Lab Assignment 2: Crawl Data using API

Due: 11:59 PM Jan 27, Tuesday

In Lab 2, you will know what are REST (**Re**presentational State **T**ransfer)) APIs, how to use a typical REST API that requires authentication. In Section 1, you will learn using the HTTP requests to download public data from a server. Section 2 is the real assignment, you are required to get a piece of Twitter data with its API, parse the JSON formatted response, and do some basic analysis.

1 A simple REST API [1]

An API, or **a**pplication **p**rogramming **i**nterface, is kind of like a coding contract: it specifies the ways a program can interact with an application. For example, if you want to write a program that reads and analyzes data from Twitter, you'd need to use the Twitter API, which would specify the process for authentication, important URLs, classes, methods, and so on.

For an API or web service to be RESTful, it must do the following:

- Separate the client from the server
- Not hold state between requests (meaning that all the information necessary to respond to a request is available in each individual request; no data, or state, is held by the server from request to request)
- Use HTTP and HTTP methods (as explained in the next section).

Try the following code out in Spyder, and

```
import requests
# Make a GET request here and assign the result to kittens:
kittens = requests.get('http://placekitten.com/')
print kittens.status_code
print kittens.text[559:1000]
```

Several things to be noticed:

- 'http://placekitten.com/' is a server name, not a website. It's where the data is stored. The URL address is also called API 'endpoint'.
- There are four types of request type: GET, POST, PUT, REMOVE. They are implemented in `urllib` or `requests` package.
- `kittens.status_code` indicates whether the request is successful or not.

[1]This part is based on http://www.codecademy.com/en/tracks/apis-python

2 Play with Twitter API

2.1 Part 1 - Get your own Twitter data

Before start, the following figure will help you to understand how API works with authentication using the OAuth protocol. Before the OAuth 1.0, users need to give username and password to an application in order to share their information. After OAuth 1.0, third-party applications will not hold your secret credentials on their server. You are safer in some sense. Twitter implements

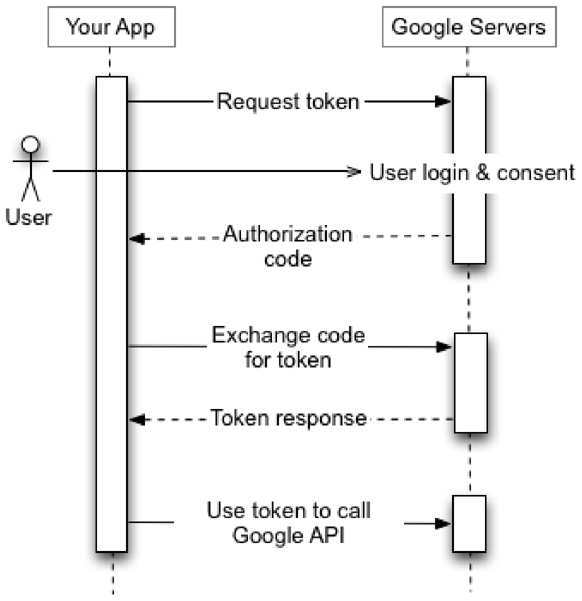


Figure 1: OAuth Web Flow

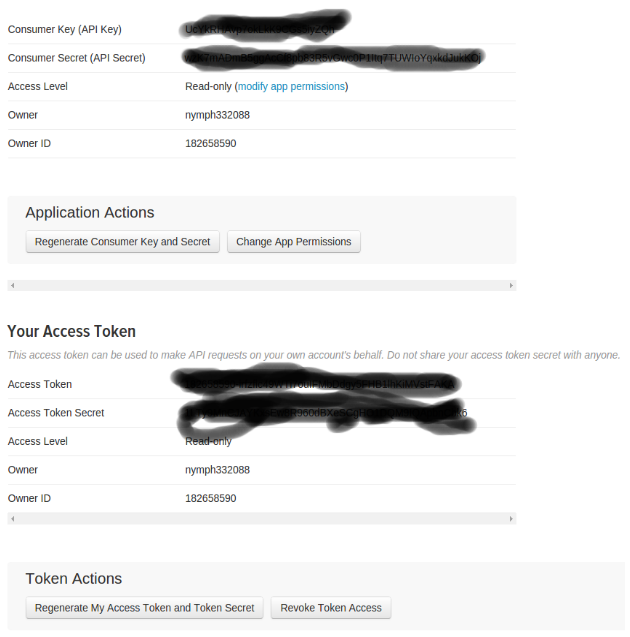
OAuth 1.0A as its standard authentication mechanism, and in order to use it to make requests to Twitter's API, you'll need to go to https://dev.twitter.com/apps and create a sample application. There are four primary identifiers you'll need to note for an OAuth 1.0A workflow: consumer key, consumer secret, access token, and access token secret. Note that you will need an ordinary Twitter account in order to login, create an app, and get these credentials. Read more about Twitter APIs

The first assignment is to use the Twitter stream API to download certain amount of data.

