# FUNDUS: A Simple-to-Use News Scraper Optimized for High Quality Extractions

**Max Dallabetta**
max.dallabetta@hu-berlin.de

**Conrad Dobberstein**
conrad.dobberstein@informatik.hu-berlin.de

**Adrian Breiding**
adrian.johannes.breiding@hu-berlin.de

**Alan Akbik**
alan.akbik@hu-berlin.de

Humboldt Universität zu Berlin

## Abstract

This paper introduces FUNDUS, a user-friendly news scraper that enables users to obtain millions of high-quality news articles with just a few lines of code. Unlike existing news scrapers, we use manually crafted, bespoke content extractors that are specifically tailored to the formatting guidelines of each supported online newspaper. This allows us to optimize our scraping for quality such that retrieved news articles are textually complete and without HTML artifacts. Further, our framework combines both crawling (retrieving HTML from the web or large web archives) and content extraction into a single pipeline. By providing a unified interface for a predefined collection of newspapers, we aim to make FUNDUS broadly usable even for non-technical users. This paper gives an overview of the framework, discusses our design choices, and presents a comparative evaluation against other popular news scrapers. Our evaluation shows that FUNDUS yields significantly higher quality extractions (complete and artifact-free news articles) than prior work.

The framework is available on GitHub under https://github.com/flairNLP/fundus and can be simply installed using *pip*.

## 1 Introduction and Motivation

Online news articles are a favored data source for a wide-ranging set of NLP applications including social/political analysis (Hamborg et al., 2019; Masud et al., 2020; Piskorski et al., 2023), market prediction (Ding et al., 2015; Li et al., 2020), and are used as training data for language models (Radford et al., 2019; Gururangan et al., 2022).

In such projects, it is often the first step to compile a corpus of news articles to analyze. This requires (1) identifying the URLs of news articles belonging to a particular set of online newspapers for download, and (2) extracting the article content from the surrounding HTML so that only the full article text remains.

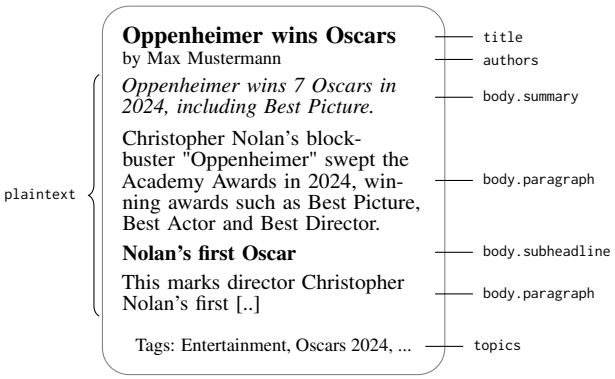


Figure 1: An example article scraped by FUNDUS. Next to the plain text of the article, attributes such as title, authors, paragraphs, subheadlines and topics are directly accessible.

In particular the second task of *content extraction* – also referred to as web scraping or boilerplate removal (Vogels et al., 2018) – is known to be challenging since each online newspaper uses different HTML and text formatting guidelines. This makes it non-trivial to distinguish between article content and other elements such as adverts, unrelated asides, captions, etc. To address these issues, several libraries have been developed that streamline the crawling and content extraction of online newspapers (Hamborg et al., 2017; Leonhardt et al., 2020; Barbaresi, 2021).

**Limitations.** However, existing libraries rely on generic methods for content extraction, based either on heuristics or trained machine learning models. This allows them to be applied across an arbitrary number of online newspapers, but comes at a cost of extraction accuracy: the quality of the news article texts varies depending on how well the heuristics or learned rules capture the HTML formatting of a particular newspaper.

For instance, the evaluation presented in this paper shows that existing frameworks encounter difficulties with at least one newspaper, resulting in F1-scores below 60% for all articles retrieved



*Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 305–314

| Library | [Info] | Language | Approach | Extractor | Crawling | F1 |
|---|---|---|---|---|---|---|
| FUNDUS | (ours) | Python | Heuristics-based: Rules | bespoke | **yes** | **97.69** |
| Trafilatura | (Barbaresi, 2021) | Python | Heuristics-based: Rules | generic | **yes** | 89.81 |
| BoilerNet | (Leonhardt et al., 2020) | Python | ML-based: Sequence labeling | generic | no | 85.77 |
| news-please | (Hamborg et al., 2017) | Python | Heuristics-based: Meta-rules | generic | **yes** | 85.81 |
| jusText | (Pomikálek, 2011) | Python | Heuristics-based: Rules | generic | no | 86.96 |
| Boilerpipe | (Kohlschütter et al., 2010) | Java | ML-based: Node classification | generic | no | 79.90 |
| BTE | (Finn et al., 2001) | Python | Heuristics-based: Tag distributions | generic | no | 87.14 |

Table 1: Comparison of FUNDUS to other prominent scraping libraries, some of which include crawling functionality. The F1-score measures the extraction quality on our benchmark, as detailed in Section 4.

from this source. This means that, due to their generic nature, existing libraries provide no guarantee and no means to ensure that scraped articles are textually complete and without artifacts.

