Trafilatura: A Web Scraping Library and Command-Line Tool for Text Discovery and Extraction

Adrien Barbaresi
Center for Digital Lexicography of German (ZDL)
Berlin-Brandenburg Academy of Sciences (BBAW)
Jgerstr. 22-23, 10117 Berlin, Germany
barbaresi@bbaw.de

Abstract

An essential operation in web corpus construction consists in retaining the desired content while discarding the rest. Another challenge finding one's way through websites. This article introduces a text discovery and extraction tool published under open-source license. Its installation and use is straightforward, notably from Python and on the command-line. The software allows for main text, comments and metadata extraction, while also providing building blocks for web crawling tasks. A comparative evaluation on real-world data also shows its interest as well as the performance of other available solutions.

The contributions of this paper are threefold: it references the software, features a benchmark, and provides a meaningful baseline for similar tasks. The tool performs significantly better than other open-source solutions in this evaluation and in external benchmarks.

1 Introduction

1.1 Gathering texts from the Web

As useful monolingual text corpora across languages are highly relevant for the NLP community (Caswell et al., 2020), web corpora seem to be a natural way to gather language data. Corpus construction usually involves "crawling, downloading, 'cleaning' and de-duplicating the data, then linguistically annotating it and loading it into a corpus query tool" (Kilgarriff, 2007). However, although text is ubiquitous on the Web, drawing accurate information from web pages can be difficult. In addition, the vastly increasing variety of corpora, text types and use cases makes it more and more difficult to assess the usefulness and appropriateness of certain web texts for given research objectives. As a result, content adequacy, focus and quality need to be evaluated after the downloads (Baroni et al., 2009).

A significant challenge lies in the ability to extract and pre-process web data to meet scientific expectations with respect to text quality. An essential operation in corpus construction consists in retaining the desired content while discarding the rest, a task carrying various names referring to specific subtasks or to pre-processing as a whole: web scraping, boilerplate removal, web page segmentation, web page cleaning, template extraction, or content extraction. This step is sometimes overlooked although it involves a series of design decisions and turning points in data processing. Depending on the purpose of data collection, adequate filtering and quality assessment can be crucial. It has a significant impact on a wide range of downstream applications like text analysis, information retrieval, link analysis, page adaptation to other terminals and screens, and especially natural language processing pipelines.

Another challenge is how to find one's way through the Web, notably as linguistic data are gathered by running targeted web crawlers (Scannell, 2007). As web crawling involves discarding much of the downloaded content (Olston and Najork, 2010), especially link filtering and prioritization can prove to be tricky for contexts in which data collection is just the first step of a project, so that time resources for this task are scarce. Data collection approaches using the CommonCrawl[1] have flourished as they allow for faster download and processing by skipping (or more precisely outsourcing) the crawling phase. Barring the fact that finding one's "own" way through the Web can be preferable, such data should not be used without forethought and exhaustive filtering. Beside the discovery of relevant websites, a major issue consists in selecting appropriate content after download and processing (Schäfer et al., 2013), which can be com-

[1] https://commoncrawl.org

plex due to unexpected machine-generated flaws and biases.

Finally, depending on the project's jurisdiction, legal aspects of retrieving and granting access to web documents can be unclear or restrictive. Boundaries of copyright law are not clear when it comes to corpus building (De Clercq and Perez, 2010) so that some corpus infrastructure projects leave it to users to decide what to do from a copyright standpoint (Benko, 2016). Copyright and intellectual property rights usually do not apply to resources such as language models or n-grams (Buck et al., 2014), so are shuffled sentences (Biemann et al., 2007). Web corpora focusing on manually selected sources under Creative Commons licenses have been built (Brunello, 2009; Lyding et al., 2014), although only a very small proportion of websites use them (Barbaresi and Würzner, 2014). Corpora based on machine-checked licenses have also been developed (Habernal et al., 2016), as well as systems to merge annotation with web parts from the CommonCrawl (Schäfer, 2016). Considering the progresses of annotation tools, is can be easier to retrieve documents directly from the Web or from archives and to process them to one's taste.

1.2 Research context

This effort is part of methods to derive information from web documents in order to build text databases for a lexicographic information platform (Geyken et al., 2017). Extracting and preprocessing web texts to the exacting standards of scientific research turned out to be a substantial challenge where existing open-source solutions were not entirely convincing in terms of accuracy, versatility, and ease of use. The current tool follows from earlier work on news and blog articles extraction (Barbaresi, 2015, 2016). Its packaging into a directly re-usable format generalizes the process and makes it available to the community, with thorough testing it has also become much more robust and versatile.

