wayback Documentation

Release 0.2.2

Contributors

Mar 28, 2020

Contents

Wayback is a Python client to the Internet Archive Wayback Machine.

CHAPTER 1

Installation

At the command line:

```
$ pip install wayback
```

CHAPTER 2

Usage

Search for historical mementos (archived copies) of a URL. Download metadata about the mementos and/or the memento content itself.

2.1 Tutorial

2.1.1 What is the earliest memento of nasa.gov?

Instantiate a `WaybackClient`.

```
In [1]: from wayback import WaybackClient

In [2]: client = WaybackClient()
```

Search for all Wayback's records for nasa.gov.

```
In [3]: results = client.search('nasa.gov')
```

This statement should execute fairly quickly because it doesn't actually do much work. The object we get back, `results`, is a *generator*, a "lazy" object from which we can pull results, one at a time. As we pull items out of it, it loads them as needed from the Wayback Machine in chronological order. We can see that `results` by itself is not informative:

```
In [4]: results
Out[4]: <generator object WaybackClient.search at 0x7f5652ba2840>
```

There are couple ways to pull items out of generator like `results`. One simple way is to use the built-in Python function `next()`, like so:

```
In [5]: record = next(results)
```

This takes a moment to run because, now that we've asked to see the first item in the generator, this lazy object goes to fetch a chunk of results from the Wayback Machine. Looking at the record in detail,

```
In [6]: record
Out[6]: CdxRecord(key='gov,nasa)/', timestamp=datetime.datetime(1996, 12, 31, 23, 58,␣
↪47), url='http://www.nasa.gov/', mime_type='text/html', status_code=200, digest=
↪'MGIGF4GRGGF5GKV6VNCBAXOE3OR5BTZC', length=1811, raw_url='http://web.archive.org/
↪web/19961231235847id_/http://www.nasa.gov/', view_url='http://web.archive.org/web/
↪19961231235847/http://www.nasa.gov/')
```

we can find our answer: Wayback's first memento of nasa.gov was in 1996. We can use dot access on `record` to access the timestamp specifically.

```
In [7]: record.timestamp
Out[7]: datetime.datetime(1996, 12, 31, 23, 58, 47)
```

2.1.2 How many times does the word 'mars' appear on nasa.gov?

Above, we access the metadata for the oldest memento on nasa.gov, stored in the variable `record`. Starting from where we left off, we'll access the *content* of the memento and do a very simple analysis.

The Wayback Machine provides two ways to look at the data it has captured. There is a copy edited for human viewers on the web, available at the record's `view_url`, and there is the original copy of what was captured when the page was originally scraped, availabe at the record's `raw_url`. For analysis purposes, we generally want the `raw_url`.

Let's download the raw content using `WaybackClient`. (You could download the content directly with an HTTP library like `requests`, but `WaybackClient` adds extra tools for dealing with Wayback Machine servers.)

```
In [8]: response = client.get_memento(record.raw_url)

In [9]: content = response.content.decode()
```

We can use the built-in method `count` on strings to count the number of times that `'mars'` appears in the content.

```
In [10]: content.count('mars')
Out[10]: 30
```

This is case-sensitive, so to be more accurate we should convert the content to lowercase first.

```
In [11]: content.lower().count('mars')
Out[11]: 39
```

We picked up a couple additional occurrences that the original count missed.

2.2 API Documentation

The Wayback Machine exposes its data through two different mechanisms, implementing two different standards for archival data, the CDX API and the Memento API. We implement a Python client that can speak both.

class wayback.**WaybackClient**(*session=None*)

A client for retrieving data from the Internet Archive's Wayback Machine.

You can use a WaybackClient as a context manager. When exiting, it will close the session it's using (if you've passed in a custom session, make sure not to use the context manager functionality unless you want to live dangerously).

Parameters

session [`requests.Session`, optional]

search(*self*, *url*, ***, *matchType=None*, *limit=None*, *offset=None*, *fastLatest=None*, *gzip=None*, *from_date=None*, *to_date=None*, *filter_field=None*, *collapse=None*, *showResumeKey=True*, *resumeKey=None*, *page=None*, *pageSize=None*, *resolveRevisits=True*, *skip_malformed_results=True*, ***kwargs*)

Search archive.org's CDX API for all captures of a given URL.

This will automatically page through all results for a given search.

Returns an iterator of CdxRecord objects. The StopIteration value is the total count of found captures.

