Python Guideline For Twitter Data Collection

updated by Sep 4, 2015

In this assignment, you are expected to collect data from Twitter and analyze it. The data collection from Twitter is facilitated by Twitter APIs and this tutorial gives a brief overview of how to use it. Also, this tutorial uses python as the interface to invoke Twitter APIs.

1 Development Environment Setup

There exist many different python libraries that can be used to access Twitter APIs. However, we will use "**python-twitter**" mainly because it is well documented. Following is the walkthrough to setup the development environment in Linux.

1. Install pip, a package management library for Python.
   (a) $ sudo apt-get install python-pip python-dev build-essential
2. Install python-simplejson, python-oauth2, python-httplib2, python-setuptools
   (a) $ sudo apt-get install python-simplejson python-oauth2 python-httplib2 python-setuptools
3. Download latest python-twitter package from link
4. Extract the source distribution and run
   (a) $ sudo python setup.py build
   (b) $ sudo python setup.py install

If you are using Mac, you could use easy_install to install all dependency packages:

1. $ sudo easy_install simplejson oauth2 httplib2 setuptools
2. Download, extract python-twitter, build and install (same as step 3 and 4 as in Linux)

If you are using Windows, you may find this link useful for setting up a proper python environment on Windows machine.

Once, these steps are done, you can start writing applications that access Twitter APIs. Simple examples can found at Section 3 on 'using python-twitter'. But, before you do that, it is important that you understand the different categories of APIs and their usage. It is recommended to write your application in the extracted python-twitter package.

2 Twitter APIs

There are three twitter API categories and for this assignment you would only require **REST API** access.

1. **REST APIs:** The REST APIs provides programmatic access to read and write Twitter data. Author a new Tweet, read author profile and follower data, and more. The REST API identifies Twitter applications and users using OAuth; responses are available in JSON.

2. **Streaming APIs:** The Streaming APIs continuously deliver new responses to REST API queries over a long-lived HTTP connection. Receive updates on the latest Tweets matching a search query, stay in sync with user profile updates, and more. If your application is rate-limited for over-polling the REST APIs the Streaming APIs may be a good solution for your needs.

3. **Ads API:** The Ads API gives partners a way to integrate Twitter advertising management in their product. Selected partners have the ability create their own tools to manage Twitter Ad campaigns while easily integrating into existing, cross-channel advertising management solutions.

Permission (OAuth): The Twitter API only allows clients to make a limited number of requests in a given time, e.g. for Get requests, **15 calls or 180 calls every 15 minutes** can be made. Details can be found at Twitter API Rate Limits page. This policy affects the APIs in different ways. *Keep a check on the number of API calls made in 15 minutes. If it reaches the limit, the application should wait before making another API call. Or you can call an API GetRateLimitStatus() to check the remaining number of calls for each method within 15-min window.* Moreover, authentication is required for applications that uses Twitter API's to access user's information. Hence, it becomes important that you get authentication before you start accessing the REST APIs.

Apart from getting an authentication for your application, it is also very important that you design your application in such a way that you dont call same API again and again, instead use caching and always store the responses.

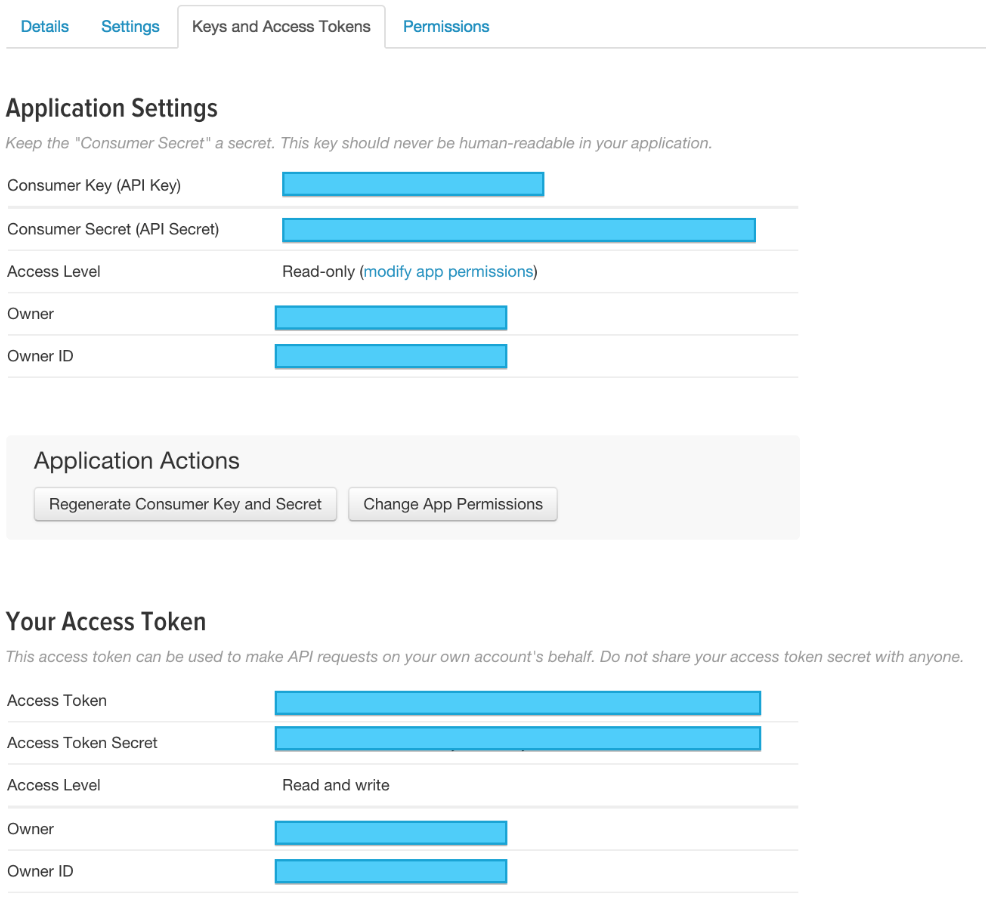
OAuth is currently the only way of getting an authentication for an application. There are two forms of authentication in the new model:

1. **Application-user authentication:** This is the most common form of resource authentication in Twitters OAuth 1.0A implementation to date. Your signed request both identifies your applications identity in addition to the identity accompanying granted permissions of the end-user youre making API calls on behalf of, represented by the users access token.