1.3 Contributions

Distinguishing between a whole page and the page's essential parts can help to alleviate many quality problems related to web text processing, notably by dealing with the noise caused by recurring elements (headers and footers, ads, links/blogroll, etc.). This can be particularly useful to de-duplicate recurring language samples. Tasks related to content extraction and language modeling also benefit from a cleaner text base. In the concrete case of linguistic and lexicographic research, it allows for content queries on meaningful parts of the documents.

The remainder of this article introduces a text extraction and web navigation tool published under open-source license. Its installation and use is straightforward, notably from Python and on the command-line. The software makes it easier to extract the main text, comments and metadata, while also providing building blocks for text discovery tasks such as web crawling. The following also entails a comparative evaluation of text extraction on real-world data. The contributions of this paper are thus threefold as it references the software, features a benchmark, and provides a fast, meaningful baseline for similar tasks.

2 State of the art

2.1 "A difficult IE problem"

Even before the "Web 2.0" paradigm with web pages assembling information from and for a variety of sources (notably the advertising industry), web pages have been known for their lack of focus on directly usable text content. Despite the quantity of pages following an article format where there is a main text to be found, web pages now accessible through archives cannot be expected to be easy to process: "Articles published on the WWW often contain extraneous clutter. Most articles consist of a main body which constitutes the relevant part of the particular page. [...] Identifying the main body of a web page in a general robust manner is a difficult information extraction problem." (Finn et al., 2001)

Web pages come in different shapes and sizes mostly because of the wide variety of platforms and content management systems, and not least because of varying reasons to publish and diverging goals followed during web publication. Web page structure is also constantly evolving from the perspective of standards. HTML 5 was first released in 2008 to provide support for multimedia and graphical elements. This standard streamlined syntax while retaining backward-compatibility. Web content extraction is also an active field of research in user experience, resulting from the need for higher download and rendering speeds as well as from a growing amount of "Web bloat" requiring the development of "reader modes" and "distillers"[2] for

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

web browsers (Ghasemisharif et al., 2019).

2.2 Wrappers

Data extraction has first been based on "wrappers" (now called "scrapers") which were mostly relying on manual design and tended to be brittle and hard to maintain (Crescenzi et al., 2001). These extraction procedures have also been used early on by blogs search engines (Glance et al., 2004). Since the genre of "web diaries" was established before the blogs in Japan, there have been attempts to target not only blog software but also regular pages (Nanno et al., 2004), in which the extraction of metadata also allows for a distinction based on heuristics. Regarding metadata extraction for pages in article form and blogs in particular, common targets include the title of the entry, the date, the author, the content, the number of comments, the archived link, and the trackback link (Glance et al., 2004); they can also aim at comments specifically (Mishne and Glance, 2006).

2.3 Generic web content extraction

Generic extraction techniques ground on Document Object Model (DOM) examination. An earlier, language-independent approach uses entropy measures applied to features, links, and content in order to discriminate among parts of a web page (Kao et al., 2004). Another notable technique, Visual Page Segmentation, applies heuristics to find visually grouped blocks (Cai et al., 2003). Other methods are based on style tree induction, that is detection of similarities of DOM trees on site-level (Yi et al., 2003; Vieira et al., 2006). Overall, efforts made to automatically generate wrappers have been centered on three different approaches (Guo et al., 2010): wrapper induction (e.g. building a grammar to parse a web page), sequence labeling (e.g. labeled examples or a schema of data in the page), and statistical analysis. This approach combined to the inspection of DOM tree characteristics (Wang et al., 2009; Guo et al., 2010) is a common ground to the information retrieval and computational linguistics communities, with the categorization of HTML elements and linguistic features (Ziegler and Skubacz, 2007) for the former and boilerplate removal for the latter.

The DOM considers a given HTML document as a tree structure whose nodes represent parts of the document to be operated on. Text, tag and/or link density have proven to be good indicators in order to select or discard content nodes, using the cumulative distribution of tags (Finn et al., 2001), or with approaches such as the content extraction via tag ratios (Weninger et al., 2010) and the content extraction via text density algorithms (Sun et al., 2011). Statistical selection of informative nodes through a combination of both methods proved more efficient on comparable datasets (Qureshi and Memon, 2012). The large majority of DOM-based approaches try to leverage semantic information conveyed by HTML tags, notably paragraphs (*p*) on which text-to-tag ratios are calculated (Carey and Manic, 2016), or tag ratios and semantic features from *id* and *class* attributes (Peters and Lecocq, 2013).