Put more plainly, it may be argued that existing libraries prioritize *quantity* (i.e. scaling across many newspapers) over *quality* (i.e. high-quality extraction of complete article texts and meta-attributes). This may cause problems in use cases in which the overall quality of a news corpus is more important than its quantity (Li et al., 2023; Marion et al., 2023).

**Contributions.** With this paper, we present FUNDUS, a news crawling library in which we pursue an orthogonal approach to prior work. Rather than aiming for a set of general rules applicable to all newspapers, our library uses separate, manually created HTML content extractors – referred to as *parsers* within the library – for each online newspaper. This allows us to match extraction methodologies specifically to a newspaper and thus manually optimize the accuracy of text extraction.

Further, as Figure 1 illustrates, this enables us to write more complex content extractors compared to prior work to preserve a news article's structure (distinguishing between paragraphs, subheadlines, and the article summary), and extract meta-attributes such as topics. In more detail, our contributions are:

1. We present the FUNDUS library, illustrate its ease of use, and discuss the merits and drawbacks of pursuing an approach of manually crafted, bespoke extractors for selected online newspapers.

2. We illustrate how FUNDUS can be used not only for news articles that are currently available online, but also scrape the extensive COMMONCRAWL web archive CC-NEWS. This allows users to create very large, high quality news corpora with only a few lines of code.

3. We comparatively evaluate FUNDUS against well-known crawling/content extraction frameworks using a newly created dataset of paragraph-wise annotated HTML files, and provide statistics on its data potential leveraging CC-NEWS.

We find that FUNDUS outperforms all other libraries in terms of yielding complete and artifact-free text (see Table 1), thus indicating its usefulness for projects in which textual quality is a priority. To enable the NLP community to use FUNDUS in their projects – and add parsers for new newspapers – we open source FUNDUS under an MIT licence[1].

## 2 Related Work

Table 1 gives an overview of popular scraping libraries and a comparison to FUNDUS.

### 2.1 Content Extraction

Existing approaches for content extraction are based either on heuristics or machine learning:

**Heuristics-based extraction.** Early heuristic methods used the assumption that fewer HTML tags are used in main text content than in other elements. For instance, BTE (Finn et al., 2001) employs a cumulative tag distribution to identify the region with the lowest tag-to-text ratio as the main content. JUSTEXT (Pomikálek, 2011) segments HTML into content blocks based on selected tag types. Thereafter, these blocks are evaluated as content and distinguished from boilerplate using metrics such as link density, block length and block complexity.

Later approaches instead focus on the underlying DOM tree using a series of XPath expressions to determine regions of the tree as main content. For instance, TRAFILATURA (Barbaresi, 2021) uses a cascade of XPath expressions to initially sanitize HTML content by removing unwanted sections and subsequently querying for relevant content. NEWS-

[1]Available at: https://github.com/flairNLP/fundus

Listing 1: Crawl US-based publishers

```
# crawl US-based publishers
crawler = Crawler(PublisherCollection.us)

# crawl 10 news articles
articles = crawler.crawl(max_articles=10)

# print them out one-by-one
for article in articles:
    print(article)
```

Listing 2: Crawl one German publisher

```
# crawl one specific German publisher
crawler = Crawler(PublisherCollection.de.DW)

# crawl 10 news articles
articles = crawler.crawl(max_articles=10)

# print them out one-by-one
for article in articles:
    print(article)
```

Figure 2: Two example usages of FUNDUS to crawl articles from (1) all supported US-based publishers, and (2) only one specific German publisher ("Deutsche Welle").

PLEASE (Hamborg et al., 2017) facilitates a combination of state-of-the-art extractors.

**ML-based extraction.** The second family of approaches formulates content extraction as a classification problem. For instance, BOILERPIPE (Kohlschütter et al., 2010) uses decision trees to classify text blocks (uninterrupted text devoid of tags) as content or boilerplate. BOILERNET (Leonhardt et al., 2020) tokenizes web pages and trains a bidirectional LSTM to classify each segment.