   In order to get OAuth authentication for an application, developers have to register their application at dev.twitter.com. Follow the steps mentioned on Twitters OAuth generating page to create an authentication signature for your own application. The four more important keys that will be generated are **Consumer key, Consumer secret, Access token, Access token secret** (See example at Figure 1). Every time a change is made in your application, the aforementioned keys can get changed.

2. **Application-only authentication:** Application-only authentication is a form of authentication where your application makes API requests on its own behalf, without a user context. API calls are still rate limited per API method, but the pool each method draws from belongs to your entire application at large, rather than from a per-user limit. For instance, in

a 15-min window, 180 *GET statuses/user_timeline* requests are allowed for application-user authentication while 300 requests can be made for application-only authentication. However, not all API methods support application-only authentication. More details on how to issue application-only requests can be find at Application-only authentication Page, which may take more steps than application-user authentication.



Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read-only (modify app permissions)
Owner	[redacted]
Owner ID	[redacted]

Application Actions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token secret with anyone.

Access Token	[redacted]
Access Token Secret	[redacted]
Access Level	Read and write
Owner	[redacted]
Owner ID	[redacted]

Figure 1: OAuth Keys Generated

In this assignment, you can choose to use any forms of authentication as long as you can collect sufficient samples to estimate the fraction.

3 Using python-twitter

Your program should first create an instance of the Twitter API class, which creates an object with several methods that your application will use.

1. import twitter

2. api = twitter.Api(consumer_key=‘consumer_key’,consumer_secret=‘consumer_secret’, access_token_key=‘access_token’, access_token_secret=‘access_token_secret’)

Now, using the *api* object you can make calls to the Twitter APIs. For instance, the following call returns a collection of the most recent Tweets posted by the user whose user id is 21:

3. status = api.GetUserTimeline(user_id=21)

4. print status[0]

The typical format of the response *status[0]*, i.e. a tweet, will look like:

```
{"created_at": "Thu Aug 27 03:14:26 +0000 2015", "favorite_count": 6, "favorited":
false, "id": 636738512679628800, "lang": "en", "retweet_count": 2, "retweeted": false,
"source": "<a href=\"http://tapbots.com/tweetbot\" rel=\"nofollow\">Tweetbot for i
\u039fS</a>", "text": "This tweet is intended for a mature audience.", "truncated":
false, "user": {"created_at": "Tue Mar 21 21:03:20 +0000 2006", "description": "Helped
create @Twitter.", "favourites_count": 15855, "followers_count": 39195,
"friends_count": 973, "geo_enabled": true, "id": 21, "lang": "en", "listed_count":
1263, "location": "San Francisco, CA", "name": "Dom Sagolla",
"profile_background_color": "FFFFFF", "profile_background_tile": false,
"profile_image_url": "https://pbs.twimg.com/profile_images/614491054608232448/
wPmPUyS9_normal.jpg", "profile_link_color": "0084B4", "profile_sidebar_fill_color":
"http://abs.twimg.com/images/themes/theme15/bg.png", "profile_text_color": "333333",
"protected": false, "screen_name": "dom", "statuses_count": 7129, "time_zone": "Pacific
Time (US & Canada)", "url": "http://t.co/CUmcFalyeE", "utc_offset": -25200}}
```

Remark: some user have left Twitter or changed their account, so you may get Exception when you query for their time lines using user id.

One can parse these strings from the obtained response. However, there is a better way of accessing these fields.

6. user = status[0].GetUser() — returns the information of the user

7. print user

The typical format of the *user* will look like:

8. print user.GetCreatedAt() — returns the datetime that the user account was created on Twitter

The meaning of these fields of a tweet and a user can be found at Tweets Page and Users Page respectively.

To read the full APIs of the Python package *twitter*, you can run the following commands:

```
{"created_at": "Tue Mar 21 21:03:20 +0000 2006", "description": "Helped create
@Twitter.", "favourites_count": 15855, "followers_count": 39194, "friends_count": 973,
"geo_enabled": true, "id": 21, "lang": "en", "listed_count": 1263, "location": "San
Francisco, CA", "name": "Dom Sagolla", "profile_background_color": "FFFFFF",
"profile_background_tile": false, "profile_image_url": "https://pbs.twimg.com/
profile_images/614491054608232448/wPmPUyS9_normal.jpg", "profile_link_color": "0084B4",
"profile_sidebar_fill_color": "http://abs.twimg.com/images/themes/theme15/bg.png",
"profile_text_color": "333333", "protected": false, "screen_name": "dom",
"statuses_count": 7129, "time_zone": "Pacific Time (US & Canada)", "url": "http://t.co/
CUmcFalyeE", "utc_offset": -25200}
```

7. $ pydoc twitter.Status

8. $ pydoc twitter.User

9. $ pydoc twitter.DirectMessage