Machine learning approaches have also been used, whose interest generally consists in leveraging advances in classification tasks by treating a HTML document as a series of blocks to be classified. Relevant algorithms include conditional random fields learning header, text, and noisy blocks with markup-based, content-based, and document-related features (Spousta et al., 2008), support vector machines trained on linguistic, structural and visual features (Bauer et al., 2007), Naive Bayes (Pasternack and Roth, 2009), multi-layer perceptron based on paragraph-level features (Schäfer and Bildhauer, 2012), or logistic regressions (Peters and Lecocq, 2013). More recently, deep learning has also been used for similar classifications, e.g. the *Web2Text* system is based on convolutional neural networks learning combinations of DOM-based features (Vogels et al., 2018).

Despite the number of article on this topic, very few systems are open-source or freely available (Alarte et al., 2019).

2.4 Corpus linguistics and NLP

There are few comparable projects coming from the linguistics or natural language processing communities and focused on making software publicly available and usable. *Boilerpipe* uses shallow text features like word counts and link density with decision tree and SVM classifiers (Kohlschütter et al., 2010). *JusText* is based on length heuristics as well as link and stop word densities (Pomikálek, 2011). Both algorithms have been prevalent since their release and are now mostly used through their subsequent forks, as software needs to be kept up-to-date. More recent initiatives explicitly targeting

distiller

corpus creation feature the *Corpus Crawler*[3] or *Texrex*[4] (Schäfer, 2017), neither of which appears to be actively maintained.

An evaluation and discussion following from the Cleaneval initiative (Baroni et al., 2008) would put the topic back into focus, as content processing on the Web is affected by both time and geography. This benchmark could be elaborated on, results are not consistent in different languages and metrics sometime fail to capture the variable influence of extractors on downstream modules (Lejeune and Zhu, 2018). Often, tools are developed with particular page styles in mind, mostly from the English-speaking world (Barbaresi and Lejeune, 2020). For certain projects, customized scrapers which are adjusted to each website remain feasible (Krasselt et al., 2020). A generic approach can really save human time and resources, albeit at a certain cost in terms of accuracy depending on the context.

3 Introducing the Trafilatura tool

3.1 Features

Trafilatura is a web scraping tool for text discovery and retrieval which seamlessly downloads, parses, and scrapes web page data. It can crawl and discover texts within a website and process them accordingly. The extractor focuses on metadata, main body text and comments while preserving parts of the text formatting and page structure. It aims to be precise enough in order not to miss texts or to discard valid documents, as it must be robust but also reasonably fast. With these objectives in mind, Trafilatura is designed to run in production on millions of web documents.

The software features parallel online and offline processing: URLs, HTML files or parsed HTML trees can be used as input. Although straight output of Python variables is possible, conversion to various common output formats makes the software more versatile: plain text (minimal formatting), CSV (with metadata, tab-separated values), JSON (with metadata), XML and XML-TEI (for metadata and structure). The latter support for TEI format (following the recommendations of the Text Encoding Initiative) also includes a validator for Python which can be used apart from the extraction. The scraping and conversion parts also work with existing archives, Raw HTML documents can be retrieved from sources such as the CommonCrawl[5] or the Internet Archive[6].

In addition, download utilities are included, notably using a multi-threaded but "polite" processing of URL queues, i.e. time restrictions based on domain names. Persistent connections are managed by a connection pool, thus maintaining connections with websites to be scraped. The tool also entails web crawling capacities which provide accessible and fail-safe ways to gather data based on a series of target sites. First, support for sitemaps (XML and TXT formats) according to the sitemap protocol. Second, support for web feeds (ATOM, RDF and RSS formats) which make it possible to build a seamless news crawler. Third, crawling components to discover content. It can also manipulate URL lists, including filtering and prioritization based on site characteristics or language-aware heuristics based on internationalization.

The package provides a relatively light-weight and modular architecture, letting users choose the components they wish to include. It has been tested on Linux, MacOS and Windows, and can be used with Python, on the command-line, with R (using the *reticulate* adapter package), and through a graphical user interface. The package documentation also acts as a manual on web text collection.[7]

3.2 Extraction process

The extraction combines two acknowledged libraries, readability-lxml[8] and jusText[9], which are used as safety nets and fallbacks. Trafilatura's own extraction algorithm is based on a cascade of rule-based filters and content heuristics:
(1) Content delimitation is performed by XPath expressions targeting common HTML elements and attributes as well as idiosyncrasies of main content management systems, first in a negative perspective with the exclusion of unwanted parts of the HTML code (e.g. *<div class="nav">*) and next by centering on the desirable content (e.g. *<section id="entry-content">*). The same operations are performed for comments in case they are part of the extraction. The selected nodes of the HTML tree are then processed, i.e. checked for relevance (notably by element type, text length and link density) and simplified as to their HTML structure.

