PSAW

Release 0.0.12

Jul 09, 2022

Contents

CHAPTER 1

Installation

```
pip install psaw
```

CHAPTER 2

Description

A minimalist wrapper for searching public reddit comments/submissions via the pushshift.io API.

Pushshift is an extremely useful resource, but the API is poorly documented. As such, this API wrapper is currently designed to make it easy to pass pretty much any search parameter the user wants to try.

Although it is not necessarily reflective of the current status of the API, you should attempt to familiarize yourself with the Pushshift API documentation to better understand what search arguments are likely to work.

- API Documentation on github
- Endpoints and parameter descriptions
- /r/pushshift

CHAPTER 3

Features

- Handles rate limiting and exponential backoff subject to maximum retries and maximum backoff limits. A minimum rate limit of 1 request per second is used as a default per consultation with Pushshift's maintainer, /u/Stuck_in_the_matrix.
- Handles paging of results. Returns all historical results for a given query by default.
- Optionally handles incorporation of `praw` to fetch objects after getting ids from pushshift
- If not using `praw`, returns results in `comment` and `submission` objects whose API is similar to the corresponding `praw` objects. Additionally, result objects have an additional `.d_` attribute that offers dict access to the associated data attributes.
- Optionally adds a `created` attribute which converts a comment/submission's `created_utc` timestamp to the user's local time. (may raise exceptions for users with certain timezone settings).
- Simple interface to pass query arguments to the API. The API is sparsely documented, so it's often fruitful to just try an argument and see if it works.
- A `stop_condition` argument to make it simple to stop yielding results given arbitrary user-defined criteria
- Commandline interface (CLI) for simplified usage outside of python environment.

CHAPTER 4

WARNINGS

- Using non-default sort may result in unexpected behavior.
- Default behavior is to continuously hit the pushshift api. If a query is taking longer than expected to return results, it's possible that psaw is pulling more data than you may want or is caught in some kind of loop.
- I strongly recommend prototyping queries by printing to stdout to ensure you're getting the desired behavior.

CHAPTER 5

Demo usage (python)

```
from psaw import PushshiftAPI

api = PushshiftAPI()
```

Or to use pushshift search to fetch ids and then use praw to fetch objects:

```
import praw
from psaw import PushshiftAPI

r = praw.Reddit(...)
api = PushshiftAPI(r)
```

5.1 100 most recent submissions

```
# The `search_comments` and `search_submissions` methods return generator objects
gen = api.search_submissions(limit=100)
results = list(gen)
```

5.2 First 10 submissions to /r/politics in 2017, filtering results to url/author/title/subreddit fields.

The `created_utc` field will be added automatically (it's used for paging).

```
import datetime as dt

start_epoch=int(dt.datetime(2017, 1, 1).timestamp())

list(api.search_submissions(after=start_epoch,
```

(continues on next page)

(continued from previous page)

```
                                subreddit='politics',
                                filter=['url','author', 'title', 'subreddit'],
                                limit=10))
```

5.3 Trying a search argument that doesn't actually work

According to the pushshift.io API documentation, we should be able to search submissions by url, but (at the time of this writing) this doesn't actually work in practice. The API should still respect the `limit` argument and possibly other supported arguments, but no guarantees. If you find that an argument you have passed is not supported by the API, best thing is to just remove it from the query and modify your api call to only utilize supported arguments to mitigate risks from of unexpected behavior.

```
url = 'http://www.politico.com/story/2017/02/mike-flynn-russia-ties-investigation-
↪235272'
url_results = list(api.search_submissions(url=url, limit=500))

len(url_results), any(r.url == url for r in url_results)
# 500, False
```

5.4 All AskReddit comments containing the text "OP"

Use the `q` parameter to search text. Omitting the `limit` parameter does a full historical search. Requests are performed in batches of size specified by the `max_results_per_request` parameter (default=500). Omitting the "max_reponse_cache" test in the demo below will return all results. Otherwise, this demo will perform two API requests returning 500 comments each. Alternatively, the generator can be queried for additional results.

```
gen = api.search_comments(q='OP', subreddit='askreddit')

max_response_cache = 1000
cache = []

for c in gen:
    cache.append(c)

    # Omit this test to actually return all results. Wouldn't recommend it though:␣
↪could take a while, but you do you.
    if len(cache) >= max_response_cache:
        break

# If you really want to: pick up where we left off to get the rest of the results.
if False:
    for c in gen:
        cache.append(c)
```

