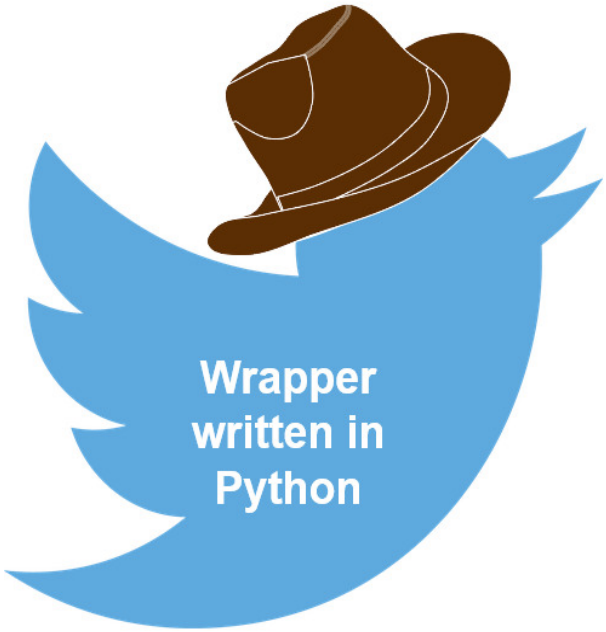
twipper

Release 0.1.6

Jan 13, 2020

Contents:


Wrapper
written in
Python

CHAPTER 1

Introduction

twipper is an acronym that stands for Twitter wrapper; so the package is made in order to cover Twitter API endpoints defined as Python functions for both Free and Premium plans, which they both include batch and stream processing functions.

CHAPTER 2

Installation

In order to get this package working you will need to install it using pip by typing on the terminal:

```
$ python -m pip install twipper --upgrade
```

Or just install the current release or a specific release version such as:

```
$ python -m pip install twipper==0.1.6
```

CHAPTER 3

Batch

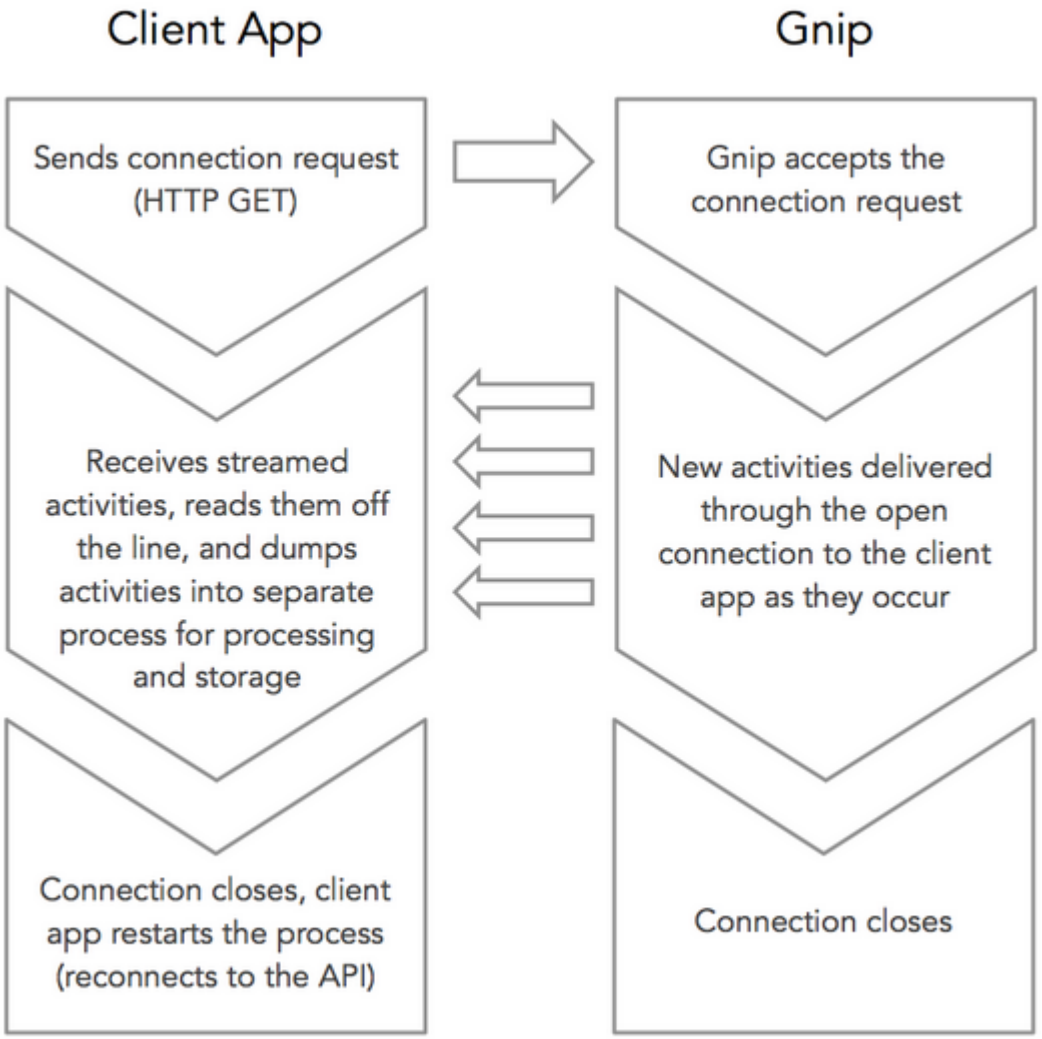
Under development.