[3]https://github.com/google/corpuscrawler
[4]https://github.com/rsling/texrex
[5]https://commoncrawl.org/
[6]https://archive.org/
[7]https://trafilatura.readthedocs.io/
[8]https://github.com/buriy/python-readability
[9]https://github.com/miso-belica/jusText

(2) If fallbacks are selected and triggered by a possibly faulty extraction, the other algorithms are run as a backup. Since they proceed differently their approach is complementary. They notably apply heuristics based on line length, text-to-markup ratio, and position/depth of elements in the HTML tree. If applicable, the output of these generic algorithms is compared to the "homegrown" extraction and heuristics are applied to determine the most efficient extraction, mostly in terms of extraction length (all algorithms are fairly reliable, so much longer is better) and "impurities" (e.g. no media elements).
(3) In case nothing worked, a baseline extraction is run in order to look for "wild" text elements that most probably have been missed, which implies to discard unwanted parts and to look for any element which may contain useful text content (e.g. *div* elements without paragraphs).

The extraction is designed to be robust and modular and provides a trade-off between precision and recall in most settings. As a result, main texts and potential comments are returned, with optional preservation of structural elements (paragraphs, titles, lists, quotes, code, line breaks, in-line text formatting). Extraction of metadata is also included, that is by descending frequency title, site name, author, date, categories and tags. For date extraction the library acts like a wrapper around *htmldate* (Barbaresi, 2020), a module specifically developed for this task.

An optional language detection can be run on the extracted content, currently using the Compact Language Detector v3 (CLD3)[10], which can be subject to accuracy issues depending on text length and language modeling (Caswell et al., 2020).

4 Evaluation

4.1 Benchmark

The evaluation focuses on the ability to retain appropriate text spans and discarded unwanted clutter, a functionality shared by many tools. Text discovery and conversion utilities are not evaluated here as most solutions do not include them. The benchmark is run on a collection of 500 documents which are either typical for Internet articles (news outlets, blogs) or non-standard and thus harder to process. Some contain mixed content (lists, tables) and/or non-standard, not fully valid HTML code. They were selected from large collections of web pages in German, for the sake of completeness a few documents in other languages are added (notably English, French, other European languages, Chinese and Arabic). The evaluation is reproducible, the needed script and instructions are available from the project repository.[11]

Target of the extraction is the main content, which is usually the part displayed centrally, without the left or right bars, the header or the footer, but including potential titles and (optionally) comments. This task is also known as web scraping, boilerplate removal, DOM-based content extraction, main content identification, or web page cleaning.

Decisive document segments of a few words each are singled out, about three per webpage are manually annotated as being part of the main text or unwanted boilerplate. They represent parts of the documents which are of high significance in the perspective of working with the texts, most notably beginnings and endings, left/right columns, additional header, author or footer information such as imprints or addresses, as well as affiliated and social network links.

Raw text segments are expected as a way to evaluate extraction quality without markup, i.e. HTML to TXT in itself, which avoids indirectly factoring in how the systems deal with markup. The chosen segments are included in a single HTML element span and they do not imply trimming or normalizing spaces, which makes the output strings directly comparable. Due to the language diversity of the sample the documents entail different text encodings. Since not all packages deal with them in a similar way, the given input string is in Unicode format.

4.2 Tools

The benchmark focuses on the Python programming language, reportedly the most popular programming language in academia and one of the most popular overall.[12] A few algorithms have been ported from other languages such as Java and JavaScript, which contributes to giving an exhaustive yet incomplete panorama of available solutions overall. In case software packages are not actively maintained the most prominent usable fork is used.

First, these packages are provided for reference as they keep the structure intact but do not focus

[10]https://github.com/google/cld3

[11]https://github.com/adbar/trafilatura/

[12]https://spectrum.ieee.org/computing/software/the-top-programming-languages-2019

on main text extraction:

- *html2text*[13] converts HTML pages to Markup language
- *html_text*[14] converts HTML code to plain text
- *inscriptis*[15] converts HTML to text with a particular emphasis on nested tables

The following packages are strictly comparable as they focus on main text extraction:

- *boilerpy3*[16] is a Python version of the *boilerpipe* algorithm (Kohlschütter et al., 2010) for boilerplate removal and fulltext extraction
- *dragnet*[17] features machine-learning and combined approaches (Peters and Lecocq, 2013) but requires more dependencies and potentially fine-tuning: it is used with its default training data
- *goose3*[18] can extract information for embedded content but doesnt preserve markup
- *jusText*[19] is designed to preserve mainly text containing full sentences along with some markup, it has been explicitly developed to create linguistic resources (Pomikálek, 2011)
- *newspaper*[20] is mostly geared towards newspaper texts, provides additional functions but no structured text or comment extraction
- *news-please*[21] is a news crawler that extracts structured information (Hamborg et al., 2017)
- *readability-lxml*[22] cleans the page and preserves some markup

The tools are compared to the raw page source and to a meaningful baseline also provided by Trafilatura which consists in extracting all the text contained in JSON data or paragraph, code or quoting elements.

