twitter-sentiment Documentation

Release stable

Nov 13, 2018

Contents

Release v0.0.1

twitter-sentiment is a light-weight Python 3 library that allows you to evaluate the sentiment of tweets. Behind the scene, twitter-sentiment classify tweets as either positive (1) or negative (0) and returns a ratio of positive tweets.

The library also includes features to get structured tweets, users, and user mentions to build and develop insights regarding users and their tweets.

```
import twitterSentiment

connection = twitterSentiment.API()
search = connection.querySearch("Los Angeles", count=1, result_type='recent', lang='en
↪')
data = twitterSentiment.StructureStatusesData(search)
sentiment = twitterSentiment.SentimentScore(data.getTweet())
print(sentiment.getSentimentClassification())

>> 0.6666
```

CHAPTER 1

Features

- Connect to the Twitter API in a matter of seconds
- Get ratio of positive tweets amongst the ones analyzed (ranges from 0 to 1)
- Get raw tweet response
- Get structured tweets, users, and user mentions data to leverage insights

CHAPTER 2

Install it now

```
pip install twitter-sentiment
```

CHAPTER 3

Table of Contents

3.1 License

3.2 Installation

3.2.1 Installing/Upgrading from pip

```
pip install --upgrade twitter-sentiment
```

Running this command will install twitter-sentiment and all the required libraries. You can find more info regarding pip here

3.2.2 Python Version

twitter-sentiment supports Python 3.6+

3.2.3 Dependencies

When installing twitter-sentiment, pip will automatically install:

- requests
- urllib3
- TextBlob

3.3 Getting Started

3.3.1 Getting Twitter Access Token

Before you can send requests to the Twitter API you will need to create an application and generate credentials. Credentials are composed of Consumer key, Access Token and Access Token Secret.

You can find more information on how to create an application here

3.3.2 Creating an Environment Variable

Once you have your credentials you will need to create two environment variables referencing your API Key and your API Secret Key. This is important as it 1) keep those two keys private and unexposed to whomever you will share your code with and 2) allows you to send requests to the Twitter API.

On Mac

Open your terminal and enter the following command

```
nano .bash_profile
```

Once your .bash_profile is open, create two environment variables by entering the following text. Make sure you do not include spaces between the "=" and your variable name and values.

```
export TWITTER_CLIENT_KEY="your_api_key"
export TWITTER_CLIENT_SECRET="your_api_secret_key"
```

On Windows

Current version has not been tested on Windows.

3.3.3 Creating an API Object

The first step is to create an *API()* object that will handle the connection to the Twitter API.

```
api = twitterSentiment.API()
```

3.3.4 Searching Tweets

Once you have instantiated an *API()* object you will be able to search for tweets by using the *searchQuery()* method.

```
tweet = api.searchQuery('Twitter', geocode=None,lang='en',result_type='recent' ,
↪count=1, until=None, since_id=None, max_id=None, include_entities=False, tweet_mode=
↪"extended", return_json=True)

print(tweet)

>> {'statuses': [{'created_at': 'Thu Sep 06 19:38:16 +0000 2018', 'id':␣
↪1037787081936973824, 'id_str': '1037787081936973824', 'full_text': ....
```

searchQuery() returns a dictionary with 1 key and 1 value. The value of the dictionary is a list with one dictionary per count of tweets returned.

3.3.5 Getting Structured Data

Once you have your tweet(s), you can create an object *StructureStatusesData()* to get the data in a structured way. *StructureStatusesData()* takes one argument, which is the value returned by the *searchQquery()* method. 4 methods can be used with *StructureStatusesData()*:

- getData()
- getTweet()
- getUser()
- getUserMentioned()

```
data = twitterSentiment.StructureStatusesData(tweet)
structured_tweets = data.getTweet()
print(data.getData())

>> ([{id': 856800998, 'name': 'Someone', 'screen_name': 'some_one', ....}])
```

The getData() method is the most general one. It returns a tuple of length 3 composed of list, each holding tweets, user, and user mentions data. Each list contains a dictionary per *count* of tweets defined in the searchQuery() method

3.3.6 Getting Sentiment Classification Ratio

With your tweets structured into a list, you can use the *getTweet()* class method to easily get the ratio of positively classified tweets in your returned values.

```
sentiment = twitterSentiment.SentimentScore(structured_tweets)
print(sentiment.getSentimentClassification())

>> 0.6666
```

