The Journal of Open Source Software

htmldate: A Python package to extract publication dates from web pages

Adrien Barbaresi[1]

1 Berlin-Brandenburg Academy of Sciences

DOI: 10.21105/joss.02439

Software

- Review
- Repository
- Archive

Editor: Daniel S. Katz

Reviewers:

- @geoffbacon
- @proycon

Submitted: 17 June 2020
Published: 30 July 2020

Introduction

Rationale

Metadata extraction is part of data mining and knowledge extraction. Being able to better qualify content allows for insights based on descriptive or typological information (e.g., content type, authors, categories), better bandwidth control (e.g., by knowing when webpages have been updated), or optimization of indexing (e.g., caches, language-based heuristics). It is useful for applications including database management, business intelligence, or data visualization. This particular effort is part of a methodological approach to derive information from web documents in order to build text databases for research, chiefly linguistics and natural language processing. Dates are critical components since they are relevant both from a philological standpoint and in the context of information technology.

Although text is ubiquitous on the Web, extracting information from web pages can prove to be difficult. Web documents come in different shapes and sizes mostly because of the wide variety of genres, platforms, and content management systems, and not least because of greatly diverse publication goals. In most cases, immediately accessible data on retrieved webpages do not carry substantial or accurate information: neither the URL nor the server response provide a reliable way to date a web document, that is to find out when it has been published or possibly modified. In that case it is necessary to fully parse the document or apply robust scraping patterns on it. Improving extraction methods for web collections can hopefully allow for combining both the quantity resulting from broad web crawling and the quality obtained by accurately extracting text and metadata and by rejecting documents which do not match certain criteria.

Research context

Fellow colleagues are working on a lexicographic information platform (Geyken et al., 2017) at the language center of the Berlin-Brandenburg Academy of Sciences (dwds.de). The platform hosts and provides access to a series of metadata-enhanced web corpora (Barbaresi, 2016). Information on publication and modification dates is crucial to be able to make sense of linguistic data, that is, in the case of lexicography to determine precisely when a given word was used for the first time and how its use evolves through time.

Large "offline" web text collections are now standard among the research community in linguistics and natural language processing. The construction of such text corpora notably involves "crawling, downloading, 'cleaning' and de-duplicating the data, then linguistically annotating it and loading it into a corpus query tool" (Kilgarriff, 2007). Web crawling (Olston & Najork, 2010) involves a significant number of design decisions and turning points in data processing, without which data and applications turn into a "Wild West" (Jo & Gebru, 2020). Researchers

face a lack of information regarding the content, whose adequacy, focus, and quality are the object of a post hoc evaluation (Baroni, Bernardini, Ferraresi, & Zanchetta, 2009). Comparably, web corpora (i.e., document collections) usually lack metadata gathered with or obtained from documents. Between opportunistic and restrained data collection (Barbaresi, 2015), a significant challenge lies in the ability to extract and pre-process web data to meet scientific expectations with respect to corpus quality.

Functionality

`htmldate` finds original and updated publication dates of web pages using heuristics on HTML code and linguistic patterns. It operates both within Python and from the command-line. URLs, HTML files, or HTML trees are given as input, and the library outputs a date string in the desired format or `None` as the output is thouroughly verified in terms of plausibility and adequateness.

The package features a combination of tree traversal and text-based extraction, and the following methods are used to date HTML documents:

1. Markup in header: common patterns are used to identify relevant elements (e.g., `link` and `meta` elements) including Open Graph protocol attributes and a large number of content management systems idiosyncrasies
2. HTML code: The whole document is then searched for structural markers: `abbr` and `time` elements as well as a series of attributes (e.g. `postmetadata`)
3. Bare HTML content: A series of heuristics is run on text and markup:
    - in `fast` mode the HTML page is cleaned and precise patterns are targeted
    - in `extensive` mode all potential dates are collected and a disambiguation algorithm determines the best one

Finally, a date is returned if a valid cue could be found in the document, corresponding to either the last update or the original publishing statement (the default), which allows for switching between original and updated dates. The output string defaults to ISO 8601 YMD format.

`htmldate` is compatible with all recent versions of Python (currently 3.4 to 3.9). It is designed to be computationally efficient and used in production on millions of documents. All the steps needed from web page download to HTML parsing, scraping, and text analysis are handled, including batch processing. It is distributed under the GNU General Public License v3.0. Markup-based extraction is multilingual by nature, and text-based refinements for better coverage currently support German, English and Turkish.

