newspaper Documentation

Release 0.0.2

Lucas Ou-Yang

Jun 18, 2020

Contents

Inspired by requests for its simplicity and powered by lxml for its speed:

> "Newspaper is an amazing python library for extracting & curating articles." – tweeted by Kenneth Reitz, Author of requests
>
> "Newspaper delivers Instapaper style article extraction." – The Changelog

Newspaper is a Python3 library! View on Github here, or, view our **deprecated and buggy** Python2 branch

CHAPTER 1

A Glance:

```
>>> from newspaper import Article

>>> url = 'http://fox13now.com/2013/12/30/new-year-new-laws-obamacare-pot-guns-and-
↪drones/'
>>> article = Article(url)
```

```
>>> article.download()

>>> article.html
'<!DOCTYPE HTML><html itemscope itemtype="http://...'
```

```
>>> article.parse()

>>> article.authors
['Leigh Ann Caldwell', 'John Honway']

>>> article.publish_date
datetime.datetime(2013, 12, 30, 0, 0)

>>> article.text
'Washington (CNN) -- Not everyone subscribes to a New Year's resolution...'

>>> article.top_image
'http://someCDN.com/blah/blah/blah/file.png'

>>> article.movies
['http://youtube.com/path/to/link.com', ...]
```

```
>>> article.nlp()

>>> article.keywords
['New Years', 'resolution', ...]
```

(continues on next page)

(continued from previous page)

```
>>> article.summary
'The study shows that 93% of people ...'
```

```
>>> import newspaper

>>> cnn_paper = newspaper.build('http://cnn.com')

>>> for article in cnn_paper.articles:
>>>     print(article.url)
http://www.cnn.com/2013/11/27/justice/tucson-arizona-captive-girls/
http://www.cnn.com/2013/12/11/us/texas-teen-dwi-wreck/index.html
...

>>> for category in cnn_paper.category_urls():
>>>     print(category)

http://lifestyle.cnn.com
http://cnn.com/world
http://tech.cnn.com
...

>>> cnn_article = cnn_paper.articles[0]
>>> cnn_article.download()
>>> cnn_article.parse()
>>> cnn_article.nlp()
...
```

```
>>> from newspaper import fulltext

>>> html = requests.get(...).text
>>> text = fulltext(html)
```

Newspaper has *seamless* language extraction and detection. If no language is specified, Newspaper will attempt to auto detect a language.

```
>>> from newspaper import Article
>>> url = 'http://www.bbc.co.uk/zhongwen/simp/chinese_news/2012/12/121210_hongkong_
↪politics.shtml'

>>> a = Article(url, language='zh') # Chinese

>>> a.download()
>>> a.parse()

>>> print(a.text[:150])


1210




>>> print(a.title)
```

If you are certain that an *entire* news source is in one language, **go ahead and use the same api :)**

```
>>> import newspaper
>>> sina_paper = newspaper.build('http://www.sina.com.cn/', language='zh')

>>> for category in sina_paper.category_urls():
>>>     print(category)
http://health.sina.com.cn
http://eladies.sina.com.cn
http://english.sina.com
...

>>> article = sina_paper.articles[0]
>>> article.download()
>>> article.parse()

>>> print(article.text)

“”
 ----



...

>>> print(article.title)
00 __
__
```

CHAPTER 2

Documentation

Check out The Documentation for full and detailed guides using newspaper.

Interested in adding a new language for us? Refer to: Docs - Adding new languages

CHAPTER 3

Features

- Multi-threaded article download framework
- News url identification
- Text extraction from html
- Top image extraction from html
- All image extraction from html
- Keyword extraction from text
- Summary extraction from text
- Author extraction from text
- Google trending terms extraction
- Works in 10+ languages (English, Chinese, German, Arabic, . . .)

```
>>> import newspaper
>>> newspaper.languages()

Your available languages are:
input code      full name

  ar              Arabic
  be              Belarusian
  bg              Bulgarian
  da              Danish
  de              German
  el              Greek
  en              English
  es              Spanish
  et              Estonian
  fa              Persian
  fi              Finnish
  fr              French
```

(continues on next page)

(continued from previous page)

```
  he              Hebrew
  hi              Hindi
  hr              Croatian
  hu              Hungarian
  id              Indonesian
  it              Italian
  ja              Japanese
  ko              Korean
  lt              Lithuanian
  mk              Macedonian
  nb              Norwegian (Bokmål)
  nl              Dutch
  no              Norwegian
  pl              Polish
  pt              Portuguese
  ro              Romanian
  ru              Russian
  sl              Slovenian
  sr              Serbian
  sv              Swedish
  sw              Swahili
  th              Thai
  tr              Turkish
  uk              Ukrainian
  vi              Vietnamese
  zh              Chinese
```

CHAPTER 4

Get it now

Run `pip3 install newspaper3k`

NOT `pip3 install newspaper`

On python3 you must install `newspaper3k`, **not** `newspaper`. `newspaper` is our python2 library. Although installing newspaper is simple with pip, you will run into fixable issues if you are trying to install on ubuntu.