Note that even URLs without wildcards may return results with different URLs. Search results are matched by url_key, which is a SURT-formatted, canonicalized URL:

- Does not differentiate between HTTP and HTTPS
- Is not case-sensitive
- Treats `www.` and `www*.` subdomains the same as no subdomain at all

Note not all CDX API parameters are supported. In particular, this does not support: *output*, *fl*, *showDupeCount*, *showSkipCount*, *lastSkipTimestamp*, *showNumPages*, *showPagedIndex*.

Parameters

url [str] The URL to query for captures of.

matchType [str, optional] Must be one of 'exact', 'prefix', 'host', or 'domain'. The default value is calculated based on the format of *url*.

limit [int, optional] Maximum number of results per page (this iterator will continue to move through all pages unless *showResumeKey=False*, though).

offset [int, optional] Skip the first N results.

fastLatest [bool, optional] Get faster results when using a negative value for *limit*. It may return a variable number of results.

gzip [bool, optional] Whether output should be gzipped.

from_date [datetime, optional] Only include captures after this date. Equivalent to the *from* argument in the CDX API. If it does not have a time zone, it is assumed to be in UTC.

to_date [datetime, optional] Only include captures before this date. Equivalent to the *to* argument in the CDX API. If it does not have a time zone, it is assumed to be in UTC.

filter_field [str, optional] A filter for any field in the results. Equivalent to the *filter* argument in the CDX API. (format: *[!]field:regex*)

collapse [str, optional] Collapse consecutive results that match on a given field. (format: *fieldname* or *fieldname:N* – N is the number of chars to match.)

showResumeKey [bool, optional] If False, don't continue to iterate through all pages of results. The default value is True

resumeKey [str, optional] Start returning results from a specified resumption point/offset. The value for this is supplied by the previous page of results when *showResumeKey* is True.

page [int, optional] If using paging start from this page number (note: paging, as opposed to the using *resumeKey* is somewhat complicated because of the interplay with indexes and index sizes).

pageSize [int, optional] The number of index blocks to examine for each page of results. Index blocks generally cover about 3,000 items, so setting *pageSize=1* might return anywhere from 0 to 3,000 results per page.

resolveRevists [bool, optional] Attempt to resolve *warc/revisit* records to their actual content type and response code. Not supported on all CDX servers. Defaults to True.

skip_malformed_results [bool, optional] If true, don't yield records that look like they have no actual memento associated with them. Some crawlers will erroneously attempt to capture bad URLs like *http://mailto:someone@domain.com* or *http://...* and so on. This is a filter performed client side and is not a CDX API argument. (Default: True)

****kwargs** Any additional CDX API options.

Yields

version: CdxRecord A *CdxRecord* encapsulating one capture or revisit

Raises

UnexpectedResponseFormat If the CDX response was not parseable.

References

- https://github.com/internetarchive/wayback/tree/master/wayback-cdx-server

get_memento(*self*, *url*, *exact=True*, *exact_redirects=None*, *target_window=86400*)

Fetch a memento from the Wayback Machine. This retrieves the content that was ultimately returned from a memento, following any redirects that were present at the time the memento was captured. (That is, if *http://example.com/a* redirected to *http://example.com/b*, this returns the memento for */b* when you request */a*.)

Parameters

url [string] URL of memento in Wayback (e.g. *http://web.archive.org/web/20180816111911id_/http://www.nws.no*

exact [boolean, optional] If false and the requested memento either doesn't exist or can't be played back, this returns the closest-in-time memento to the requested one, so long as it is within *target_window*. Default: True

exact_redirects [boolean, optional] If false and the requested memento is a redirect whose *target* doesn't exist or or can't be played back, this returns the closest- in-time memento to the intended target, so long as it is within *target_window*. If unset, this will be the same as *exact*.

target_window [int, optional] If the memento is of a redirect, allow up to this many seconds between the capture of the redirect and the capture of the target URL. (Note this does NOT apply when the originally requested memento didn't exist and wayback redirects to the next-closest-in- -time one. That will always raise a MementoPlaybackError.) Defaults to 86,400 (24 hours).

Returns

dict [requests.Response] An HTTP response with the content of the memento, including a history of any redirects involved.

class wayback.**CdxRecord**(*key*, *timestamp*, *url*, *mime_type*, *status_code*, *digest*, *length*, *raw_url*, *view_url*)

Item from iterable of results returned by *WaybackClient.search()*

These attributes contain information provided directly by CDX.

digest
Content hashed as a base 32 encoded SHA-1.

key
SURT-formatted URL

length
Size of captured content in bytes, such as `2767`. This may be innacurate. If the record is a "revisit record", indicated by MIME type `'warc/revisit'`, the length seems to be the length of the reference, not the length of the content itself.

mime_type
MIME type of record, such as `'text/html'`, `'warc/revisit'` or `'unk'` ("unknown") if this information was not captured.

status_code
Status code returned by the server when the record was captured, such as `200`. This is may be `None` if the record is a revisit record.

timestamp
The capture time represented as a `datetime.datetime`, such as `datetime.datetime(1996, 12, 31, 23, 58, 47)`.

url
The URL that was captured by this record, such as `'http://www.nasa.gov/'`.

