TwitterAds Documentation

Release 0.0.0

Jacob Gillespie

March 15, 2016

Contents

Welcome to TwitterAds; a Python wrapper around the Twitter Ads API.

The Twitter Ads API exposes a bunch of useful resources but interacting with each resource demands a certain familiarity with the API and the Twitter advertising platform. This package aims to ease these requirements where possible.

CHAPTER 1

Getting started

Getting up and running is extremely easy. In short,

- Clone the repository
- Replace the credentials in *config.py* with your own
- Get some data

```
>>> import twitter
>>> twitter.get_accounts()
<Twitter Response [OK]>
```

A more detailed guide follows:

1.1 Usage guide

Here, we cover in some detail how to use the various components of this library. It's split into three sections:

1.1.1 Installation

First, let's install the library. The required steps are as follows:

- Clone the repository.
- Install the required modules with

```
pip install -r requirements.txt
```

- Replace the dictionary keys in `config.py` with your credentials. See this page for more information on obtaining these credentials.
- With the project directory in your `PYTHONPATH` variable, open up a console and run

```
>>> import twitter
>>> twitter.get_accounts()
<Twitter Response [OK]>
```

If you see the above output, you're all set. Head here to start using the API.

Common issues

If, instead of the above, you see

```
>>> twitter.get_accounts()
<Twitter Response [Incomplete]>
```

then you are likely unauthorized. For further detail on the issue, run

```
>>> accounts = twitter.get_accounts()
>>> accounts.errors
[{'errors': [{'message': 'This request is not properly authenticated', 'code': 'UNAUTHORIZED_ACCESS'
```

1.1.2 Using the API

Let's call some of the functions exposed by the API module.

Accounts

First, let's get all accounts

```
>>> accounts = twitter.get_accounts()
>>> accounts
<Twitter Response [OK]>
```

An `[OK]` response implies that we retrived all accounts successfully. We can see this is true by checking the `errors` attribute of `accounts`

```
>>> accounts.errors
[]
```

You'll notice that if `get_accounts` is unsuccessful, the response object looks different. Let's force an unsuccessful response by passing something other than a boolean in as `with_deleted`

```
>>> accounts = twitter.get_accounts(with_deleted='ops')
>>> accounts
<Twitter Response [Incomplete]>
```

Notice the difference? Checking `accounts.errors` reveals that Twitter was expecting us to pass a boolean to `with_deleted`

```
>>> accounts.errors
[{'errors': [{'parameter': 'with_deleted', 'message': 'Expected Boolean, got "ops" for with_deleted',
```

Campaigns

Of course, we can access other resources too.

```
>>> campaigns = twitter.get_campaigns()
>>> campaigns
<Twitter Response [OK]>
```

That likely took a few seconds to complete. This is beacuse the `get_campaigns` function has to make many individual requests to the Twitter Ads API. Specifically, the API demands do separate requests per account and that we request at most 1000 campaigns at once.

To speed things up, we can request campaigns for specific accounts

```
>>> accounts = twitter.get_accounts()
>>> account_ids = [account['id'] for account in accounts.data[:3]]  # Get the first three account IDs
>>> campaigns = twitter.get_campaigns(account_ids=account_ids)
>>> campaigns
<Twitter Response [OK]>
```

Note that we can choose whether to get deleted campaigns or not too.

```
>>> len(campaigns.data)
2742

>>> live_campaigns = twitter.get_campaigns(account_ids=account_ids, with_deleted=False)
>>> len(live_campaigns.data)
2495
```

The `with_deleted` parameter applies at the level of the resource being requested. For higher-level entities, we'll always get all items. For example, in `get_campaigns`, we get campaigns whose status reflects the value of `with_deleted` but we consider all accounts (recall that in order to get campaigns, we must specify the account), deleted or otherwise.

Line Items

We have a number of other entities available to us. One of these is line items and is called in exactly the same way as campaigns.

```
>>> twitter.get_line_items(account_ids=account_ids)
<Twitter Response [OK]>
```

Promoted tweets

Another of these entities is promoted tweets. Again, it is called in the same way as campaigns

```
>>> twitter.get_promoted_tweets(account_ids=account_ids)
<Twitter Response [OK]>
```

Stats

The last available function is `get_stats`. A word of warning before using this function; as with the other requests listed here, this function sends multiple requests to the Twitter Ads API. Comparatively though, this sends significantly more [1]. Further, Twitter have put rate limiting in place against this resource so it's necessary to sleep at times. As such, it's definitely worth specifying `account_ids` initially and easing this constraint as you grow more comfortable with the function.

```
>>> account_ids = [account['id'] for account in accounts.data[:1]]
>>> twitter.get_stats(account_ids=account_ids)
<Twitter Response [OK]>
```

A final note is that stats are pulled at promoted tweet level only. In the future, you'll have more control over this.

That covers all functions available to you in the API. If you're more familiar with the Twitter Ads API and would like further tailor your requests, head here to see how.

[1] The Twitter Ads API demands that you pull stats against at most 20 promoted tweets at a time.

1.1.3 Building your own requests

CHAPTER 2

Acknowledgements

This library wouldn't exist without **requests** whose model is the source of inspiration for the one you see here. Not only is the model a good one, the library is so wonderfully written that understanding it is simply a case of reading it.

E

P