Best Practices for Querying Twitter API using (forked) tweepy

Published on April 28, 2013.

Over the past few weeks I have been accessing Twitter API from python using tweepy. In the course of doing so, I added support in tweepy for using multiple authentication handlers, monitoring rate limits, waiting when running out of calls, support for search using Twitter API v1.1, proper pagination of results and more. In this blog post I will describe the major changes in my forked version of the tweepy package and provide some best practices and examples for querying the Twitter API robustly. Fork me on GitHub or follow my pull request on the main tweepy repo #282 to get the complete code referenced in this post.

New Features

- **Support for multiple authentication handlers**: sometimes you just need more than a single authentication handler. You can provide a list of authentication handlers to tweepy and manually switch between them:

```
import tweepy

# create authentication handlers given pre-existing keys
auths = []
for consumer_key, consumer_secret, access_key, access_secret in oauth_keys:
    auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
    auth.set_access_token(access_key, access_secret)
    auths.append(auth)

api = tweepy.API(auths)

# switch to the second authentication handler (zero-based index)
api.auth_idx = 1
```

- **Monitoring rate limit**: switching authentication handlers manually is nice, but it is far more useful to have tweepy do that automatically based on the number of remaining api calls. The monitor_rate_limit argument does exactly that - monitor to number of remaining calls to the Twitter API endpoint and switch to the next authentication handler when running out of calls. This feature lets you be a good citizen and not supersede your API calls quota.

```
api = tweepy.API(auths, monitor_rate_limit=True)
```

- **Blocking when running out of calls**: using the argument wait_on_rate_limit you can have tweepy wait when running out of calls. By monitoring the http headers returned from Twitter, tweepy now knows exactly how long to wait until the next lump of api calls is available. This is useful when you need more than the number of calls allowed by the rate limit, but don't want your program to fail with an exception or lose track of the current position in your results.

```
api = tweepy.API(auths,
            monitor_rate_limit=True, wait_on_rate_limit=True)
```

- **Proper iteration of result pages (pagination)**: prior to my forked version of tweepy, if you used the Cursor object to iterate over results you probably have missed some tweets in your result set. A good explanation of why this has happened is in Twitter own documentation on working with timelines. Now iteration uses max_id parameter to properly traverse results when using the cursor object. If you're wondering what's happening behind the scenes, take a look at MaxIdIterator class in tweepy/cursor.py.
- **Search endpoint now supports Twitter API v1.1**.
- **Unit testing for all of the above**.

Best Practices

- **Retrying on errors**: little know fact, unless you don't dig into the tweepy code, is that you can have tweepy retry making calls for you when certain error codes are returned. The retry feature is useful for handling temporary failures. You specify how many retries will be performed, delay in seconds between calls and the error response codes upon which tweepy will retry making calls. For a complete list of error response codes, see the Twitter API documentation on Error Codes & Responses. Here is an example of using the retry feature:

```
api = tweepy.API(auths, retry_count=3, retry_delay=5,
                 retry_errors=set([401, 404, 500, 503]))
```

- **How to use pagination with search?** here is the example from examples/paginated_search.py:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API([auth],
        # support for multiple authentication handlers
        # retry 3 times with 5 seconds delay when getting these error codes
        # For more details see
        # https://dev.twitter.com/docs/error-codes-responses
        retry_count=3,retry_delay=5,retry_errors=set([401, 404, 500, 503]),
        # monitor remaining calls and block until replenished
        monitor_rate_limit=True, wait_on_rate_limit=True
)

query = 'cupcake OR donut'
page_count = 0
for tweets in tweepy.Cursor(api.search, q=query, count=100,
                result_type="recent", include_entities=True).pages():
page_count += 1
# print just the first tweet out of every page of 100 tweets
print tweets[0].text.encode('utf-8')
# stop after retrieving 200 pages
if page_count >= 200:
    break
```

- **Response as python dictionary**: sometimes you don't want tweepy to spend time on parsing json into dedicated python object and have the results simply as a dictionary. The example at examples/json_results.py demonstrate querying a user timeline, tim oreilly in this case, using a json payload type:

