worthwhile investment or if a simpler heuristic model isn't a better choice.

4.5 Ensemble Extractors

To see whether we can actually improve the extraction quality, we defined the following three ensembles on top of the results of the individual extraction systems discussed previously.

Majority Vote. We performed a simple HTML-to-text conversion of all text nodes (excluding the `script`, `style`, and `noscript`) of the document and split the result into white-space delimited tokens. Then for each token t_i and for all extraction systems, we

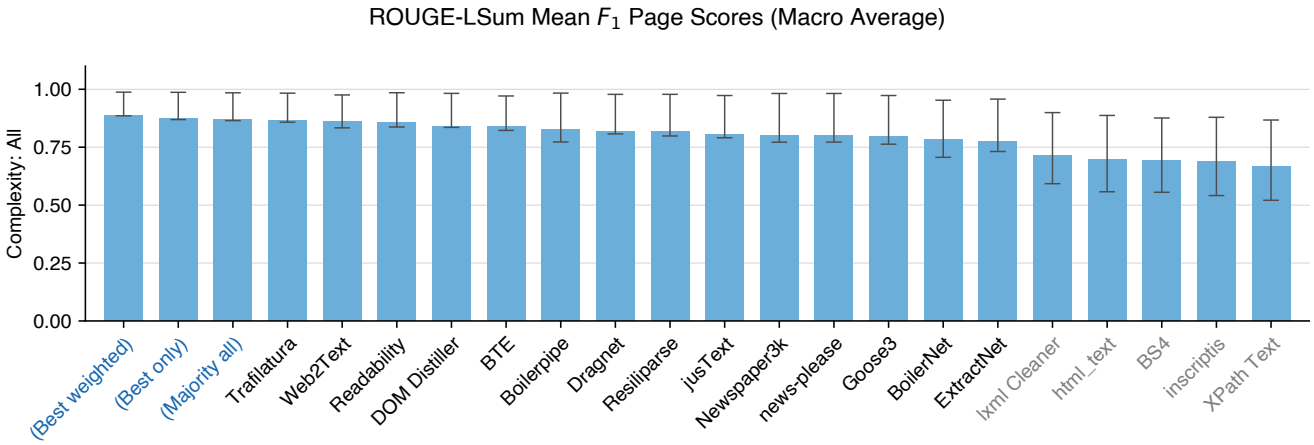

Figure 3: Mean extraction performance. The error bars indicate the interquartile range (IQR), i.e., the middle 50 % of ROUGE-LSum F_1 scores. As a result from the score distribution, the mean performance is skewed towards the bottom with some bars falling only barely inside the IQR. Scores are macro averages over the datasets to compensate for the page count imbalance.

checked whether either the left n-gram $(t_{i-n+1}, \ldots, t_i)$ or the right n-gram $(t_i, \ldots, t_{i+n-1})$ appeared in the extractor's output. If this was the case, we recorded a "vote" for this token. In the end, we labeled all tokens as main content which had received votes from at least two thirds (66 %) of the extractors. As n-gram size, we chose 5 and we padded the texts with $n-1$ zero tokens on either side to also capture tokens at the beginning and at the end of a text.

Majority Vote Best. This ensemble works by the same principle as the simple majority voting ensemble, but we included only the models that are actual main content extractors and that performed best both in terms of mean F_1 performance and variance. The included models are: Readability, Trafilatura, Go DOM Distiller, Goose3, Web2Text, Resiliparse, BTE, justText, and Boilerpipe. We excluded BoilerNet, Newspaper3k, news-please, Dragnet, and ExtractNet due to their rather unstable performance (although the latter two performed well on complex pages).

Majority Vote Best (Weighted). For the third and final ensemble, we chose the same nine content extractors as above, but doubled the voting power of Readability and Trafilatura (most robust systems overall) and Goose3 (highest median precision).

All ensembles outperform the individual extractors in mean and median macro-average F_1 with the weighted vote ensemble being the strongest of the three (see Table 3). Only on the most complex pages does Readability outperform the simple majority vote by an edge (Figure 2, bottom). In general, the differences between the top models are very minor, but it shows that we can indeed improve the state of the art further by reducing the variance of the models and reducing outliers towards the bottom.