Two variants of Trafilatura are evaluated, first using its own algorithm and second including its fallback mechanisms based on external libraries.

[13]https://github.com/Alir3z4/html2text
[14]https://github.com/TeamHG-Memex/html-text
[15]https://github.com/weblyzard/inscriptis
[16]https://github.com/jmriebold/BoilerPy3
[17]https://github.com/dragnet-org/dragnet
[18]https://github.com/goose3/goose3
[19]https://github.com/miso-belica/jusText
[20]https://github.com/codelucas/newspaper
[21]https://github.com/fhamborg/news-please
[22]https://github.com/buriy/python-readability

4.3 Results

The results are listed in Table 1. Baseline extraction is simple and fast, it beats a few systems, showing its interest. JusText is highly configurable and tweaking its configuration leads to better performance than its generic settings, that is why it has been done here. The only solid conclusions which can be drawn for execution times are that goose3 and newspaper are slower than the rest while news-pleases execution time isn't comparable because of operations unrelated to text extraction. The *newspaper* and *boilerpy3* modules do not work without errors on every HTML file in the test set, probably because of malformed HTML, encoding or parsing bugs.

It turns out that rule-based approaches such as Trafilatura's own algorithm ("fast" option) obtain balanced results despite a lack of precision. Although the library in itself is already above the rest, it performs significantly better than the other tested solutions when combined with generic algorithmic approaches.

4.4 External evaluations

A few external evaluations are already available, they ground on early releases of the software during its development. A previous version of Trafilatura is the most efficient open-source library in ScrapingHub's article extraction benchmark.[23] Significantly better results are also reported in the case of French and Swedish for a previous version (Laippala et al., 2020), as well as the best overall macro-mean on the multilingual and manually-annotated DANIEL corpus comprising about 1,600 webpages in five different languages (Lejeune and Barbaresi, 2020). In a further context, the tool has proven to be efficient on main text extraction to create Russian-Turkic parallel corpora (Khusainov et al., 2020).

4.5 Discussion

In some cases, no text is returned, but there is no way to return text at all costs without impacting precision. Trafilatura as a whole is currently made for users aiming for better text quality. While rule-based approaches are both easier to use and to parameterize and could be more efficient in the long-run (Barbaresi and Lejeune, 2020), extraction presets would be useful in order to make the

[23]https://github.com/scrapinghub/article-extraction-benchmark

Python Package	**Precision**	**Recall**	**Accuracy**	**F-Score**	**Diff.**
naive baseline: raw HTML	0.527	0.878	0.547	0.659	0
html2text 2020.1.16	0.488	0.714	0.484	0.580	8.9x
html_text 0.5.2	0.526	**0.958**	0.548	0.679	1.9x
inscriptis 1.1	0.531	**0.958**	0.556	0.683	2.4x
justext 2.2.0 (custom)	0.870	0.584	0.749	0.699	6.1x
newspaper3k 0.2.8	0.921	0.574	0.763	0.708	12.9x
boilerpy3 1.0.2 (article mode)	0.851	0.696	0.788	0.766	4.8x
goose3 3.1.9	**0.950**	0.644	0.806	0.767	18.8x
trafilatura baseline	0.746	0.804	0.766	0.774	**1x**
dragnet 2.0.4	0.906	0.689	0.810	0.783	3.1x
readability-lxml 0.8.1	0.917	0.716	0.826	0.804	5.9x
news-please 1.5.21	0.924	0.718	0.830	0.808	60x
trafilatura 0.8.2 (fast)	0.925	0.868	0.899	0.896	3.9x
trafilatura 0.8.2	0.934	0.890	**0.914**	**0.912**	8.4x

Table 1: Benchmark on 500 documents, 1487 text and 1496 boilerplate segments.

tool more adaptable to research contexts, such as precision-based settings where discarding more elements is paramount or recall-based settings where empty or nearly empty documents are a concern (Gao et al., 2020).

Even if text encoding detection is performed at least as well and possibly better than the competition, a compromise has to be found between speed and accuracy. This issue impedes results to a variable extent, as character sequences are improperly recognized or completely skipped.