State of the art

Diverse extraction and scraping techniques are routinely used on web document collections by companies and research institutions alike. Content extraction mostly draws on Document Object Model (DOM) examination, that is, on considering a given HTML document as a tree structure whose nodes represent parts of the document to be operated on. Less thorough and not necessarily faster alternatives use superficial search patterns such as regular expressions in order to capture desirable excerpts.

Alternatives

There are comparable software solutions in Python. The following date extraction packages are open-source and work out-of-the-box:

- `articleDateExtractor` detects, extracts, and normalizes the publication date of an online article or blog post (Geva, 2018),
- `date_guesser` extracts publication dates from a web pages along with an accuracy measure which is not tested here (Carroll & Valiukas, 2019),
- `goose3` can extract information for embedded content (Grangier, Barrus, & Sidorov, 2019),
- `htmldate` is the software package described here; it is designed to extract original and updated publication dates of web pages (Barbaresi, 2019),
- `newspaper` is mostly geared towards newspaper texts (Ou-Yang & Prezument, 2019),
- `news-please` is a news crawler that extracts structured information (Hamborg, Meuschke, Breitinger, & Gipp, 2017),

Two alternative packages are not tested here but that also could be used:

- `datefinder` (Koumjian, sudobangbang, & Senecal, 2020) features pattern-based date extraction for texts written in English,
- if dates are nowhere to be found, using `CarbonDate` (Atkins, DarkAngelZT, & Nwala, 2018) can be an option, however this is computationally expensive.

Benchmark

Test set

The experiments below are run on a collection of documents that are either typical for Internet articles (news outlets, blogs, including smaller ones) or non-standard and thus harder to process. They were selected from large collections of web pages in German. For the sake of completeness, a few documents in other languages were added (English, European languages, Chinese, and Arabic).

Evaluation

The evaluation script is available in the project repository: `tests/comparison.py`. The tests can be reproduced by cloning the repository, installing all necessary packages and running the evaluation script with the data provided in the `tests` directory.

Only documents with dates that are clearly able to be determined are considered for this benchmark. A given day is taken as unit of reference, meaning that results are converted to `%Y-%m-%d` format if necessary in order to make them comparable.

Time

The execution time (best of 3 tests) cannot be easily compared in all cases as some solutions perform a whole series of operations which are irrelevant to this task.

Errors

goose3's output is not always meaningful and/or in a standardized format, so these cases were discarded. news-please seems to have trouble with some encodings (e.g., in Chinese), in which case it leads to an exception.

Results

The results in Table 1 show that date extraction is not a completely solved task but one for which extractors have to resort to heuristics and guesses. The figures documenting recall and accuracy capture the real-world performance of the tools as the absence of a date output impacts the result.

Table 1: 225 web pages containing identifiable dates (as of 2020-07-29)

Python Package	Precision	Recall	Accuracy	F-Score	Time
newspaper 0.2.8	0.888	0.407	0.387	0.558	81.6
goose3 3.1.6	0.887	0.441	0.418	0.589	15.5
date_guesser 2.1.4	0.809	0.553	0.489	0.657	40.0
news-please 1.5.3	0.823	0.660	0.578	0.732	69.6
articleDateExtractor 0.20	0.817	0.635	0.556	0.714	6.8
htmldate 0.7.0 *(fast)*	**0.903**	0.907	0.827	0.905	**2.4**
htmldate[all] 0.7.0 *(extensive)*	0.889	**1.000**	**0.889**	**0.941**	3.8

Precision describes if the dates given as output are correct: `newspaper` and `goose3` fare well precision-wise but they fail to extract dates in a large majority of cases (poor recall). The difference in accuracy between `date_guesser` and `newspaper` is consistent with tests described on the website of the former.

It turns out that `htmldate` performs better than the other solutions overall. It is also noticeably faster than the strictly comparable packages (`articleDateExtractor` and `date_guesser`). Despite being measured on a sample, the higher accuracy and faster processing time are highly significant. Especially for smaller news outlets, websites, and blogs, as well as pages written in languages other than English (in this case mostly but not exclusively German), `htmldate` greatly extends date extraction coverage without sacrificing precision.