5 CONCLUSION

We performed an extensive literature research into existing web page main content extraction (or boilerplate removal) systems and annotated datasets for this task. We identified 14 state-of-the art extraction systems in the academic literature and in the open source community, as well as eight datasets. We cleaned the datasets of certain artifacts, such as duplicated ground truths, categorized the pages according to their extraction complexity, and combined them into a coherent gold-standard dataset for main content extraction from HTML web pages. While some of the datasets we found were quite recent, the majority of the web pages contained are rather old and the web has changed quite a bit since the datasets were assembled. Hence there is a clear need for more up-to-date (and larger) annotated datasets for main content extraction. Particularly the CleanEval dataset with a majority of overly simplistic pages should find a worthy successor. Our combined dataset is a small step in this direction, but no replacement for an entirely new large-scale gold-standard corpus of recent web pages.

We benchmarked the 14 main content extractors together with five HTML-to-text conversion baselines on all eight datasets and evaluated their ROUGE-LSum scores based on the ground-truth plain texts. Finally, we defined three different majority voting ensembles on the outputs of the individual extractors.

Apart from our ensemble systems, no single extractor clearly outperforms all the others. Most systems perform reasonably well on pages of low extraction complexity, but only few also perform well on more complex pages. Mean and median performance deviate substantially as all systems struggle with outlier pages. Of all tested systems, Readability performs the best, followed by other rule-based extractors. The deep neural extractors perform well on simple pages, but otherwise lag behind the heuristic models. While there is still some performance to be gained (as demonstrated by our ensemble methods), this challenges the assumption that more complex models are needed for a more accurate extraction. Combining multiple simple models may be a better way forward, instead. In any case, one should carefully choose the correct model for each page based on its estimated extraction complexity, as there is no one-size-fits-all extractor at the moment.

ACKNOWLEDGMENTS

This work has been funded by the European Commission under GA 101070014 (OpenWebSearch.eu)

REFERENCES

[1] Julie Abadji, Pedro Suàrez, Javier Ortiz, Laurent Romary, and Benoît Sagot. 2021. Ungoliant: An optimized pipeline for the generation of a very large-scale multilingual web corpus. In *Proceedings of the Workshop on Challenges in the Management of Large Corpora (CMLC-9) 2021. Limerick, 12 July 2021 (Online-Event)*. Leibniz-Institut für Deutsche Sprache, 1–9.

[2] Julien Abadji, Pedro Ortiz Suarez, Laurent Romary, and Benoît Sagot. 2022. Towards a Cleaner Document-Oriented Multilingual Crawled Corpus. (Jan. 2022). arXiv:2201.06642 [cs.CL]

[3] Julian Alarte, David Insa, Josep Silva, and Salvador Tamarit. 2018. Main Content Extraction from Heterogeneous Webpages. In *Web Information Systems Engineering – WISE 2018*. Springer International Publishing, 393–407.

[4] Julián Alarte, Josep Silva, and Salvador Tamarit. 2019. What Web Template Extractor Should I Use? A Benchmarking and Comparison for Five Template Extractors. *ACM Trans. Web* 13, 2 (March 2019), 1–19.

[5] Martin Armstrong. 2021. Infographic: How many websites are there? https://web.archive.org/web/20230131222529/https://www.statista.com/chart/19058/number-of-websites-online/ Captured: 31 Jan, 2023.

[6] Adrien Barbaresi. 2021. Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction. In *Proceedings of the Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL 2021 - System Demonstrations, Online, August 1-6, 2021*. Association for Computational Linguistics, Online, 122–131.

[7] Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. 2008. CleanEval: a competition for cleaning webpages. In *Lrec*. European Language Resources Association (ELRA), Marrakech, Morocco.

[8] Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. Extracting Content Structure for Web Pages Based on Visual Representation. In *Web Technologies and Applications, 5th Asian-Pacific Web Conference, APWeb 2003, Xian, China, April 23-25, 2002, Proceedings*. Springer Berlin Heidelberg, 406–417.

