Generic Web Content Extraction with Open-Source Software
Adrien Barbaresi

To cite this version:
Adrien Barbaresi. Generic Web Content Extraction with Open-Source Software. KONVENS 2019, Oct 2019, Erlangen, Germany. pp.267-268. hal-02447264

HAL Id: hal-02447264
https://hal.archives-ouvertes.fr/hal-02447264
Submitted on 21 Jan 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - NonCommercial - ShareAlike 4.0 International License
Abstract

Web corpus construction involves numerous design decisions. The software packages presented here can help facilitate collection and enhance corpus quality.

1 Problem description

Large “offline” web corpora are now standard throughout disciplines among the research community. Corpus construction notably involves “crawling, downloading, ‘cleaning’ and de-duplicating the data, then linguistically annotating it and loading it into a corpus query tool.” (Kilgarriff, 2007) As such, this process involves a significant number of design decisions and turning points in data processing. Depending on the purpose of data collection, a substantial filtering and quality assessment may also be needed. While some large-scale algorithms can be expected to smooth out irregularities, uses requiring a low margin of error as well as close reading approaches imply constant refinements and improvements in the constitution of the dataset and its processing, for example in the context of an aggregated lexical information platform (Geyken et al., 2017).

Recently, approaches using the CommonCrawl 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, it is clear that such data should not be used without some filtering. Corresponding to the potential lack of metadata is a lack of information regarding the content, whose adequacy, focus and quality are the object of a post hoc evaluation (Baroni et al., 2009). Because of the vastly increasing variety of corpora, text types and use cases, it becomes more and more difficult to assess the usefulness and appropriateness of certain web texts for given research objectives. Most notably, an essential operation in corpus construction consists in retaining the desired content while discarding the rest, a polyonymous task referring to peculiar sub-tasks or to the whole, most notably web scraping, boilerplate removal, web page cleaning, or web content extraction (Lejeune and Zhu, 2018).

Consequently, a significant challenge lies in the ability to extract and pre-process web data to meet scientific expectations with respect to corpus quality (Barbaresi, 2019b). In the following, two libraries grounding on previous efforts (Barbaresi, 2016) are presented which can help enhancing the quality of webcorpora. They are both relying on Python, currently one of the most used programming languages, within and outside of academia.2

2 htmldate: finding the publishing date

The htmldate library (Barbaresi, 2019a) can find both the original and the updated publication dates of web pages. It involves a rule-based examination of the semantic structure of HTML documents, using a combination of tree traversal, common structural patterns, text-based heuristics and robust date extraction. First, it uses the markup in the document header, where common patterns are used to identify relevant elements (e.g. link and meta elements) including common standards and idiosyncrasies of content management systems. Second, it looks for cues within the HTML code as the whole document is searched for structural markers: abbr/time elements and a series of attributes (e.g. postmetadata). Finally, a series of heuristics is run on text and markup. The library currently focuses on texts written in English and German, it is used in production and is documented online.3

---

1https://commoncrawl.org
3https://github.com/adbar/htmldate
3 trafilatura: targeting the main content

The second software component focuses on 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 comments. Distinguishing between whole page and essential parts can help to alleviate many quality problems related to web texts. While this is particularly useful for de-duplication, other tasks related to content extraction also benefit from a cleaner text base. In the concrete case of linguistic and lexicographic research, it allows for content checks on the only portion of the document that really counts.

Although most corresponding Python modules are not actively maintained, the following alternatives perform similar tasks: dragnet\footnote{https://github.com/dragnet-org/dragnet} features include the extraction of comments (separated from the rest), duplicate detection at sentence, paragraph and document level using a least recently used (LRU) cache, TEI/XML output, and language detection on the extracted content.

4 Conclusions

This ongoing work constitutes a step towards the ability to extract and pre-process web texts in order to make them available in clearly definable and coherent collections. In both software components presented here, all the operations needed from web page download to HTML parsing are handled, including scraping and textual analysis. URLs, HTML files or parsed HTML trees are given as input and the libraries output strings in the desired format. They can be used on common operating systems, by themselves, within Python, or on the command-line. Their versatility allows for work on different languages and corpus types as well as for inclusion in various processing chains.

References


---

\footnote{https://github.com/dragnet-org/dragnet}
\footnote{https://github.com/buriy/python-readability}
\footnote{https://github.com/Alir3z4/html2text}
\footnote{https://tei-c.org}
\footnote{https://github.com/adbar/trafilatura}