CHAPTER 4

Streaming

Twitter offers a Streaming service to retrieve real-time tweets that match a given query or location. This service uses an OAuth1 authentication method which requires both consumer (`consumer_key`, `consumer_secret`) and access keys (`access_token`, `access_token_secret`) and those keys will be used to authenticate the Twitter App, and so on to access the Streaming service via OAuth1.

As described below, for further details on Twitter Streaming insights please check the following diagram proposed by Twitter on Consuming Streaming Data.



So on, twipper has been created in order to cover Twitter Streaming process to retrieve real-time tweets. The sample usage of **twipper** when it comes to streaming data will be as it follows:

```
import twipper

from twipper.streaming import stream_tweets

cred = twipper.Twipper(consumer_key='consumer_key',
                       consumer_secret='consumer_secret',
                       access_token='access_token',
                       access_token_secret='access_token_secret')

tweets = stream_tweets(access=cred,
                       query='cats',
                       language='en',
                       filter_retweets=True,
                       tweet_limit=100,
                       date_limit=None,
                       retry='no_limit')
```

(continues on next page)

(continued from previous page)

```
results = list()

for index, tweet in enumerate(tweets, 1):
    print('Inserting tweet number ' + str(index))
    results.append(tweet)
```

The previous block of code retrieves up to 100 tweets written in English matching the word *cats* as specified on the query, filtering out retweets and with an infinite number of tries if the retrieval process fails, until the objective is reached. Later on, as the returned data from the function are `yield` values inserted into a generator, a for loop is needed in order to store all the available retrieved data on a list. That list will contain a tweet formatted as a JSON object as described by Twitter.

```
{
    "created_at": "Thu May 10 15:24:15 +0000 2018",
    "id_str": "850006245121695744",
    "text": "Here is the Tweet message.",
    "user": {

    },
    "place": {

    },
    "entities": {

    },
    "extended_entities": {

    }
}
```

More JSON samples can be found at https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json.html.

Note: For further `twipper.streaming` insights or information please use the streaming API Reference where functions are described and sorted out so to understand its usage and how the params should be formatted in order to execute useful queries and retrieve the desired information from Twitter.

CHAPTER 5

Premium

Under development.

CHAPTER 6

Queries

Since the Twitter querying system is a little bit confusing due to some differences between Standard and Streaming queries, some additional functions have been created by twipper in order to fix this minor issue when it comes to data retrieval from Twitter via its API, using either **twipper** or any other Twitter Wrapper for Python.

So on, a simple keyword based queries format has been created in order to convert every traditional query using logical operators to the standard query required by Twitter on its Standard or Streaming search. The created queries have the following operators:

- **AND**: logical operator that implies that the tweets-to-search match both of the introduced keywords, but not on the specified order.
- **OR**: logical operator that means that the tweets-to-search will match every tweet containing either one word or another, can be applied for N keywords.
- **NOT**: operator that means that the tweets-to-search do not contain the introduced keyword.