**Content extraction in FUNDUS.** Unlike prior work, we use bespoke extractors for each newspaper, thus allowing us to manually optimize for accuracy and attribute coverage. Although our approach inherently prioritizes quality, it also incurs a trade-off in terms of quantity, as it necessitates humans to create a separate extraction logic for each online newspaper. To manage this, we pursue a community-based approach and provide simple abstractions (and tutorials) to enable open source contributors to add support for new newspapers.

### 2.2 Crawling

Next to content extraction, identifying and downloading pages at scale can also be challenging. Such a system, which we refer to as a *crawler*, should be "polite" (crawling only permissive online newspapers and keeping server workload low) and able to filter for pages relevant to the use case. However, the majority of existing libraries (see Table 1) focus solely on content extraction, thus requiring users to resort to separate tools.

**Crawling in FUNDUS.** We combine both crawling and content extraction in a single library. Unlike prior work which requires complex external configuration or comprehension of content maps like RSS feeds and sitemaps, FUNDUS provides predefined settings for each supported newspaper. In FUNDUS, users are only required to select a list of newspapers to include and issue a single method call, without additional configuration. This directly yields already extracted text. By hiding the underlying complexity, we aim to make FUNDUS broadly usable even for non-technical users.

## 3 FUNDUS

We introduce FUNDUS with a usage example, discuss our article and publisher-based logic, and illustrate how we distinguish between *forward* and *backward* crawling.

### 3.1 Usage Example

We provide two example snippets on how to use FUNDUS to scrape news articles in Figure 2:

**Listing 1: Crawl all US-based publishers.** This example demonstrates the process of scraping news articles from a selection of US-based publishers. First, we instantiate the `Crawler` object by passing `PublisherCollection.us` to it. This indicates that all US-based publishers currently supported in FUNDUS should be used as data sources. We then instruct the crawler to gather articles until a threshold of 10 articles is met (by passing `max_articles=10`). This returns a generator[2] of `Article` objects, encapsulating the plain text of each news article alongside structured information such as the title, the author, and the date of crawling. Finally, we iterate over the generator, printing each article successively for human inspection.

**Listing 2: Crawl one specific source.** In the second example, we focus on articles from a particular publisher. We choose the German publisher DW ("Deutsche Welle") for this example. The code structure mirrors that of Listing 1, except that we instantiate the `Crawler` by passing `PublisherCollection.de.DW`. This narrows the search to a single publisher.

### 3.2 Articles

FUNDUS' metadata and content extractions can be accessed through a single dataclass called `Article`.

[2] FUNDUS uses generators to prioritize responsiveness by delivering articles as they become available, rather than accumulating them for subsequent retrieval.

| Attribute | Description |
|---|---|
| title | Title of the news article |
| body | All article text with pararaph structure |
| authors | Creators of the article |
| publish_date | Article release date |
| topics | Publisher-assigned topics |
| free_access | Boolean indicating free accessibility |
| ld | Parsed JSON+LD metadata |
| meta | Parsed HTML metadata |
| plaintext | Concatenated article body |
| lang | Auto-detected article language |
| html | HTML content and meta information |
| exception | Indicates whether an exception occurred during extraction |

Table 2: Directly accessible attributes for each scraped `Article` (more details found in the Appendix, Table 6).

As indicated in the examples, users can obtain a quick overview of an article by simply printing it. This will output the article's title, a snippet of the extracted text content, the URL and publisher from which it was crawled, along with a timestamp indicating the publication time.

All attributes for an `Article` are listed in Table 2. They are directly accessible using Python's dot notation. Attributes for each `Article` include its *title*, textual *body*, *authors*, *publishing_date*, *topics*, etc. The body attribute in particular captures the entire article structure including a summary, paragraphs and subheadlines, as depicted in Figure 1.

### 3.3 Publishers and Collections

As the usage examples illustrate, users may specify which (set of) publishers to target when crawling for news articles. For FUNDUS, a publisher refers to an individual online newspaper, such as "Deutsche Welle". FUNDUS assumes that each online newspaper adheres to its own HTML and formatting guidelines. This means that for each publisher, we specify (1) where to find the URLs of each news article, and (2) how to extract the main textual content from downloaded HTML pages. This specification is created once (e.g. by a contributor to the FUNDUS repository) by creating a `Publisher`-specific enum object for the newly supported online newspaper.