If you are on Debian / Ubuntu, install using the following:

- Install `pip3` command needed to install `newspaper3k` package:

```
$ sudo apt-get install python3-pip
```

- Python development version, needed for Python.h:

```
$ sudo apt-get install python-dev
```

- lxml requirements:

```
$ sudo apt-get install libxml2-dev libxslt-dev
```

- For PIL to recognize .jpg images:

```
$ sudo apt-get install libjpeg-dev zlib1g-dev libpng12-dev
```

NOTE: If you find problem installing `libpng12-dev`, try installing `libpng-dev`.

- Download NLP related corpora:

```
$ curl https://raw.githubusercontent.com/codelucas/newspaper/master/download_
↪corpora.py | python3
```

- Install the distribution via pip:

```
$ pip3 install newspaper3k
```

If you are on OSX, install using the following, you may use both homebrew or macports:

```
$ brew install libxml2 libxslt

$ brew install libtiff libjpeg webp little-cms2

$ pip3 install newspaper3k

$ curl https://raw.githubusercontent.com/codelucas/newspaper/master/download_corpora.
↪py | python3
```

Otherwise, install with the following:

NOTE: You will still most likely need to install the following libraries via your package manager

- PIL: `libjpeg-dev zlib1g-dev libpng12-dev`
- lxml: `libxml2-dev libxslt-dev`
- Python Development version: `python-dev`

```
$ pip3 install newspaper3k

$ curl https://raw.githubusercontent.com/codelucas/newspaper/master/download_corpora.
↪py | python3
```

Using python 2.X? We support python 2, however development work has stopped on the 2.X branch for a few years now so it is behind in features and is more buggy. See python 2 installation instructions here

CHAPTER 5

Development

If you'd like to contribute and hack on the newspaper project, feel free to clone a development version of this repository locally:

```
git clone git://github.com/codelucas/newspaper.git
```

Once you have a copy of the source, you can embed it in your Python package, or install it into your site-packages easily:

```
$ pip3 install -r requirements.txt
$ python3 setup.py install
```

Feel free to give our testing suite a shot, everything is mocked!:

```
$ python3 tests/unit_tests.py
```

Planning on tweaking our full-text algorithm? Add the `fulltext` parameter:

```
$ python3 tests/unit_tests.py fulltext
```

CHAPTER 6

User Guide

6.1 Quickstart

Eager to get started? This page gives a good introduction in how to get started with newspaper. This assumes you already have newspaper installed. If you do not, head over to the Installation section.

6.1.1 Building a news source

Source objects are an abstraction of online news media websites like CNN or ESPN. You can initialize them in two *different* ways.

Building a `Source` will extract its categories, feeds, articles, brand, and description for you.

You may also provide configuration parameters like `language`, `browser_user_agent`, and etc seamlessly. Navigate to the *advanced* section for details.

```
>>> import newspaper
>>> cnn_paper = newspaper.build('http://cnn.com')

>>> sina_paper = newspaper.build('http://www.lemonde.fr/', language='fr')
```

However, if needed, you may also play with the lower level `Source` object as described in the *advanced* section.

6.1.2 Extracting articles

Every news source has a set of *recent* articles.

The following examples assume that a news source has been initialized and built.

```
>>> for article in cnn_paper.articles:
>>>     print(article.url)
```

(continues on next page)

(continued from previous page)

```
u'http://www.cnn.com/2013/11/27/justice/tucson-arizona-captive-girls/'
u'http://www.cnn.com/2013/12/11/us/texas-teen-dwi-wreck/index.html'
...

>>> print(cnn_paper.size()) # cnn has 3100 articles
3100
```

6.1.3 Article caching

By default, newspaper caches all previously extracted articles and **eliminates any article which it has already extracted**.

This feature exists to prevent duplicate articles and to increase extraction speed.

```
>>> cbs_paper = newspaper.build('http://cbs.com')
>>> cbs_paper.size()
1030

>>> cbs_paper = newspaper.build('http://cbs.com')
>>> cbs_paper.size()
2
```

The return value of `cbs_paper.size()` changes from 1030 to 2 because when we first crawled cbs we found 1030 articles. However, on our second crawl, we eliminate all articles which have already been crawled.