> **Warning:** Note that the NOT operator will only work for standard queries since it is not supported by Twitter Streaming

Once the basics have been explained a simple example on its usage is proposed. Suppose that you want to search tweets containing either the word dog or cat, and you want to filter out the ones that even matching the previous condition also have the word frog. So on the query will look like: `DOG AND CAT AND NOT frog`, then, the twipper querying formatter will convert that query to either a standard or a streaming query.

Then, twipper should be used the following way in order to convert the previous query into a standard twitter query:

```
from twipper.utils import standard_query

query = "dog OR cat AND NOT frog"
standard = standard_query(query)

print(standard)
>>> "dog OR cat -frog"
```

As already said before, the NOT operator just works for the standard search as it is not supported for streaming tweets, if we wanted to filter out tweets containing certain words we should do it while the tweets are being retrieved. Anyways the basic usage for streaming queries should be like:

```
from twipper.utils import streaming_query

query = "dog OR cat"
streaming = streaming_query()

print(streaming)
>>> "dog,cat"
```

For further help you can either check the API Reference.

CHAPTER 7

Usage

As **twipper** is a Twitter Wrapper written in Python its main purpose is to wrap all the available endpoints listed on the Twitter API for both versions (Free and Premium), so to use them from a simple function call. So on the main step is to validate Twitter API credentials since they are mandatory in order to work with the Twitter API.

```
import twipper

cred = twipper.Twipper(consumer_key='consumer_key',
                       consumer_secret='consumer_secret',
                       access_token='access_token',
                       access_token_secret='access_token_secret')
```

Now once the `Twipper` credentials object has been properly created we can use it in order to work with the Twitter API using Python. In the case that we want to retrieve data from Twitter based on a query, e.g. we want to search *cat* tweets to analyze its content to launch a cat campaign for our brand (random example because everybody loves cats).

```
from twipper.batch import search_tweets

tweets = search_tweets(access=cred,
                       query='cats',
                       page_count=1,
                       filter_retweets=True,
                       verified_account=False,
                       language='en',
                       result_type='popular',
                       count=10)
```

So on, using `batch` functions you can retrieve historical *tweets* from the last 7-30 days matching the introduced query, in this case the query is *cats* due to our cat campaign, remember it. Anyways, params can be adjusted to our desires and/or needs as described on the API Reference.

Note: For further **twipper** functions insights check the API Reference.

CHAPTER 8

API Reference

8.1 twipper.batch

twipper.batch.**search_tweets**(*access*, *query*, *page_count=1*, *filter_retweets=False*, *verified_account=False*, *language=None*, *result_type='mixed'*, *count=100*)

This function retrieves historical tweets on batch processing. These tweets contain the specified words on the query, which can use operators such as AND or OR, as specified on https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/standard-operators. Additionally some extra arguments can be specified in order to particularize the tweet search, so on to retrieve the exact content the user is looking for. API Reference: https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html

Parameters

- **access** (twipper.credentials.Twipper) – object containing all the credentials needed to access api.twitter
- **query** (str) – contains the query with the words to search along Twitter historic data.
- **page_count** (int, optional) – specifies the amount of pages (100 tweets per page) to retrieve data from, default value is 1.
- **filter_retweets** (boolean, optional) – can be either *True* or *False*, to filter out retweets or not, respectively, default value is *False*.
- **verified_account** (boolean, optional) – can either be True or False to retrieve tweets just from verified accounts or from any account type, respectively.
- **language** (str, optional) – is the language on which the tweet has been written, default value is *None*.
- **result_type** (str, optional) – value to indicate which type of tweets want to be retrieved, it can either be *mixed*, *popular* or *recent*
- **count** (int, optional) – number of tweets per requests to retrieve (default and max is 100).

Returns Returns a `list` containing all the retrieved tweets from Twitter API, based on the search query previously specified on the function arguments.