Users can then pass this object to our `Crawler` to target this newspaper. To enhance accessibility and provide locality, we group publishers by their countries of origin within the `PublisherCollection`. This allows users to crawl all supported publishers of a specific country using their two-letter ISO 3166-2 language code. We illustrate this by crawling all US-based publishers in Listing 1. As of writing, the framework supports **39 publishers** spanning 5 different regions.

### 3.4 Forward and Backward Crawling

Internally, each `Publisher` defines one or multiple HTML sources, determining how a crawler locates the URLs of its news articles. Here, we distinguish between *forward* and *backward* crawling.

**Forward crawling.** With forward crawling, we refer to accessing news articles that are currently available online on the news sites of supported newspapers. To identify URLs, we support the use of content maps like RSS feeds and sitemaps provided by the individual publisher. Sitemaps are typically exposed to crawlers via a `"robots.txt"` file, which also outlines user-agent-specific restrictions on subdomains and crawl intervals.

**Backward crawling.** With backward crawling, we refer to accessing news articles in a static web dump. Specifically, we support the **CC-NEWS**[3] dataset provided by the COMMONCRAWL initiative. At the time of writing, this dataset comprises around 40 terabytes of WARC-formatted data, containing millions of news articles dating back to 2016. To handle such a large volume of data efficiently, FUNDUS offers the option to narrow crawls by a date range. Additionally, we stream WARC files and utilize the *FastWARC* library (Bevendorff et al., 2021) for in-memory processing to mitigate storage requirements.

### 3.5 Content Extraction

The central component of FUNDUS' content extraction is the `Parser` class. It is individually implemented for every publisher and combines both generic and newspaper-specific extraction methods. The generic heuristics target structured information such as paywall restrictions, language detection, and meta-information (HTML tags and JSON+LD) and can be manually overwritten for specific publishers if necessary.

They are complemented with hand-tailored rules to extract the core parts of a news article such as the title, the textual body, and the authors. These rules are formulated as simple selectors (CSSSelect/XPath expressions) or metadata keys, and can typically be easily determined by inspecting the DOM tree of a few HTML examples.

[3] https://commoncrawl.org/blog/news-dataset-available

| Scraper | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| FUNDUS | 100 | 100 | 100 | 100 | 99 | 100 | 89 | 92 | 100 | 99 | 99 | 100 | 100 | 87 | 100 | 98 |
| BTE | 87 | 97 | 83 | 99 | 95 | 91 | 78 | 68 | 97 | **50** | 84 | 85 | 99 | 90 | 96 | 96 |
| jusText | 79 | 94 | 85 | 96 | 95 | 89 | 58 | 89 | 97 | **52** | 95 | 97 | 99 | 74 | 95 | 97 |
| Trafilatura | 93 | 99 | **42** | 99 | 97 | 94 | 84 | 97 | 100 | 74 | 100 | 95 | 100 | 67 | 97 | 100 |
| news-please | 100 | 91 | 93 | 100 | 81 | 95 | 78 | 97 | 97 | 82 | 85 | 85 | 71 | **32** | 100 | 85 |
| Boilerpipe | 82 | 96 | **5** | 97 | 75 | 93 | 75 | 96 | 96 | **52** | 75 | 95 | 88 | 77 | 91 | 87 |
| BoilerNet | **51** | 77 | 88 | 95 | 94 | 90 | 65 | 92 | 97 | 70 | 84 | 93 | 99 | 90 | 90 | 96 |

Table 3: Rounded mean F1 scores of compared scrapers per publisher with scores below 60 highlighted. Publishers are: **A**: AP News; **B**: CNBC; **C**: Fox News; **D**: The Washington Free Beacon; **E**: The LA Times; **F**: Occupy Democrats; **G**: Reuters; **H**: The Gateway Pundit; **I**: The Guardian; **J**: The Independent; **K**: The Intercept; **L**: The Nation; **M**: The New Yorker; **N**: The Telegraph; **O**: The Washington Times; **P**: iNews

Extraction rules are encapsulated as class methods for each parser and "registered" as attributes using a decorator. Each attribute in a parsed article can be directly accessed (c.f. Section 3.2).

## 4 Evaluation

We comparatively evaluate FUNDUS against prominent scraping libraries. Our goals are to (1) determine the quality of our bespoke content extraction approach compared to the generic approaches of prior work, and to (2) better understand the data potential of FUNDUS, i.e. to estimate the size of news corpora that FUNDUS can create.