And these attributes are synthesized from the information provided by CDX.

raw_url
The URL to the raw captured content, such as `'http://web.archive.org/web/19961231235847id_/http://www.nasa.gov/'`.

view_url
The URL to the public view on Wayback Machine. In this view, the links and some subresources in the document are rewritten to point to Wayback URLs. There is also a navigation panel around the content. Example URL: `'http://web.archive.org/web/19961231235847/http://www.nasa.gov/'`.

class wayback.**WaybackSession**(*retries=6*, *backoff=2*, *timeout=None*, *user_agent=None*)
A custom session object that network pools connections and resources for requests to the Wayback Machine.

Parameters

retries [int, optional] The maximum number of retries for requests.

backoff [int or float, optional] Number of seconds from which to calculate how long to back off and wait when retrying requests. The first retry is always immediate, but subsequent retries increase by powers of 2:

seconds = backoff * 2 ^ (retry number - 1)

So if this was *4*, retries would happen after the following delays: 0 seconds, 4 seconds, 8 seconds, 16 seconds, ...

timeout [int or float or tuple of (int or float, int or float), optional] A timeout to use for all requests. If not set, there will be no no explicit timeout. See the Requests docs for more: http://docs.python-requests.org/en/master/user/advanced/#timeouts

user_agent [str, optional] A custom user-agent string to use in all requests. Defaults to: *wayback/{version} (+https://github.com/edgi-govdata-archiving/wayback)*

reset(*self*)
Reset any network connections the session is using.

2.2.1 Utility Functions

wayback.**memento_url_data**(*memento_url*)
Get the original URL and date that a memento URL represents a capture of.

Examples

Extract original URL and date.

```
>>> url = ('http://web.archive.org/web/20170813195036/'
...        'https://arpa-e.energy.gov/?q=engage/events-workshops')
>>> memento_url_data(url)
('https://arpa-e.energy.gov/?q=engage/events-workshops',
 datetime.datetime(2017, 8, 13, 19, 50, 36))
```

2.2.2 Exception Classes

class wayback.exceptions.**WaybackException**
Base exception class for all Wayback-specific errors.

class wayback.exceptions.**UnexpectedResponseFormat**
Raised when data returned by the Wayback Machine is formatted in an unexpected or unparseable way.

class wayback.exceptions.**BlockedByRobotsError**
Raised when a URL can't be queried in Wayback because it was blocked by a site's *robots.txt* file.

class wayback.exceptions.**MementoPlaybackError**
Raised when a Memento can't be 'played back' (loaded) by the Wayback Machine for some reason. This is a server-side issue, not a problem in parsing data from Wayback.

class wayback.exceptions.**WaybackRetryError**(*retries*, *total_time*, *causal_error*)
Raised when a request to the Wayback Machine has been retried and failed too many times. The number of tries before this exception is raised generally depends on your *WaybackSession* settings.

Attributes

retries [int] The number of retries that were attempted.

cause [Exception] The actual, underlying error that would have caused a retry.

time [int] The total time spent across all retried requests, in seconds.

CHAPTER 3

Release History

3.1 v0.2.2 (2020-02-13)

When errors were raised or redirects were involved in `WaybackClient.get_memento()`, it was previously possible for connections to be left hanging open. Wayback now works harder to make sure connections aren't left open.

This release also updates the default user agent string to include the repo URL. It now looks like: `wayback/0.2.2 (+https://github.com/edgi-govdata-archiving/wayback)`

3.2 v0.2.1 (2019-12-01)

All custom exceptions raised publicly and used internally are now exposed via a new module, `wayback.exceptions`.

3.3 v0.2.0 (2019-11-26)

Initial release of this project. See v0.1 below for information about a separate project with the same name that has since been removed from PyPI.

3.4 v0.1

This version number is reserved because it was the last published release of a separate Python project also named `wayback` that has since been deleted from the Python Package Index and subsequently superseded by this one. That project, which focused on the Wayback Machine's timemap API, was maintained by Jeff Goettsch (username `jgoettsch` on the Python Package Index). Its source code is still available on BitBucket at https://bitbucket.org/jgoettsch/py-wayback/.

Index