This means **2** new articles have been published since our first extraction.

You may opt out of this feature with the `memoize_articles` parameter.

You may also pass in the lower level‘‘Config‘‘ objects as covered in the *advanced* section.

```
>>> import newspaper

>>> cbs_paper = newspaper.build('http://cbs.com', memoize_articles=False)
>>> cbs_paper.size()
1030

>>> cbs_paper = newspaper.build('http://cbs.com', memoize_articles=False)
>>> cbs_paper.size()
1030
```

6.1.4 Extracting Source categories

```
>>> for category in cnn_paper.category_urls():
>>>     print(category)

u'http://lifestyle.cnn.com'
u'http://cnn.com/world'
u'http://tech.cnn.com'
...
```

6.1.5 Extracting Source feeds

```
>>> for feed_url in cnn_paper.feed_urls():
>>>     print(feed_url)

u'http://rss.cnn.com/rss/cnn_crime.rss'
u'http://rss.cnn.com/rss/cnn_tech.rss'
...
```

6.1.6 Extracting Source brand & description

```
>>> print(cnn_paper.brand)
u'cnn'

>>> print(cnn_paper.description)
u'CNN.com delivers the latest breaking news and information on the latest...'
```

6.1.7 News Articles

Article objects are abstractions of news articles. For example, a news `Source` would be CNN while a news `Article` would be a specific CNN article. You may reference an `Article` from an existing news `Source` or initialize one by itself.

Referencing it from a `Source`.

```
>>> first_article = cnn_paper.articles[0]
```

Initializing an `Article` by itself.

```
>>> from newspaper import Article
>>> first_article = Article(url="http://www.lemonde.fr/...", language='fr')
```

Note the similar `language=` named paramater above. All the config parameters as described for `Source` objects also apply for `Article` objects! **Source and Article objects have a very similar api**.

Initializing an `Article` with the particular content-type ignoring.

There is option to skip loading of articles with particular content-type, that can be useful if it is not desired to have delays because of long PDF resources. The default html value for the particular content type can be provided and then used in order to define the actual content-type of the article

```
>>> from newspaper import Article
>>> pdf_defaults = {"application/pdf": "%PDF-",
                    "application/x-pdf": "%PDF-",
                    "application/x-bzpdf": "%PDF-",
                    "application/x-gzpdf": "%PDF-"}
>>> pdf_article = Article(url='https://www.adobe.com/pdf/pdfs/ISO32000-
↪1PublicPatentLicense.pdf',
                                        ignored_content_types_defaults=pdf_defaults)
>>> pdf_article.download()
>>> print(pdf_article.html)
%PDF-
```

There are endless possibilities on how we can manipulate and build articles.

6.1.8 Downloading an Article

We begin by calling `download()` on an article. If you are interested in how to quickly download articles concurrently with multi-threading check out the *advanced* section.

```
>>> first_article = cnn_paper.articles[0]

>>> first_article.download()

>>> print(first_article.html)
u'<!DOCTYPE HTML><html itemscope itemtype="http://...'

>>> print(cnn_paper.articles[7].html)
u'' fail, not downloaded yet
```

6.1.9 Parsing an Article

You may also extract meaningful content from the html, like authors and body-text. You **must** have called `download()` on an article before calling `parse()`.

```
>>> first_article.parse()

>>> print(first_article.text)
u'Three sisters who were imprisoned for possibly...'

>>> print(first_article.top_image)
u'http://some.cdn.com/3424hfd4565sdfgdg436/

>>> print(first_article.authors)
[u'Eliott C. McLaughlin', u'Some CoAuthor']

>>> print(first_article.title)
u'Police: 3 sisters imprisoned in Tucson home'

>>> print(first_article.images)
['url_to_img_1', 'url_to_img_2', 'url_to_img_3', ...]

>>> print(first_article.movies)
['url_to_youtube_link_1', ...] # youtube, vimeo, etc
```

6.1.10 Performing NLP on an Article

Finally, you may extract out natural language properties from the text. You **must** have called both `download()` and `parse()` on the article before calling `nlp()`.

As of the current build, nlp() features only work on western languages.

```
>>> first_article.nlp()

>>> print(first_article.summary)
u'...imprisoned for possibly a constant barrage...'

>>> print(first_article.keywords)
[u'music', u'Tucson', ... ]
```

(continues on next page)

(continued from previous page)

```
>>> print(cnn_paper.articles[100].nlp()) # fail, not been downloaded yet
Traceback (...
ArticleException: You must parse an article before you try to..
```

`nlp()` is expensive, as is `parse()`, make sure you actually need them before calling them on all of your articles! In some cases, if you just need urls, even `download()` is not necessary.