### 4.1 Experimental Setup

#### 4.1.1 Evaluation Dataset

To evaluate content extraction, we require a dataset of raw HTML pages and corresponding gold annotations of the journalistic content found on each page. This allows us to test whether content extraction libraries are capable of correctly distinguishing the article's text content from surrounding elements. Further, the dataset should cover the publishers in FUNDUS. As our survey of related work found no suitable datasets, we manually created our own[4].

**Data selection and annotation.** We select the 16 English-language publishers FUNDUS currently supports as the data source, and retrieve five articles for each publisher from the respective RSS feeds/sitemaps. We stress that the evaluation corpus consists only of articles that were published after the respective FUNDUS extractors were finalized. There is therefore no data contamination in our evaluation dataset.

The selection process yielded an evaluation corpus of 80 news articles. From it, we manually extracted the plain text from each article and stored it together with information on the original paragraph structure. Annotation was separately performed by two authors of this publication. Our annotation guidelines can be found in Appendix C and include the option to mark individual paragraphs as "optional". To check for consistency between the two annotators, the first article of every publisher was annotated by both. Of 16 doubly annotated articles, 3 disagreements were discussed and resolved.

[4]The dataset, scores, and evaluation metrics can be found at: https://github.com/dobbersc/fundus-evaluation

| Scraper | Precision | Recall | F1-Score |
|---|---|---|---|
| FUNDUS | **99.89**±0.57 | 96.75±12.75 | **97.69**±9.75 |
| Trafilatura | 90.54±18.86 | 93.23±23.81 | 89.81±23.69 |
| BTE | 81.09±19.41 | **98.23**±8.61 | 87.14±15.48 |
| jusText | 86.51±18.92 | 90.23±20.61 | 86.96±19.76 |
| news-please | 92.26±12.40 | 86.38±27.59 | 85.81±23.29 |
| BoilerNet | 84.73±20.82 | 90.66±21.05 | 85.77±20.28 |
| Boilerpipe | 82.89±20.65 | 82.11±29.99 | 79.90±25.86 |

Table 4: Overall performance of FUNDUS and compared scrapers in terms of averaged ROUGE-LSum precision, recall and F1-score and their standard deviation. The table is sorted in descending order over the F1-score.

#### 4.1.2 Evaluation Metric

We follow prior work by Bevendorff et al. (2023) and use the ROUGE-LSum (Lin, 2004) score, which is commonly used for evaluating the similarity between two text sequences, particularly in tasks such as machine translation. Here, we compare the extracted article text to the gold text.

For each article in the dataset, we calculate the precision, recall and F1-score using the ROUGE-LSum metric. This computation is performed with every possible combination of optional paragraphs removed from the ground truth, selecting the best F1 score from all options. To determine the final score, we aggregate the scores of individual articles by computing the mean and the standard deviation.

| Year | B | C | E | G | H | I | J | L | M | N | O | P | Total |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2023 total | 19,628 | 75,363 | 40,048 | 63,664 | 12,951 | 55,899 | 176,913 | 2,380 | 2,973 | 57,600 | 15,388 | 28,911 | 551,718 |
| 2023 body | 14,048 | 72,660 | 28,259 | 63,403 | 12,672 | 50,961 | 166,070 | 2,125 | 2,528 | 57,441 | 15,374 | 28,911 | 514,452 |
| 2022 total | 21,820 | 209,452 | 40,531 | 115,811 | 0 | 96,504 | 217,238 | 2,296 | 4,904 | 61,053 | 19,947 | 30,285 | 819,841 |
| 2022 body | 16,903 | 67,700 | 0 | 115,642 | 0 | 87,176 | 202,829 | 2,293 | 4,126 | 60,042 | 19,928 | 30,283 | 606,922 |
| 2021 total | 26,741 | 101,906 | 47,019 | 248,619 | 40 | 93,954 | 112,498 | 2,345 | 4,652 | 73,953 | 45,184 | 20,791 | 777,702 |
| 2021 body | 26,388 | 101,316 | 0 | 81,364 | 40 | 80,046 | 104,392 | 2,345 | 1,009 | 71,768 | 45,116 | 20,791 | 534,575 |
| 2020 total | 31,725 | 109,155 | 54,901 | 399,925 | 33 | 97,174 | 0 | 2,839 | 5,318 | 89,393 | 90,065 | 71,070 | 951,598 |
| 2020 body | 31,018 | 108,185 | 0 | 0 | 32 | 4,449 | 0 | 2,838 | 0 | 84,152 | 90,046 | 70,919 | 391,639 |

Table 5: Total number of articles extracted from **CC-NEWS** in the timeframe 01/01/2020 – 01/01/2024, including a breakdown by online newspaper. Publisher identities correspond to those delineated in Table 3.