5 Conclusions and outlook

The variety of contexts and text genres leads to important design decisions impacting web corpora: could and should the tooling be adapted to particular sources that are targeted or should the extraction be as generic as possible to provide opportunistic ways of gathering information? Due to corpus size or limited resources, the second option is often best. The software package introduced here can help facilitate text data collection and enhance corpus quality. It can answer two research questions related to web corpus construction: How can an accessible generic extraction be run on web pages? And how can text content be found given a list of websites? In the evaluation, Trafilatura performs significantly better than other open-source solutions, which is corroborated by external benchmarks. The article also provided a fast and meaningful baseline which can be used in similar extraction tasks.

Most scraping tools are developed considering particular page styles, whereas linguistic and geographic factors are most probably reflected in HTML structure diversity. In addition, different eras of web development result in diverging "HTMLects". These discrepancies deeply affect extraction processes and can lead to diverging performances. Trafilatura tries to mitigate these biases but cannot bridge all potential gaps. While some large-scale natural language processing and language modeling algorithms can be expected to smooth out irregularities to a certain extent, uses requiring a low margin of error and close reading approaches can greatly benefit from refinements during construction and processing of corpora. As this tool has been released under an open-source license and field-tested by users, feedback loops and collaborative work will hopefully be carried on and foster further improvements.

Although the extraction parameters are configurable, recall- and precision-oriented settings will be made available to make major extraction settings more convenient. Presets corresponding to different usage scenarios could be developed. Comment extraction still has to be evaluated although most libraries do not offer this functionality. Forthcoming additions include refinements of navigation functions, notably further work on a spider in order to be able to derive links from websites which do not provide sitemaps or web feeds.

References

Julian Alarte, Josep Silva, and Salvador Tamarit. 2019. What Web Template Extractor Should I Use?A Benchmarking and Comparison for Five Template Extractors. *ACM Transactions on the Web (TWEB)*, 13(2):1–19.

Adrien Barbaresi. 2015. *Ad hoc and general-purpose corpus construction from web sources*. Ph.D. thesis, École Normale Supérieure de Lyon.

Adrien Barbaresi. 2016. Efficient construction of metadata-enhanced web corpora. In *Proceedings of the 10th Web as Corpus Workshop*, pages 7–16. Association for Computational Linguistics.

Adrien Barbaresi. 2020. htmldate: A Python package to extract publication dates from web pages. *Journal of Open Source Software*, 5(51):2439.

Adrien Barbaresi and Gaël Lejeune. 2020. Out-of-the-Box and Into the Ditch? Multilingual Evaluation of Generic Text Extraction Tools. In *Proceedings of the 12th Web as Corpus Workshop*, pages 5–13.

Adrien Barbaresi and Kay-Michael Würzner. 2014. For a fistful of blogs: Discovery and comparative benchmarking of republishable German content. In *Proceedings of KONVENS 2014, NLP4CMC workshop*, pages 2–10. Hildesheim University Press.

Marco Baroni, Silvia Bernardini, Adriano Ferraresi, and Eros Zanchetta. 2009. The WaCky Wide Web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, 43(3):209–226.

Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. 2008. Cleaneval: a Competition for Cleaning Web Pages. In *Proceedings of the 6th Conference on Language Resources and Evaluation (LREC'08)*, pages 638–643. ELRA.

Daniel Bauer, Judith Degen, Xiaoye Deng, Priska Herger, Jan Gasthaus, Eugenie Giesbrecht, Lina Jansen, Christin Kalina, Thorben Kräger, Robert Märtin, Martin Schmidt, Simon Scholler, Johannes Steger, Egon Stemle, and Stefan Evert. 2007. FIASCO: Filtering the internet by automatic subtree classification. In *Building and Exploring Web Corpora: Proceedings of the 3rd Web as Corpus Workshop (WAC-3)*, pages 111–121.

Vladimír Benko. 2016. wo Years of Aranea: Increasing Counts and Tuning the Pipeline. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*, pages 4245–4248. ELRA.

Chris Biemann, Gerhard Heyer, Uwe Quasthoff, and Matthias Richter. 2007. The Leipzig Corpora Collection: Monolingual Corpora of Standard Size. In *Proceedings of the Corpus Linguistics Conference*.

Marco Brunello. 2009. The creation of free linguistic corpora from the web. In *Proceedings of the Fifth Web as Corpus Workshop (WAC5)*, pages 9–16. Elhuyar Fundazioa.

Christian Buck, Kenneth Heafield, and Bas Van Ooyen. 2014. N-gram Counts and Language Models from the Common Crawl. In *Proceedings of the 9th International Conference on Language Resources and Evaluation (LREC'14)*. ELRA.

