

Lab Assignment 2: Crawl Data using API

Due: 11:59 PM Jan 27, Tuesday

In Lab 2, you will know what are REST (**R**epresentational **S**tate **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.

¹This 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

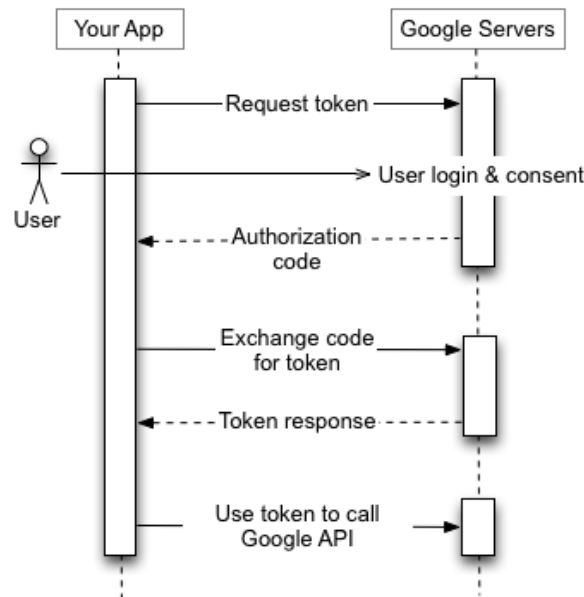


Figure 1: OAuth Web Flow

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>"

```

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 terminal
$ 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 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 terminal
$ head -n 20 twitter_data.txt > submission.txt
# 2. From Windows command line (not tested)
$ 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 look 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 3 functions `print_dict()`, `word_count()` and `top_k()` in this part. 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 functionality of `print_dict()`: For a dictionary, print out each key value pair on a single line, with a space between key and value. See the examples in the following paragraphs.

The desired output for `word_count()`: The output should be a dictionary. And you should be able to call your `print_dict()` function to display the output to console. Each key in the dictionary is a term, the value is the frequency of that term in the **entire 'twitter_data.txt' file**. For example, for three tweets: ['twitter good', 'twitter twitter', 'too too']. The desired printing would be as followed (the rank is not important):

```
twitter 3  
good 1  
too 2
```

Even if 2 tweets contain the word 'twitter', the term 'twitter' should only appear once in your output (and the frequency will be at least 2!). You should print all **unique** terms in the file.

The desired output for `top_k()`: 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). For example, if `top_k(2)` is called, the desired printing would be:

```
twitter 3  
too 2
```

2.3 Part 3

Assignment: In part 2, you calculated the term 'too' twice in the third tweet. It might not be reasonable to do so, since the repetition might be insignificant. In part 3, implement another `word_count_2()` function. The same output format is required as for the `word_count()` function. While for the same three tweets: ['twitter good', 'twitter twitter', 'too too'], the desired output is.

```
twitter 2
good 1
too 1
```

It means the frequency of a term in a single tweet will be counted only once, no matter how many repetitions there are.

2.4 Part 4: Preprocessing

Assignment: After the first 3 parts, you can see the top terms might contain some stop words like ‘the’, ‘a’, ‘an’, or punctuations ‘@’, ‘...’. Implement a **clean()** function.

The desired functionality of **clean()**:

- Remove stop words in the tweets
- Remove punctuations in the tweets

Using the clean function to preprocess the tweets. Then call **word_count()**, **word_count_2()**, **top_k()** again. Check whether there is any difference of the output. Report your findings in doc or txt, e.g, will the preprocess really make a difference? It shouldn’t be too long, 1-3 paragraphs would be fine.

2.5 Part 5: Have fun

Assignment: You have done a lot of analysis for tweets from Philadelphia, can you do the same for tweets from Boston district or any other place. Compare the top terms of the two places you found, what can you find? What are the top words for different places? Are the top words different? Summarize your findings in doc or txt.

3 Submission Requirements

Requirements: You can modify the skeleton of ‘wordcount.py’ in order to make your functions easily to be used. The only rule is to follow the output format or the functionality requirements of each function.

Submission: For Part 1, submit the ‘submission.txt’. Write all your code in ‘wordcount.py’, submit ‘wordcount.py’. Submit the report for Part 4. Submit the report for Part 5.