```
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)
```

```
api = tweepy.API([auth])
json_user_timeline = tweepy.binder.bind_api(
    path = '/statuses/user_timeline.json',
    payload_type = 'json', payload_list = True,
    allowed_param = ['id', 'user_id', 'screen_name', 'since_id',
                     'max_id', 'count', 'page', 'include_rts'])

# iterate over 50 of tim oreilly's tweets
for tweet in tweepy.Cursor(json_user_timeline, api,
screen_name='timoreilly', count=20, include_rts=True).items(50):
print tweet['text'].encode('utf-8')
```

Have a comment? an idea of how to do things more elegantly? leave a comment or contact me on twitter @grinbergnir.

Share!

Comments

7 Comments **Three.Fourteen** 1 **Login**

Recommend **Share** **Sort by Best**

Join the discussion…

Alexandru Stanciu • 2 years ago
Heh, sorry I didn't find this post before..
I did pretty much the same thing in my fork here https://github.com/svven/tweep... - have a pull request as well but no replies - https://github.com/tweepy/twee....
Reply • Share ›

TM ➦ Alexandru Stanciu • 2 years ago
Throws an error when I try to install your fork
Reply • Share ›

Alexandru Stanciu ➦ TM • 2 years ago
Did you try this?
`pip install git+https://github.com/svven/tweep...`
What error exactly?
Reply • Share ›

Guest • 2 years ago
thanks for the post buddy!
Reply • Share ›

djepssp • 2 years ago
Hi nirg, nice blogpost! I'm wondering if you don't get blocked (blacklisted) by having too many authenticated apps/tokens within the same IP/host.

Also, what are your remarks if you have to retrieve all tweets from more than one million users?

Thanks!
Reply • Share ›

grinbergnir Mod ➦ djepssp • 2 years ago
Hi **@djepssp**, I didn't get blacklisted by IP, but I'm pretty sure that if you try to get all tweets for million users you will be blacklisted. I'd suggest you use the streaming endpoint for that purpose.
Reply • Share ›

djepssp ➦ grinbergnir • 2 years ago
Well, that is not an option I need the past, at least to the 3200 tweets limit. To be more specific I don't need all tweets from all followers of this particular account, I only need all tweets from all followers who has geo_enabled = True and other simple conditions.
Reply • Share ›

Subscribe **Add Disqus to your site Add Disqus Add** **Privacy**

Latest posts

Jan 10

Changes in Engagement Before and After Posting to Facebook
Our recent work got accepted to CHI 2016! See the publications page for the full paper.
I think this work is interesting for three main reasons: the questions we addressed, the methodology for answering them, and the answers we arrived at.
The fundamental question we addressed in this work is ...

Feb 28

WSDM 2014 Summary, Opinions and Other Thoughts

About the author

I am a PhD candidate in the Computer Sc
Cornell University and part of the Jacobs
Tech. My advisor is Mor Naaman and I am

It was a great attending WSDM this year, right at the heart of NYC! I could not possibly cover all that went down there in the three days of the conference, but would use this post to highlight sessions and talks that I attended and found particularly interesting.
The conference ...

May
19

Our Semantic Keyword Expansion API is ON!

If you ever wanted to track a certain topic on Twitter, you probably wondered what keywords people are using to refer to it. Our new Semantic Expansion API lets you do exactly that - given a short list of keywords, it finds keywords that appear in a similar context or directly ...

Apr
28

Best Practices for Querying Twitter API using (forked) tweepy

Over the past few weeks I have been accessing Twitter API from python using tweepy. In the course of doing so, I added support in tweepy for using multiple authentication handlers, monitoring rate limits, waiting when running out of calls, support for search using Twitter API v1.1, proper pagination ...

Mar
30

Extracting Diurnal Patterns of Real World Activity from Social Media

Our most recent work got accepted to ICWSM 2013! See the publications page for the full paper.
Here is the abstract:
In this study, we develop methods to identify verbal expressions in social media streams that refer to real-world activities. Using aggregate daily patterns of Foursquare checkins, our methods extract ...

great Social Technologies Lab. I was fortu
past summers as a research intern ...

Find out more

Contact

Email
nir -AT- cs.cornell.edu

Address
111 8th Ave. Suite 1202, New York, NY 100

Blogroll

CS @ Cornell
Social Technologies Lab @ Cornell Tech
Facebook Core Data Science