Deng Cai, Shipeng Yu, Ji-Rong Wen, and Wei-Ying Ma. 2003. VIPS: a Vision-based Page Segmentation Algorithm. Technical report, Microsoft Technical Report (MSR-TR-2003-79).

Howard J. Carey and Milos Manic. 2016. HTML web content extraction using paragraph tags. In *25th International Symposium on Industrial Electronics (ISIE)*, pages 1099–1105. IEEE.

Isaac Caswell, Theresa Breiner, Daan van Esch, and Ankur Bapna. 2020. Language ID in the wild: Unexpected challenges on the path to a thousand-language web text corpus. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6588–6608. International Committee on Computational Linguistics.

Valter Crescenzi, Giansalvatore Mecca, and Paolo Merialdo. 2001. Roadrunner: Towards Automatic Data Extraction From Large Web Sites. In *Proceedings of the 27th VLDB Conference*, pages 109–118.

Orphée De Clercq and Maribel Montero Perez. 2010. Data Collection and IPR in Multilingual Parallel Corpora. Dutch Parallel Corpus. In *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'10)*, pages 3383–3388. ELRA.

Aidan Finn, Nicholas Kushmerick, and Barry Smyth. 2001. Fact or Fiction: Content Classification for Digital Libraries. In *Joint DELOS-NSF Workshop: Personalization andRecommender Systems in Digital Libraries*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The Pile: An 800GB Dataset of Diverse Text for Language Modeling. *arXiv preprint arXiv:2101.00027*.

Alexander Geyken, Adrien Barbaresi, Jrg Didakowski, Bryan Jurish, Frank Wiegand, and Lothar Lemnitzer. 2017. Die Korpusplattform des "Digitalen Wrterbuchs der deutschen Sprache" (DWDS). *Zeitschrift fr germanistische Linguistik*, 45(2):327–344.

Mohammad Ghasemisharif, Peter Snyder, Andrius Aucinas, and Benjamin Livshits. 2019. SpeedReader: Reader Mode Made Fast and Private. In *Proceedings of the World Wide Web Conference*, pages 526–537.

Natalie Glance, Matthew Hurst, and Takashi Tomokiyo. 2004. Blogpulse: Automated Trend Discovery for Weblogs. In *WWW 2004 Workshop on the Weblogging Ecosystem: Aggregation, Analysis and Dynamics*.

Yan Guo, Huifeng Tang, Linhai Song, Yu Wang, and Guodong Ding. 2010. ECON: an Approach to Extract Content from Web News Page. In *Proceedings of 12th International Asia-Pacific Web Conference (APWEB)*, pages 314–320. IEEE.

Ivan Habernal, Omnia Zayed, and Iryna Gurevych. 2016. C4Corpus: Multilingual Web-size Corpus with Free License. In *Proceedings of the 10th Conference on Language Resources and Evaluation (LREC'16)*, pages 914–922.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. 2017. news-please: A Generic News Crawler and Extractor. In *Proceedings of the 15th International Symposium of Information Science*, pages 218–223.

Hung-Yu Kao, Shian-Hua Lin, Jan-Ming Ho, and Ming-Syan Chen. 2004. Mining web informative structures and contents based on entropy analysis. *IEEE Transactions on Knowledge and Data Engineering*, 16(1):41–55.

Aidar Khusainov, Dzhavdet Suleymanov, Rinat Gilmullin, Alina Minsafina, Lenara Kubedinova, and Nilufar Abdurakhmonova. 2020. First Results of the TurkLang-7 Project: Creating Russian-Turkic Parallel Corpora and MT Systems. In *Proceedings of the Computational Models in Language and Speech Workshop (CMLS 2020)*, pages 90–101. CEUR.

Adam Kilgarriff. 2007. Googleology is bad science. *Computational Linguistics*, 33(1):147–151.

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, WSDM 10*, pages 441–450.

Julia Krasselt, Philipp Dressen, Matthias Fluor, Cerstin Mahlow, Klaus Rothenhäusler, and Maren Runte. 2020. Swiss-AL: A multilingual Swiss Web corpus for applied linguistics. In *Proceedings of the 12th Conference on Language Resources and Evaluation (LREC'20)*, pages 4145–4151. ELRA.

Veronika Laippala, Samuel Rönnqvist, Saara Hellström, Juhani Luotolahti, Liina Repo, Anna Salmela, Valtteri Skantsi, and Sampo Pyysalo. 2020. From Web Crawl to Clean Register-Annotated Corpora. In *Proceedings of the 12th Web as Corpus Workshop*, pages 14–22.