6.1.11 Easter Eggs

Here are random but hopefully useful features! `hot()` returns a list of the top trending terms on Google using a public api. `popular_urls()` returns a list of popular news source urls.. In case you need help choosing a news source!

```
>>> import newspaper

>>> newspaper.hot()
['Ned Vizzini', Brian Boitano', Crossword Inventor', 'Alex & Sierra', ... ]

>>> newspaper.popular_urls()
['http://slate.com', 'http://cnn.com', 'http://huffingtonpost.com', ... ]

>>> newspaper.languages()

Your available languages are:
input code      full name

  ar              Arabic
  de              German
  en              English
  es              Spanish
  fr              French
  he              Hebrew
  it              Italian
  ko              Korean
  no              Norwegian
  fa              Persian
  pl              Polish
  pt              Portuguese
  sv              Swedish
  zh              Chinese
  uk              Ukrainian
  sw              Swahili
  bg              Bulgarian
  hr              Croatian
  ro              Romanian
  sl              Slovenian
  sr              Serbian
  et              Estonian
  ja              Japanese
  be              Belarusian
  lt              Lithuanian
```

6.2 Advanced

This section of the docs shows how to do some useful but advanced things with newspaper.

6.2.1 Multi-threading article downloads

Downloading articles one at a time is slow. But spamming a single news source like cnn.com with tons of threads or with ASYNC-IO will cause rate limiting and also doing that is very mean.

We solve this problem by allocating 1-2 threads per news source to both greatly speed up the download time while being respectful.

```
>>> import newspaper
>>> from newspaper import news_pool

>>> slate_paper = newspaper.build('http://slate.com')
>>> tc_paper = newspaper.build('http://techcrunch.com')
>>> espn_paper = newspaper.build('http://espn.com')

>>> papers = [slate_paper, tc_paper, espn_paper]
>>> news_pool.set(papers, threads_per_source=2) # (3*2) = 6 threads total
>>> news_pool.join()

At this point, you can safely assume that download() has been
called on every single article for all 3 sources.

>>> print(slate_paper.articles[10].html)
u'<html> ...'
```

6.2.2 Keeping Html of main body article

Keeping the html of just an article's body text is helpbut because it allows you to retain some of the semantic information in the html. Also it will help if you end up displaying the extracted article somehow.

Here is how to do so:

```
>>> from newspaper import Article

>>> a = Article('http://www.cnn.com/2014/01/12/world/asia/north-korea-charles-smith/
↪index.html'
    , keep_article_html=True)

>>> a.download()
>>> a.parse()

>>> a.article_html
u'<div> \n<p><strong>(CNN)</strong> -- Charles Smith insisted Sunda...'
```

The lxml (dom object) and top_node (chunk of dom that contains our 'Article') are also cached incase users would like to use them.

Access **after parsing()** with:

```
>>> a.download()
>>> a.parse()
>>> a.clean_dom
<lxml object ...  >

>>> a.clean_top_node
<lxml object ...  >
```

6.2.3 Adding new languages

First, please reference this file and read from the highlighted line all the way down to the end of the file.

https://github.com/codelucas/newspaper/blob/master/newspaper/text.py#L57

One aspect of our text extraction algorithm revolves around counting the number of **stopwords** present in a text. Stopwords are: *some of the most common, short function words, such as the, is, at, which, and on* in a language.

Reference this line to see it in action: https://github.com/codelucas/newspaper/blob/master/newspaper/extractors.py#L668

So for latin languages, it is pretty basic. We first provide a list of stopwords in `stopwords-<language-code>.txt` form. We then take some input text and tokenize it into words by splitting the white space. After that we perform some bookkeeping and then proceed to count the number of stopwords present.

For non-latin languages, as you may have noticed in the code above, we need to tokenize the words in a different way, *splitting by whitespace simply won't work for languages like Chinese or Arabic*. For the Chinese language we are using a whole new open source library called *jieba* to split the text into words. For arabic we are using a special nltk tokenizer to do the same job.

So, to add full text extraction to a new (non-latin) language, we need:

1. Push up a stopwords file in the format of `stopwords-<2-char-language-code>.txt` in `newspaper/resources/text/.`

2. Provide a way of splitting/tokenizing text in that foreign language into words. Here are some examples for Chinese, Arabic, English

For latin languages:

1. Push up a stopwords file in the format of `stopwords-<2-char-language-code>.txt` in `newspaper/resources/text/.` and we are done!