### 4.2 Results and Discussion

Table 4 summarizes our findings. We make the following observations in regard to FUNDUS:

**Highest overall F1-score.** We first note that our approach yields the highest quality extractions as measured by the ROUGE-LSum F1-score. This confirms our hypothesis that bespoke content extractors are naturally well-suited for high-quality text extraction. Further, this validates our assumption that publishers follow internally consistent formatting guidelines across all news articles.

**Lower standard deviation.** We also note that FUNDUS has a lower variability of extraction quality – as measured by the standard deviation – than other approaches. This indicates that our extractors are more consistent than generic approaches based on heuristics or on ML models. We visualize this property in Table 3 in the Appendix.

**Existing libraries struggle with at least one newspaper (Table 3).** To get a better insight into the extraction capability of each compared library, we compute the F1-scores on a per-publisher basis. As Table 3 shows, we find that the F1-score widely varies from publisher to publisher for generic approaches, whereas FUNDUS yields consistent quality extractions.

**Errors remain.** However, we also note that despite manually-crafted, bespoke rules, our extraction is not perfect. Upon manual inspection, we find that a small portion of articles of a publisher deviates from standard formatting, for instance to emphasize quotations or include nested paragraphs. This particularly affected *live news tickers* which some newspapers feature for selected events.

### 4.3 Scalability

Since FUNDUS is limited to a set of supported newspapers, a natural question is how much data one can expect to crawl using FUNDUS.

**Data potential (Table 5).** To investigate, we extract news articles from the CC-NEWS web archive spanning the years 2020 to 2024. We find that 12 of our 16 English-language publishers are included in the archive. Further, despite crafting extraction rules targeting articles from 2023 onward, we note robust backward compatibility, with a significant decrease only noticeable in 2020 (e.g. fewer URLs that yield text bodies). In total, we extract over 2 million articles with bodies.

**Performance.** We evaluated the crawling performance of FUNDUS using a machine equipped with 2 Xeon 6254 CPUs, 756 GB of RAM, and a bandwidth of 10 Gbit/s. For CC-NEWS, we estimate the performance by focusing on the year 2023, as it constitutes the largest data dump among the four years evaluated. It comprises 201,586,338 unique URLs sourced from 34,229 different domains, resulting in approximately 8.2 terabytes of gzip-compressed WARC data. FUNDUS took 2.1 hours to yield the results presented in Table 5.

In terms of forward crawling, we scraped 10,000 articles across all 39 supported publishers. Employing a delay of 1 second for subsequent calls on the same publisher, the process took 549 seconds.

## 5 Conclusion and Outlook

We presented FUNDUS, an easy-to-use news scraper built on the idea of bespoke content extractors for supported online newspapers. Our evaluation shows that our approach successfully optimizes for quality, indicating that FUNDUS is a viable option for use cases in which data quality is a priority. Further, we combine both crawling and scraping functionality in a single pipeline, and support access to the static web archive CC-NEWS.

With FUNDUS' open-source approach, we invite the community to contribute support for additional online newspapers. To assist in this process, we plan to investigate semi-automatic methods to suggest extraction rules in future work.

## Limitations

The main limitation of our approach is its inherent lack of scalability across many online newspapers, since manual rules need to be written for each supported newspaper. As we argue in our paper, the benefits of extraction quality of our manual approach may outweigh quantity considerations depending on whether quality is a priority in an NLP use case. Additionally, though our approach does not easily scale across newspapers, it does scale across large web archives, meaning that we can retrieve large news corpora even with a limited number of supported publishers. Further, we aim to make it easy for the open source community to add support for new newspapers.

A related limitation is that regular maintenance of extractions is necessary, since online newspapers might change their formatting guidelines over time. To monitor this, we automatically check whether FUNDUS is able to extract text content from currently online articles on a periodic basis. This flags whenever formatting guidelines have changed.

## Ethics Statement

Newspapers play a pivotal role in modern society, often referred to as the fourth estate or fourth power. Maintaining independence necessitates self-financing for news media, thus evoking an inherent need for good-quality content to be adequately paid. However, the advent of Large Language Models (LLMs) revealed that web corpora, particularly news corpora, are often used for commercial benefit in a non-consensual manner. In response, our approach prioritizes the ethical acquisition of news articles by providing a simple option to crawl only those unrestricted by paywalls.