3.4 API Reference

3.4.1 API Class

This class is directly imported from twitterSentiment. It handles the basic connection to the API as well as tweet searches.

```
from twitterSentiment import API
```

twitterSentiment.API() - ***API object*** the API() class manages the connection to the Twitter API. API() does not take any parameter, though, it is required to create a *TWITTER_CLIENT_KEY* and *TWITTER_CLIENT_SECRET* environment variable in your system to connect to your Twitter application. It is required to create an API object before calling any of the class methods.

getBearerToken() - ***string*** Returns a string object referencing the value of the bearer token. It is used in searchQuery() class method to authorize the request to the Twitter API.

searchQuery() - ***dictionnary***

It is the main class to send search query to the Twitter API. It returns a dictionnary object. The class has 11 parameters:

- **q** - a string. This is where the search keyword is passed [REQUIRED - optional if geocode is not specified]
- **geocode** - a string composed of 3 values separated by commas (lat, long, radius). Radius needs to be specified in either miles or kilometer by passing “mi” or “km” [OPTIONAL - required if q not specified]
- **lang** - a string representing the language filter for the tweets to return [OPTIONAL]
- **result_type** - a string. This parameter accepts only 2 values “mixed” or “recent” [REQUIRED]
- **count** - integer. The number of tweets to return per request. Default to 15 [REQUIRED]
- **until** - [OPTIONAL]
- **since_id** - [OPTIONAL]
- **max_id** - [OPTIONAL]
- **include_entities** - [OPTIONAL]
- **tweet_mode** - a string. Default to “extended” [REQUIRED]
- **return_json** - a string. Format of the response. Default to “JSON”. Changing this argument may yield unexpected results [REQUIRED]

client_key - ***string*** Will return the client key value saved in the environment variable.

client_secret - ***string*** Will return the client secret value saved in the environement variable.

base_url - ***string*** Will return the main root of the API URL.

token_url_extension - ***string*** Will return the API URL extension for the token authorization.

search_url_extension - ***string*** Will return the API URL extension used for the request (default to the standard API *1.1/search/tweets.json?*).

self.search_url - ***string*** Will return the full search URL used for the request.

self.params - ***list*** Will return the list of parameters to pass to the search_url.

3.4.2 StructureStatusesData Class

This class is directly imported from twitterSentiment. It is use to get a list of formated data with only specific data points - as opposed to the raw json response returned by *twitterSentiment.searchQuery()*.

```
form twitterSentiment import StructureStatusesData
```

twitterSentiment.StructureStatusesData() - ***StructureStatusesData object*** Takes one argument. The argument should be the value returned by the *searchQuery()* class method of *API()*. The raw response returned by the twitter API should also work - though unexpected behaviors could happen.

getData() - ***list*** Returns a list of length 3 comprised of sub list of length equals to the *count* argument value from *twitterSentiment.searchQuery()*.

getTweet() - ***list***

Returns a list of length ***count*** **with tweets values. The values returned by** ***getTweet()*** **are:**

- id
- created_at
- full_text
- geo
- coordinates
- place
- retweet_count
- favorite_count
- **entities**
  - hashtags
  - **user_mentiones**
    - id
- **metadata**
  - iso_language_code
- **user**
  - id

getUser() - ***list***

Returns a list of length ***count*** **with tweets values. The values returned by** ***getUser()*** **are:**

- **user**
  - id
  - name
  - screen_name
  - location
  - description
  - followers_count
  - friends_count

- listed_count
- favourites_count
- verified
- statuses_count
- lang

getUserMentioned() - ***list***

Returns a list of length *count* with tweets values. The values returned by *getUserMentioned()* are:

- tweet_id
- **entities**
  - **user_mentions**
    - id
    - name
    - screen_name

self.statuses - ***dictionnary*** Returns the raw response from the API request.

3.4.3 SentimentScore Class

This class is directly imported from twitterSentiment. It is used to classify a list of tweets returned by *twitterSentiment.StructureStatusesData().getTweet()*.

```
form twitterSentiment import StructureStatusesData
```

twitterSentiment.SentimentScore() - ***SentimentScore object*** Takes the value returned by *getTweet()* as an argument.

getSentimentClassification() - ***float*** Returns the ratio of tweets classified as positive by TextBlob *NaiveBayesAnalyzer()* model.

self.tweet_list - ***list*** Returns a list of tweets without any URLs. Cleaned up for sentiment analysis.

self.blobber - ***Blobber Class*** Returns a blobber class. Initialized to prevent retraining of model for each tweet analyzed.