Finally, add the new language to the list of available languages in the following files:

- README.rst
- docs/index.rst
- docs/user_guide/quickstart.rst
- newspaper/utils.py

6.2.4 Explicitly building a news source

Instead of using the `newspaper.build(..)` api, we can take one step lower into newspaper's `Source` api.

```
>>> from newspaper import Source
>>> cnn_paper = Source('http://cnn.com')

>>> print(cnn_paper.size()) # no articles, we have not built the source
0

>>> cnn_paper.build()
>>> print(cnn_paper.size())
3100
```

Note the `build()` method above. You may go lower level and de-abstract it for absolute control over how your sources are constructed.

```
>>> cnn_paper = Source('http://cnn.com')
>>> cnn_paper.download()
>>> cnn_paper.parse()
>>> cnn_paper.set_categories()
>>> cnn_paper.download_categories()
>>> cnn_paper.parse_categories()
>>> cnn_paper.set_feeds()
>>> cnn_paper.download_feeds()
>>> cnn_paper.generate_articles()

>>> print(cnn_paper.size())
3100
```

And voila, we have mimic'd the `build()` method. In the above sequence, every method is dependant on the method above it. Stop whenever you wish.

6.2.5 Parameters and Configurations

Newspaper provides two api's for users to configure their `Article` and `Source` objects. One is via named parameter passing **recommended** and the other is via `Config` objects.

Here are some named parameter passing examples:

```
>>> import newspaper
>>> from newspaper import Article, Source

>>> cnn = newspaper.build('http://cnn.com', language='en', memoize_articles=False)

>>> article = Article(url='http://cnn.com/french/...', language='fr', fetch_
↪images=False)

>>> cnn = Source(url='http://latino.cnn.com/...', language='es', request_timeout=10,
                                                    number_threads=20)
```

Here are some examples of how Config objects are passed.

```
>>> import newspaper
>>> from newspaper import Config, Article, Source

>>> config = Config()
>>> config.memoize_articles = False

>>> cbs_paper = newspaper.build('http://cbs.com', config)
```

(continues on next page)

(continued from previous page)

```
>>> article_1 = Article(url='http://espn/2013/09/...', config)

>>> cbs_paper = Source('http://cbs.com', config)
```

Here is a full list of the configuration options:

`keep_article_html`, default False, "set to True if you want to preserve html of body text"

`http_success_only`, default True, "set to False to capture non 2XX responses as well"

`MIN_WORD_COUNT`, default 300, "num of word tokens in article text"

`MIN_SENT_COUNT`, default 7, "num of sentence tokens"

`MAX_TITLE`, default 200, "num of chars in article title"

`MAX_TEXT`, default 100000, "num of chars in article text"

`MAX_KEYWORDS`, default 35, "num of keywords in article"

`MAX_AUTHORS`, default 10, "num of author names in article"

`MAX_SUMMARY`, default 5000, "num of chars of the summary"

`MAX_SUMMARY_SENT`, default 5, "num of sentences in summary"

`MAX_FILE_MEMO`, default 20000, "python setup.py sdist bdist_wininst upload"

`memoize_articles`, default True, "cache and save articles run after run"

`fetch_images`, default True, "set this to false if you don't care about getting images"

`follow_meta_refresh`, default False, "follows a redirect url in a meta refresh html tag"

`image_dimension_ration`, default 16/9.0, "max ratio for height/width, we ignore if greater"

`language`, default 'en', "run `newspaper.languages()` to see available options."

`browser_user_agent`, default 'newspaper/%s' % __version__

`request_timeout`, default 7

`number_threads`, default 10, "number of threads when mthreading"

`verbose`, default False, "turn this on when debugging"

You may notice other config options in the `newspaper/configuration.py` file, however, they are private, **please do not toggle them**.

6.2.6 Caching

TODO

6.2.7 Specifications

Here, we will define exactly *how* newspaper handles a lot of the data extraction.

TODO

CHAPTER 7

Demo

View a working online demo here: http://newspaper-demo.herokuapp.com This is another working online demo: http://newspaper.chinazt.cc/

CHAPTER 8

LICENSE

Authored and maintained by Lucas Ou-Yang.

Parse.ly sponsored some work on newspaper, specifically focused on automatic extraction.

Newspaper uses a lot of python-goose's parsing code. View their license here.

Please feel free to email & contact me if you run into issues or just would like to talk about the future of this library and news extraction in general!