Moreover, we advocate for the non-commercial use of FUNDUS, aligning with our ethos of respecting intellectual property rights and promoting fair compensation for content creators. By fostering a culture of respect for intellectual property and fair compensation for content creators, we can help ensure the continued production of high-quality news and information for the benefit of society as a whole.

Another source of ethical concern stems from inherent biases in datasets obtained from the web (Bender et al., 2021), as prior work has shown that language models trained over biased data tend to reflect these biases (Haller et al., 2023). With FUNDUS, users can specifically select which newspaper to include when creating a news corpus, thus giving some degree of control (for instance, over political biases) during corpus creation.

Lastly, also the datasets themselves are worthy of a discussion, as FUNDUS provides easy access to the CC-News dataset. The Common Crawl Foundation has measures in place to respect the resources and work of content creators and comply with the US Fair Use doctrine, which provides a legal basis. Baack (2024) and Xue (2024) also illustrate limitations and biases of the Common Crawl dataset, which should be taken into account as well as acknowledging that not necessarily every rights-holder actively approved of their data being crawled and used.

## Acknowledgements

Max Dallabetta, Conrad Dobberstein and Alan Akbik are supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Emmy Noether grant “Eidetic Representations of Natural Language” (project number 448414230). Alan Akbik is further supported under Germany’s Excellence Strategy "Science of Intelligence" (EXC 2002/1, project number 390523135).

## References

Stefan Baack. 2024. A critical analysis of the largest source for generative ai training data: Common crawl. In *Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency*, FAccT ’24, page 2199–2208, New York, NY, USA. Association for Computing Machinery.

Adrien Barbaresi. 2021. Trafilatura: A web scraping library and command-line tool for text discovery and extraction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 122–131, Online. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

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, pages 2594–2603. Association for Computing Machinery.

Janek Bevendorff, Martin Potthast, and Benno Stein. 2021. FastWARC: Optimizing Large-Scale Web Archive Analytics. In *3rd International Symposium on Open Search Technology (OSSYM 2021)*. International Open Search Symposium.

Xiao Ding, Yue Zhang, Ting Liu, and Junwen Duan. 2015. Deep learning for event-driven stock prediction. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, page 2327–2333. AAAI Press.

Aidan Finn, Nicholas Kushmerick, and Barry Smyth. 2001. Fact or fiction: Content classification for digital libraries. In *DELOS Workshops / Conferences*.

Suchin Gururangan, Dallas Card, Sarah Dreier, Emily Gade, Leroy Wang, Zeyu Wang, Luke Zettlemoyer, and Noah A. Smith. 2022. Whose language counts as high quality? measuring language ideologies in text data selection. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2562–2580, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Patrick Haller, Ansar Aynetdinov, and Alan Akbik. 2023. Opiniongpt: Modelling explicit biases in instruction-tuned llms.

Felix Hamborg, Karsten Donnay, and Bela Gipp. 2019. Automated identification of media bias in news articles : an interdisciplinary literature review. *International Journal on Digital Libraries*, 20(4):391–415.

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*, number 70 in Schriften zur Informationswissenschaft, pages 218–223, Glückstadt. Verlag Werner Hülsbusch.

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, page 441–450, New York, NY, USA. Association for Computing Machinery.

Jurek Leonhardt, Avishek Anand, and Megha Khosla. 2020. Boilerplate removal using a neural sequence labeling model. In *Companion Proceedings of the Web Conference 2020*, WWW '20, page 226–229, New York, NY, USA. Association for Computing Machinery.

Xiaodong Li, Pangjing Wu, and Wenpeng Wang. 2020. Incorporating stock prices and news sentiments for stock market prediction: A case of hong kong. *Information Processing & Management*, 57(5):102212.

Yuanzhi Li, Sébastien Bubeck, Ronen Eldan, Allie Del Giorno, Suriya Gunasekar, and Yin Tat Lee. 2023. Textbooks are all you need ii: phi-1.5 technical report.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. 2023. When less is more: Investigating data pruning for pretraining llms at scale.

Sarah Masud, Subhabrata Dutta, Sakshi Makkar, Chhavi Jain, Vikram Goyal, Amitava Das, and Tanmoy Chakraborty. 2020. Hate is the new infodemic: A topic-aware modeling of hate speech diffusion on twitter.

Jakub Piskorski, Nicolas Stefanovitch, Giovanni Da San Martino, and Preslav Nakov. 2023. SemEval-2023 task 3: Detecting the category, the framing, and the persuasion techniques in online news in a multilingual setup. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 2343–2361, Toronto, Canada. Association for Computational Linguistics.