- Create a twitter account if you do not already have one.
- Go to https://apps.twitter.com/ and log in with your twitter credentials.
- Click "Create New App".
- Fill out the Name, Description, Website fields and agree the terms. Put in any website you want if you don't have one you want to use.
- On the next page, click the "Keys and Access Tokens" tab along the top, then scroll all the way down until you see the section "Your Access Token"
- Click the button "Create My Access Token".

- You will now copy four values into twitter_stream_api.py. These values are your "Consumer Key (API Key)", your "Consumer Secret (API Secret)", your "Access token" and your "Access token secret". All four should now be visible on the "Keys and Access Tokens" page. Open twitter_stream_api.py and set the variables corresponding to the API key, API secret, access token, and access secret. You will see code like the below:

```
ckey = "<Enter api key>"
csecret = "<Enter api secret>"
atoken = "<Enter your access token key here>"
asecret = "<Enter your access token secret here>"
```



Consumer Key (API Key)	[redacted]
Consumer Secret (API Secret)	[redacted]
Access Level	Read-only (modify app permissions)
Owner	nymph332088
Owner ID	182658590

Application Actions

Regenerate Consumer Key and Secret | Change App Permissions

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-only
Owner	nymph332088
Owner ID	182658590

Token Actions

Regenerate My Access Token and Token Secret | Revoke Token Access

Figure 2: Twitter Token Generation Page

- Run the twitter_stream_api.py from either way. You will see the flood of twitter data showing on the console.

```
# 1. From command line
$ python twitter_stream_api.py
# 2. From the Spyder
In []: runfile('twitter_stream_api.py')
```

- Redirect the standard output to the 'twitter_data.txt' file.

```
# 1. From Windows or Linux terminal
$ python twitter_stream_api.py > twitter_data.txt
# 2. From the Spyder
1). Uncomment line 30
2). In []: runfile('twitter_stream_api.py')
```

Assignment: Get your own Twitter data by selecting other parameters for **filter()** function. For example, setting the **filter**(locations = [-75.280291,39.867005,-74.955831,40.137959]), you will get all tweets from Philadelphia. To get bounding box for other areas, the klokantech's tool can be used. Run the program for at least 5 minutes. Submit the first 20 lines of the 'twitter_data.txt' file. You can get the first 20 lines without opening the 'twitter_data.txt' by doing the following.

```
# 1. From Linux terminal
$ head -n 20 twitter_data.txt > submission.txt
# 2. From Windows command line
$ gc twitter_data.txt | select -first 20 > submission.txt
```

Submissions: 'submission.txt'

2.2 Part 2 - Wordcount program

From Part 1, you will get the twitter data in 'twitter_data.txt', which is in JSON format (You can consider it as a complex Python dictionary of lists of dictionaries..., read more). Python provides JSON module to parse the JSON format into Python dictionary (they looks identical to each other, but they are 'different'). In Spyder, type the following command and see the first line of your output.

```
import json
from pprint import pprint
twitter = open('twitter_data.txt','r')
line = next(twitter)
data = json.loads(line)
pprint(data)
```

You will see something printed like this:

```
{u'contributors': None,
 u'coordinates': None,
 u'created_at': u'Sun Jan 11 15:42:08 +0000 2015',
 u'entities': {u'hashtags': [],
               u'symbols': [],
               u'trends': [],
               u'urls': [],
               u'user_mentions': []},
 u'favorite_count': 0,
 u'favorited': False,
 u'filter_level': u'medium',
 u'geo': None,
 u'id': 554302247526752257,
 u'id_str': u'554302247526752257',
 u'in_reply_to_screen_name': None,
 u'in_reply_to_status_id': None,
 u'in_reply_to_status_id_str': None,
 u'in_reply_to_user_id': None,
 u'in_reply_to_user_id_str': None,
 ...
```

The ‘text’ field matters and you can access data simply by dictionary indexing `data['text']`.

```
        In []: data['text']
        Out[]: u'Worst basketball game of my life'
```

Assignment: In this assignment, you need to analyse the downloaded ‘tweets’, i.e. the ‘text’ field in each line of the ‘twitter_data.txt’. In the skeleton script wordcount.py, you need to complete two functions `word_count()` and `top_k()`. If you run the script in Spyder, you will get

```
        In []: runfile('wordcount.py',args='twitter\_data.txt')
        Out[]: Hello twitter_data.txt
                Number of lines: 470
```

The desired output for `word_count()`: The output should be printed to the console. Each line of output should contain a term, followed by a space, followed by the frequency of that term in the **entire ‘twitter_data.txt’ file**. There should be one line per **unique** term in the entire file. Even if 25 tweets contain the word ‘lol’, the term ‘lol’ should only appear once in your output (and the frequency will be at least 25!). Each line should be in the format *term:frequency*, for example, *basketball 36*. You should print all **unique** terms in the file.

The desired output for `top_k()`: The output should be printed to the console. The format is as the same as that for `word_count()` function. While, instead of printing all terms, you only need to print the top k most frequent terms (with top k largest frequencies, k is specified by the user).

Submission: ’wordcount.py’