Gaël Lejeune and Adrien Barbaresi. 2020. Bien choisir son outil d'extraction de contenu à partir du Web (Choosing the appropriate tool for Web Content Extraction). In *Actes de la 6e conférence conjointe JEP, TALN, RÉCITAL. Volume 4: Démonstrations et résumés d'articles internationaux*, pages 46–49.

Gaël Lejeune and Lichao Zhu. 2018. A New Proposal for Evaluating Web Page Cleaning Tools. *Computación y Sistemas*, 22(4).

Verena Lyding, Egon Stemle, Claudia Borghetti, Marco Brunello, Sara Castagnoli, Felice Dell'Orletta, Henrik Dittmann, Alessandro Lenci, and Vito Pirrelli. 2014. The PAISÁ Corpus of Italian Web Texts. In *9th Web as Corpus Workshop (WaC-9) @ EACL 2014*, pages 36–43. European chapter of the Association for Computational Linguistics.

Gilad Mishne and Natalie Glance. 2006. Leave a Reply: An Analysis of Weblog Comments. In *Third Annual Workshop on the Weblogging Ecosystem, WWW 2006*.

Tomoyuki Nanno, Toshiaki Fujiki, Yasuhiro Suzuki, and Manabu Okumura. 2004. Automatically Collecting, Monitoring, and Mining Japanese Weblogs. In *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, pages 320–321. ACM.

Christopher Olston and Marc Najork. 2010. Web Crawling. *Foundations and Trends in Information Retrieval*, 4(3):175–246.

Jeff Pasternack and Dan Roth. 2009. Extracting article text from the web with maximum subsequence segmentation. In *Proceedings of the 18th international conference on World Wide Web*, pages 971–980.

Matthew E. Peters and Dan Lecocq. 2013. Content extraction using diverse feature sets. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 89–90.

Jan Pomikálek. 2011. *Removing boilerplate and duplicate content from web corpora*. Ph.D. thesis, Masaryk University.

Pir Abdul Rasool Qureshi and Nasrullah Memon. 2012. Hybrid model of content extraction. *Journal of Computer and System Sciences*, 78(4):1248–1257.

Kevin P. Scannell. 2007. The Crúbadán Project: Corpus building for under-resourced languages. In *Proceedings of the 3rd Web as Corpus Workshop*, volume 4, pages 5–15, Louvain-la-Neuve.

Roland Schäfer. 2016. CommonCOW: massively huge web corpora from CommonCrawl data and a method to distribute them freely under restrictive EU copyright laws. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC'16)*, pages 4500–4504. ELRA.

Roland Schäfer. 2017. Accurate and efficient general-purpose boilerplate detection for crawled web corpora. *Language Resources and Evaluation*, 51(3):873–889.

Roland Schäfer, Adrien Barbaresi, and Felix Bildhauer. 2013. The Good, the Bad, and the Hazy: Design Decisions in Web Corpus Construction. In *Proceedings of the 8th Web as Corpus Workshop*, pages 7–15.

Roland Schäfer and Felix Bildhauer. 2012. Building Large Corpora from the Web Using a New Efficient Tool Chain. In *Proceedings of the 8th Conference on Language Resources and Evaluation (LREC'12)*, pages 486–493. ELRA.

Miroslav Spousta, Michal Marek, and Pavel Pecina. 2008. Victor: the Web-Page Cleaning Tool. In *4th Web as Corpus Workshop (WAC-4)*, pages 12–17.

Fei Sun, Dandan Song, and Lejian Liao. 2011. DOM-based content extraction via text density. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 245–254.

Karane Vieira, Altigran S Da Silva, Nick Pinto, Edleno S De Moura, Joao MB Cavalcanti, and Juliana Freire. 2006. A Fast and Robust Method for Web Page TemplateDetection and Removal. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 258–267.

Thijs Vogels, Octavian-Eugen Ganea, and Carsten Eickhoff. 2018. Web2text: Deep structured boilerplate removal. In *European Conference on Information Retrieval*, pages 167–179. Springer.

Junfeng Wang, Xiaofei He, Can Wang, Jian Pei, Jiajun Bu, Chun Chen, Ziyu Guan, and Gang Lu. 2009. News Article Extraction with Template-Independent Wrapper. In *Proceedings of the WWW 2009*, pages 1085–1086. ACM.

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*, pages 971–980.

Lan Yi, Bing Liu, and Xiaoli Li. 2003. Eliminating noisy information in web pages for data mining. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 296–305.

Cai-Nicolas Ziegler and Michal Skubacz. 2007. Content Extraction from News Pages using Particle Swarm Optimization on Linguistic and Structural Features. In *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence*, pages 242–249. IEEE.