Jan Pomikálek. 2011. *Removing Boilerplate and Duplicate Content from Web Corpora [online]*. Doctoral theses, dissertations, Masaryk University, Faculty of InformaticsBrno. SUPERVISOR: prof. PhDr. Karel Pala, CSc.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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 40*, pages 167–179. Springer.

Alex Xue. 2024. Balancing discovery and privacy: A look into opt–out protocols. Common Crawl Blog.

## A Live Demo

You can find a short live demonstration of our library on YouTube following this link: https://youtu.be/9GJExMelhdI

## B Article Attributes

Table 6 provides a comprehensive overview of all attributes of the FUNDUS Article class alongside additional information concerning the content extraction process, the methodology employed, and the Python data type utilized to represent each attribute internally.

## C Annotation Guidelines

For any given article we expect to extract the main textual content providing information on the article's topic which should align with editorial standards and be relevant to the headline. Additionally, relevant meta-information, e.g. declaration of third parties involved, additional information related, but not part of the main content, can also be extracted. Explicitly excluded are:

- The headline
- Captions of figures, images, and other objects
- Tables, due to the lack of a normalized representation

All extracted paragraphs are to be considered non-optional, unless one or more of the following conditions are fulfilled:

- The paragraph's sole purpose is formatting
- The paragraph is or is part of a summary of the article's contents
- The paragraph solely consists of meta-information (e.g. mentioning a contributing third party)
- The paragraph is not directly semantically related to the articles' content

## D Standard Deviation of Compared Libraries

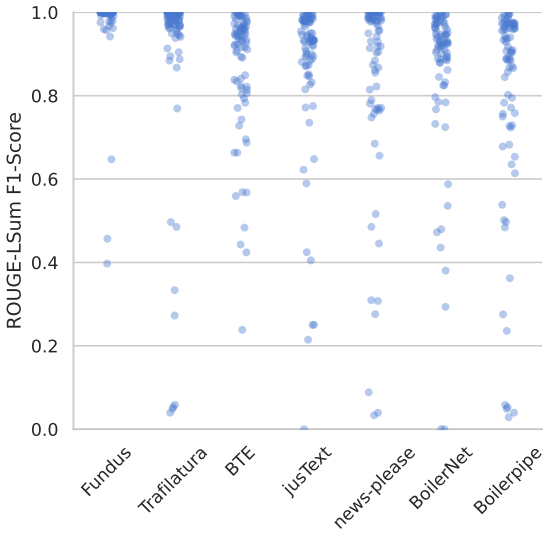


Figure 3: Distribution of ROUGE-LSum F1-scores of scraper extractions. The scrapers are sorted in descending order over the F1-score.

## E CC-NEWS Crawling

In addition to the examples outlined in Section 3.1, we aim to illustrate the ease of transitioning from forward to backward crawling. As depicted in Figure 4, this transition can be effortlessly achieved by substituting the employed crawler. Moreover, we offer a unified extraction interface, ensuring that switching between crawlers does not mandate parameter adjustments.

Listing 3: Crawl US-based publishers

```
crawler = Crawler(PublisherCollection.us)

# To retrieve articles from CC-NEWS instead
    one must simply exchange the pipeline
crawler =
    CCNewsCrawler(PublisherCollection.us)

for article in
    crawler.crawl(max_articles=100):
    print(article)
```

Figure 4: An example usage of Fundus to crawl articles from CC-NEWS

| Attribute | Description | Extraction | Methodology | Python type |
|---|---|---|---|---|
| **title** | Title of the news article | rule-based | metadata | str |
| **body** | Object that allows direct access to paragraphs | rule-based | selectors | custom |
| **authors** | Creators of the article | rule-based | mixed | list |
| **publishing_date** | Release date provided by the publisher | rule-based | mixed | datetime |
| **topics** | Publisher-assigned topics | rule-based | mixed | list |
| **free_access** | Boolean indicating free accessibility | mixed | mixed | bool |
| ld | JSON+LD data as extracted from the article | generic | selectors | custom |
| meta | HTML meta tags as parsed from the article | generic | selectors | dict |
| plaintext | Concatenated, stripped, and cleaned article body | - | - | str |
| lang | Auto-detected article language | - | - | str |
| html | contains raw HTML, origin URL, crawl date, and crawl source | - | - | custom |
| exception | Exception indicating if an exception occurred during extraction | - | - | Exception |

Table 6: Article attributes alongside their description, extraction method, the applied methodology, and used Python type.