Note on the different versions:

- `htmldate[all]` means that additional components are added for performance and coverage. They can be installed with `pip/pip3/pipenv htmldate[all]` and result in differences with respect to accuracy (due to further linguistic analysis) and potentially speed (faster date parsing).
- The fast mode does not output as many dates (lower recall) but its guesses are more often correct (better precision).

Acknowledgements

This work has been supported by the ZDL research project (*Zentrum für digitale Lexikographie der deutschen Sprache*, zdl.org). Thanks to Yannick Kozmus (evaluation), user evolutionoftheuniverse (patterns for Turkish) and further contributors for testing and working on the package. Thanks to Daniel S. Katz, Geoff Bacon and Maarten van Gompel for reviewing this JOSS submission.

The following Python modules have been of great help: `lxml`, `ciso8601`, and `dateparser`. A few patterns are derived from `python-goose`, `metascraper`, `newspaper` and `articleDateExtractor`; this package extends their coverage and robustness significantly.

References

Atkins, G., DarkAngelZT, & Nwala, A. et a. (2018). CarbonDate: Estimating the age of web resources. *GitHub repository*. GitHub. Retrieved from https://github.com/oduwsdl/CarbonDate

Barbaresi, A. (2015). *Ad hoc and general-purpose corpus construction from web sources* (PhD thesis). École Normale Supérieure de Lyon.

Barbaresi, A. (2016). Efficient construction of metadata-enhanced web corpora. In P. Cook, S. Evert, R. Schäfer, & E. Stemle (Eds.), *Proceedings of the 10th Web as Corpus Workshop* (pp. 7–16). Association for Computational Linguistics. doi:10.18653/v1/w16-2602

Barbaresi, A. (2019). Generic Web Content Extraction with Open-Source Software. In *Proceedings of KONVENS 2019, Kaleidoscope Abstracts* (pp. 267–268). GSCL.

Baroni, M., Bernardini, S., Ferraresi, A., & Zanchetta, E. (2009). The WaCky Wide Web: a collection of very large linguistically processed web-crawled corpora. *Language Resources and Evaluation*, *43*(3), 209–226. doi:10.1007/s10579-009-9081-4

Carroll, C. D., & Valiukas, L. et a. (2019). Date Guesser: A library to extract a publication date from a web page, along with a measure of the accuracy. *GitHub repository*. GitHub. Retrieved from https://github.com/mitmedialab/date_guesser

Geva, R. et a. (2018). articleDateExtractor: Automatically extracts and normalizes an online article or blog post publication date. *GitHub repository*. GitHub. Retrieved from https://github.com/Webhose/article-date-extractor

Geyken, A., Barbaresi, A., Didakowski, J., Jurish, B., Wiegand, F., & Lemnitzer, L. (2017). Die Korpusplattform des "Digitalen Wörterbuchs der deutschen Sprache" (DWDS). *Zeitschrift für germanistische Linguistik*, *45*(2), 327–344. doi:10.1515/zgl-2017-0017

Grangier, X., Barrus, T., & Sidorov, I. et a. (2019). Goose3: A Python 3 compatible version of goose. *GitHub repository*. GitHub. Retrieved from https://github.com/goose3/goose3

Hamborg, F., Meuschke, N., Breitinger, C., & Gipp, B. (2017). news-please: A Generic News Crawler and Extractor. In M. Gaede, V. Trkulja, & V. Petra (Eds.), *Proceedings of the 15th International Symposium of Information Science* (pp. 218–223). Berlin.

Jo, E. S., & Gebru, T. (2020). Lessons from Archives: Strategies for Collecting Sociocultural Data in Machine Learning. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (pp. 306–316).

Kilgarriff, A. (2007). Googleology is bad science. *Computational Linguistics*, *33*(1), 147–151. doi:10.1162/coli.2007.33.1.147

Koumjian, A., sudobangbang, & Senecal, J. et a. (2020). datefinder: Find dates inside text using Python and get back datetime objects. *GitHub repository*. GitHub. Retrieved from https://github.com/akoumjian/datefinder

Olston, C., & Najork, M. (2010). Web Crawling. *Foundations and Trends in Information Retrieval*, *4*(3), 175–246.

Ou-Yang, L., & Prezument, Y. et a. (2019). Newspaper: News, full-text, and article metadata extraction in Python 3. *GitHub repository*. GitHub. Retrieved from https://github.com/codelucas/newspaper