5.5 Using the `aggs` argument to summarize search results

When an aggs parameter is provided to a search method, the first result yielded by the generator will contain the aggs result.

```
api = PushshiftAPI()
gen = api.search_comments(author='nasa', aggs='subreddit')
next(gen)
#  {'subreddit': [
#    {'doc_count': 300, 'key': 'IAmA'},
#    {'doc_count': 6, 'key': 'space'},
#    {'doc_count': 1, 'key': 'ExposurePorn'},
#    {'doc_count': 1, 'key': 'Mars'},
#    {'doc_count': 1, 'key': 'OldSchoolCool'},
#    {'doc_count': 1, 'key': 'news'},
#    {'doc_count': 1, 'key': 'pics'},
#    {'doc_count': 1, 'key': 'reddit.com'}]}
len(list(gen)) # 312
```

5.6 Using the `redditor_subreddit_activity` convenience method

If you want to profile a redditors activity as in the `aggs` example, the `redditor_subreddit_activity` provides a simple shorthand for profiling a user by the subreddits in which they are active, counting comments and submissions separately in a single call, and returning Counter objects for commenting and posting activity, respectively.

```
api = PushshiftAPI()
result = api.redditor_subreddit_activity('nasa')
result
#{'comment':
#   Counter({
#       'ExposurePorn': 1,
#       'IAmA': 300,
#       'Mars': 1,
#       'OldSchoolCool': 1,
#       'news': 1,
#       'pics': 1,
#       'reddit.com': 1,
#       'space': 6}),
# 'submission':
#   Counter({
#       'IAmA': 3,
#       'ISS': 1,
#       'Mars': 1,
#       'space': 3,
#       'u_nasa': 86})}
```

5.7 Using the `stop_condition` argument to get the most recent submission by a bot account

```
gen = api.search_submissions(stop_condition=lambda x: 'bot' in x.author)

for subm in gen:
    pass
```

(continues on next page)

(continued from previous page)

```
print(subm.author)
```

5.8 Collecting results in a `pandas.DataFrame` for analysis

```
import pandas as pd

df = pd.Dataframe([thing.d_ for thing in gen])
```

5.9 Logging

If you would like to peek under the hood (e.g. to see the URL of the GET request to the API), just turn on logging like so:

```
import logging

handler = logging.StreamHandler()
handler.setLevel(logging.INFO)

logger = logging.getLogger('psaw')
logger.setLevel(logging.INFO)
logger.addHandler(handler)
```

For more detailed messages, set the logging level to *DEBUG* instead of *INFO*.

5.10 Special Convenience Attributes

Consider the following simple query:

```
gen = api.search_submissions(subreddit='pushshift')
thing = next(gen)
```

Special attributes:

- `thing.d_` a dict containing all of the data attributes attached to the thing (which otherwise would be accessed via dot notation). One specific convenience this enables is simplifying pushing results into a pandas dataframe (above).
- `api.metadata_` The metadata data provided by pushshift (if any) from the most recent successful request. The most useful metadata attributes, IMHO, are:
  - `api.metadata_.get('shards')` - For checking if any shards are down, which can impact the result cardinality.
  - `api.metadata_.get('total_results')` - The database-side count of how many total items were found in the query and should be returned after paging through all results. Users have encountered rare edge cases that don't return all expected results, probably due to more than 500 items sharing the same timestamp in a result range. See issue #47 for progress resolving this behavior.

CHAPTER 6

Demo usage (CLI)

For CLI documentation, run

```
psaw --help
```

CHAPTER 7

License

PSAW's source is provided under the Simplified BSD License.

CHAPTER 8

Indices and tables

- genindex
- modindex
- search