[9] Xingyu Chen, Zihan Zhao, Lu Chen, Danyang Zhang, Jiabao Ji, Ao Luo, Yuxuan Xiong, and Kai Yu. 2021. WebSRC: A Dataset for Web-Based Structural Reading Comprehension. (Jan. 2021). arXiv:2101.09465 [cs.CL]

[10] Xiang Deng, Prashant Shiralkar, Colin Lockard, Binxuan Huang, and Huan Sun. 2022. DOM-LM: Learning Generalizable Representations for HTML Documents. (Jan. 2022). arXiv:2201.10608 [cs.CL]

[11] István Endrédy and Attila Novák. 2013. More Effective Boilerplate Removal - the GoldMiner Algorithm. *Polibits - Research journal on Computer science and computer engineering* 48 (2013), 79–83.

[12] Aidan Finn, Nicholas Kushmerick, and Barry Smyth. 2001. Fact or Fiction: Content Classification for Digital Libraries. In *Proceedings of the Second DELOS Network of Excellence Workshop on Personalisation and Recommender Systems in Digital Libraries, DELOS 2001, Dublin, Ireland, June 18-20, 2001*. ERCIM.

[13] Mohammad Ghasemisharif, Peter Snyder, Andrius Aucinas, and Benjamin Livshits. 2019. SpeedReader: Reader Mode Made Fast and Private. In *The World Wide Web Conference* (San Francisco, CA, USA) *(WWW '19)*. Association for Computing Machinery, New York, NY, USA, 526–537.

[14] Thomas Gottron. 2007. Evaluating content extraction on HTML documents. In *Proceedings of the 2nd International Conference on Internet Technologies and Applications (ITA'07)*. researchgate.net, 123–132.

[15] Thomas Gottron. 2008. Content Code Blurring: A New Approach to Content Extraction. In *19th International Workshop on Database and Expert Systems Applications (DEXA 2008), 1-5 September 2008, Turin, Italy*. ieeexplore.ieee.org, 29–33.

[16] Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please : a Generic News Crawler and Extractor. In *Everything changes, everything stays the same : Understanding Information Spaces; Proceedings of the 15th International Symposium of Information Science (ISI 2017), Berlin, Germany, 13th-15th March 2017 (Schriften zur Informationswissenschaft)*. Verlag Werner Hülsbusch, 218–223.

[17] Qiang Hao, Rui Cai, Yanwei Pang, and Lei Zhang. 2011. From one tree to a forest: a unified solution for structured web data extraction. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 775–784.

[18] David Insa, Josep Silva, and Salvador Tamarit. 2013. Using the words/leafs ratio in the DOM tree for content extraction. *The Journal of Logic and Algebraic Programming* 82, 8 (Nov. 2013), 311–325.

[19] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*. Association for Computational Linguistics, Valencia, Spain, 427–431.

[20] Geunseong Jung, Sungjae Han, Hansung Kim, Kwanguk Kim, and Jaehyuk Cha. 2021. Don't read, just look: Main content extraction from web pages using visual features. (Oct. 2021). arXiv:2110.14164 [cs.IR]

[21] Johannes Kiesel, Florian Kneist, Lars Meyer, Kristof Komlossy, Benno Stein, and Martin Potthast. 2020. Web Page Segmentation Revisited: Evaluation Framework and Dataset. In *29th ACM International Conference on Information and Knowledge Management (CIKM 2020)*. ACM, 3047–3054.

[22] Johannes Kiesel, Lars Meyer, Florian Kneist, Benno Stein, and Martin Potthast. 2021. An Empirical Comparison of Web Page Segmentation Algorithms. In *Advances in Information Retrieval. 43rd European Conference on IR Research (ECIR 2021) (Lecture Notes in Computer Science, Vol. 12657)*. Springer, Berlin Heidelberg New York, 62–74.

[23] Christian Kohlschütter, Peter Fankhauser, and Wolfgang Nejdl. 2010. Boilerplate detection using shallow text features. In *Proceedings of the third ACM international conference on Web search and data mining* (New York, New York, USA) *(WSDM '10)*. Association for Computing Machinery, New York, NY, USA, 441–450.