Return type `list` - tweets

Raises `ValueError` – raised if the introduced arguments do not match or errored.

`twipper.batch.`**`search_user_tweets`**(*access*, *screen_name*, *page_count=1*, *filter_retweets=False*, *language=None*, *result_type='mixed'*, *count=100*)

This function retrieves historical tweets from a Twitter user by their screen_name (@), whenever they grant the application access their tweets for commercial purposes on ReadOnly permission. Retrieved tweets are stored on a `list` which will be returned to the user. API Reference: https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets.html

Parameters

- **`access`** (`twipper.credentials.Twipper`) – object containing all the credentials needed to access api.twitter
- **`screen_name`** (`str`) – contains the username of the user from which tweets are going to be retrieved.
- **`page_count`** (`int`, optional) – specifies the amount of pages (100 tweets per page) to retrieve data from, default value is 1.
- **`filter_retweets`** (`boolean`, optional) – can be either *True* or *False*, to filter out retweets or not, respectively, default value is *False*.
- **`language`** (`str`, optional) – is the language on which the tweet has been written, default value is *None*.
- **`result_type`** (`str`, optional) – value to indicate which type of tweets want to be retrieved, it can either be *mixed*, *popular* or *recent*
- **`count`** (`int`, optional) – number of tweets per requests to retrieve (default and max is 100).

Returns Returns a *list* containing all the retrieved tweets from Twitter, which means all the available tweets from the user specified on the arguments of the function.

Return type `list` - tweets

Raises `ValueError` – raised if the introduced arguments do not match or errored.

8.2 `twipper.streaming`

`twipper.streaming.`**`stream_country_tweets`**(*access*, *country*, *language=None*, *filter_retweets=False*, *tweet_limit=None*, *date_limit=None*, *retry=5*)

This function retrieves streaming tweets matching the given query, so on, this function will open a stream to the Twitter Streaming API to retrieve real-time tweets. By the time these tweets are retrieved, they are handled and returned as a `list`. API Reference: https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters.html

Parameters

- **`access`** (`twipper.credentials.Twipper`) – object containing all the credentials needed to access api.twitter
- **`country`** (`str`) – contains the country name from where generic tweets will be retrieved.
- **`language`** (`str`, optional) – is the language on which the tweet has been written, default is *None*.

- **filter_retweets** (boolean, optional) – can be either *True* or *False*, to filter out retweets or not, respectively.
- **tweet_limit** (int, optional) – specifies the amount of tweets to be retrieved on streaming, default is 10k tweets.
- **date_limit** (str, optional) – specifies the date (format *yyyymmddhhmm*) where the stream will stop, default is *None*
- **retry** (int or str, optional) – value to set the number of retries if connection to api.twitter fails, it can either be an int or a str which can just be the value *no_limit* in the case that no retry limits want to be set. Default value is 5 retries whenever connection fails, until function finishes.

Returns Yields a list containing all the retrieved tweets from Twitter, which means all the available tweets from the user specified on the arguments of the function.

Return type list - tweets

Raises ValueError – raised if the introduced arguments do not match or errored.

twipper.streaming.**stream_tweets**(*access*, *query*, *language=None*, *filter_retweets=False*, *tweet_limit=None*, *date_limit=None*, *retry=5*)

This function retrieves streaming tweets matching the given query, so on, this function will open a stream to the Twitter Streaming API to retrieve real-time tweets. By the time these tweets are retrieved, they are handled and returned as a list. API Reference: https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/basic-stream-parameters.html

Parameters

- **access** (twipper.credentials.Twipper) – object containing all the credentials needed to access api.twitter
- **query** (str) – contains the query with the words to search along the Twitter Streaming.
- **language** (str, optional) – is the language on which the tweet has been written, default is *None*.
- **filter_retweets** (boolean, optional) – can be either *True* or *False*, to filter out retweets or not, respectively.
- **tweet_limit** (int, optional) – specifies the amount of tweets to be retrieved on streaming, default is 10k tweets.
- **date_limit** (str, optional) – specifies the date (format *yyyymmddhhmm*) where the stream will stop, default is *None*
- **retry** (int or str, optional) – value to set the number of retries if connection to api.twitter fails, it can either be an int or a str which can just be the value *no_limit* in the case that no retry limits want to be set. Default value is 5 retries whenever connection fails, until function finishes.