[24] Jurek Leonhardt, Avishek Anand, and Megha Khosla. 2020. Boilerplate Removal using a Neural Sequence Labeling Model. In *Companion Proceedings of the Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 226–229.

[25] Chin-Yew Lin. 2004. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*. Association for Computational Linguistics, Barcelona, Spain, 74–81.

[26] Konstantin Lopukhin. 2021. *In-depth analysis and evaluation on the quality of article body extraction.* Technical Report. Zyte.

[27] Bao-Dai Nguyen-Hoang, Bao-Tran Pham-Hong, Yiping Jin, and Phu T V Le. 2018. Genre-Oriented Web Content Extraction with Deep Convolutional Neural Networks and Statistical Methods. In *Proceedings of the 32nd Pacific Asia Conference on Language, Information and Computation*. Association for Computational Linguistics, Hong Kong.

[28] Matthew E Peters and Dan Lecocq. 2013. Content extraction using diverse feature sets. In *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume* (Rio de Janeiro, Brazil) *(WWW '13 Companion)*. International World Wide Web Conferences Steering Committee / {ACM}, New York, NY, USA, 89–90.

[29] Jan Pomikálek. 2011. *Removing Boilerplate and Duplicate Content from Web Corpora.* Ph. D. Dissertation. Masaryk University Brno.

[30] Aidan San, Jan Bakus, Colin Lockard, David Ciemiewicz, Yangfeng Ji, Sandeep Atluri, Kevin Small, and Heba Elfardy. 2022. PLAtE: A Large-scale Dataset for List Page Web Extraction. (May 2022). arXiv:2205.12386 [cs.CL]

[31] Fei Sun, Dandan Song, and Lejian Liao. 2011. DOM based content extraction via text density. In *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011* (Beijing, China) *(SIGIR '11)*. Association for Computing Machinery, New York, NY, USA, 245–254.

[32] Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. 2018. Web2Text: Deep Structured Boilerplate Removal. In *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*. Springer International Publishing, 167–179.

[33] Michael Völske, Janek Bevendorff, Johannes Kiesel, Benno Stein, Maik Fröbe, Matthias Hagen, and Martin Potthast. 2021. Web Archive Analytics. In *50. Jahrestagung der Gesellschaft für Informatik, INFORMATIK 2020 (Lecture Notes in Informatics, LNI, Vol. P-307)*. Gesellschaft für Informatik, GI, 61–72.

[34] Qifan Wang, Yi Fang, Anirudh Ravula, Fuli Feng, Xiaojun Quan, and Dongfang Liu. 2022. WebFormer: The Web-page Transformer for Structure Information Extraction. In *Proceedings of the ACM Web Conference 2022* (Virtual Event, Lyon, France) *(WWW '22)*. Association for Computing Machinery, New York, NY, USA, 3124–3133.

[35] Albert Weichselbraun. 2021. Inscriptis - A Python-based HTML to text conversion library optimized for knowledge extraction from the Web. *Journal of open source software* 6, 66 (Oct. 2021), 3557.

[36] Tim Weninger, William H Hsu, and Jiawei Han. 2010. CETR: Content extraction via tag ratios. In *Proceedings of the 19th international conference on World wide web* (Raleigh, North Carolina, USA) *(WWW '10)*. Association for Computing Machinery, New York, NY, USA, 971–980.

[37] the free encyclopedia Wikipedia. 2022. Readability (service). https://en.wikipedia.org/wiki/Readability_(service) Accessed: 18 Feb, 2023.

[38] Hao Zhang and Jie Wang. 2021. Boilerplate Detection via Semantic Classification of TextBlocks. In *International Joint Conference on Neural Networks (IJCNN'21)*. IEEE, 1–8.

[39] Yichao Zhou, Ying Sheng, Nguyen Vo, Nick Edmonds, and Sandeep Tata. 2021. Simplified DOM Trees for Transferable Attribute Extraction from the Web. (Jan. 2021). arXiv:2101.02415 [cs.LG]