Returns Yields a list containing all the retrieved tweets from Twitter, which means all the available tweets from the user specified on the arguments of the function.

Return type list - tweets

Raises ValueError – raised if the introduced arguments do not match or errored.

8.3 twipper.premium

twipper.premium.**search_tweets**(*access*, *query*, *page_count*, *from_date*, *to_date*, *language=None*, *filter_retweets=False*)

This function retrieves historical tweets on batch processing from Twitter's Full Archive or 30Day. These tweets contain the specified words on the query, which can use premium operators as specified on https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/premium-operators. Additionally, some extra arguments can be specified in order to particularize the tweet search, so on to retrieve the exact content we are looking for. Retrieved tweets are stored on a `list` which will lated be returned once it is filled with the retrieved tweets.

Parameters

- **access** (`twipper.credentials.Twipper`) – object containing all the credentials needed to access api.twitter
- **query** (`str`) – contains the query with the words to search along Twitter historic data.
- **page_count** (`int`) – specifies the amount of pages (100 tweets per page) to retrieve data from.
- **from_date** (`str`) – starting date of the time interval to retrieve tweets from (*yyyymmddhhmm* format)
- **to_date** (`str`) – end date of the time interval to retrieve tweets from (*yyyymmddhhmm* format)
- **language** (`str`) – is the language on which the tweet has been written.
- **filter_retweets** (`boolean`, optional) – can be either *True* or *False*, to filter out retweets or not, respectively.

Returns

description Returns a `list` containing all the retrieved tweets from Twitter, which means all the available tweets from the user specified on the arguments of the function.

Return type tweets (`list`)

Raises `ValueError` – raised if the introduced arguments do not match or errored.

twipper.premium.**search_user_tweets**(*access*, *screen_name*, *page_count*, *from_date*, *to_date*, *language=None*, *filter_retweets=False*)

This function retrieves historical tweets on batch processing from Twitter's Full Archive or 30Day from a specific user via its screen name (Twitter name). These tweets contain the specified words on the query, which can use premium operators as specified on https://developer.twitter.com/en/docs/tweets/rules-and-filtering/overview/premium-operators. Additionally, some extra arguments can be specified in order to particularize the tweet search, so on to retrieve the exact content we are looking for. Retrieved tweets are stored on a `list` which will lated be returned once it is filled with the retrieved tweets.

Parameters

- **access** (`twipper.credentials.Twipper`) – object containing all the credentials needed to access api.twitter
- **screen_name** (`str`) – is the Twitter's public name of the account that tweets are going to be retrieved, note that the account must be public.
- **page_count** (`int`) – specifies the amount of pages (100 tweets per page) to retrieve data from.
- **from_date** (`str`) – starting date of the time interval to retrieve tweets from (*yyyymmddhhmm* format)

- **to_date** (str) – end date of the time interval to retrieve tweets from (*yyyymmddhhmm* format)
- **language** (str) – is the language on which the tweet has been written.
- **filter_retweets** (boolean, optional) – can be either *True* or *False*, to filter out retweets or not, respectively.

Returns

description Returns a list containing all the retrieved tweets from Twitter, which means all the available tweets from the user specified on the arguments of the function.

Return type tweets (list)

Raises ValueError – raised if the introduced arguments do not match or errored.

CHAPTER 9

Contribute

As this is an open source project it is open to contributions, bug reports, bug fixes, documentation improvements, enhancements and ideas.

Also there is an open tab of issues where anyone can contribute opening new issues if needed or navigate through them in order to solve them or contribute to its solving.

CHAPTER 10

Disclaimer

This package has been created in order to cover Premium Twipper API functions since tweepy, the most used Python package working as a Twitter API wrapper. Anyways, twipper also covers both Free and Premium functions, which include batch processing and stream processing.

Conclude that this is the result of a research project, so this package has been developed with research purposes and no profit is intended.

Note: For further information or any question feel free to contact me via email You can also check my Medium Publication where I post Data Science research content.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index

S

T