Health Monitor Analytics

GROUP 1

Demo 1

User Documentation

D. Takacs

G. Bati

J. Abdulbaqi

I. Paraskevakos

K. Dong

Table of Contents

1. Interactive Map 3
2. PerCapita HeatMap 4
3. CalorieMeter 4
4. Play 7

1. Interactive Map

To make this code work, first, you have to install the following software:

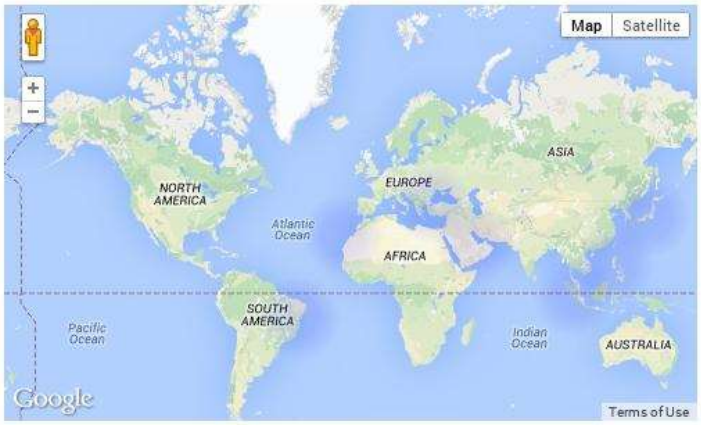
- Python (www.python.org)
- MongoDB (www.mongodb.org)
- redis server (www.redis.io)
- Flask (flask.pocoo.org)
- Tweepy (www.tweepy.org)

Then you have to make this configuration:

- Twitter Streaming API credentials (you have to register at twitter development website dev.twitter.com)
- create a capped collection in MongoDB named "tweets" using the following query: db.createCollection("log", { capped : true, size : 5242880, max : 5000 })

Now, follow these steps to make it work (bash commands in the parenthesis):

1. start MongoDB ($ mongod).
2. start Redis server ($ redis-server).
3. execute ‘twitter_stream.py’ ($ python twitter_stream.py)
4. execute ‘data.py’ ($ python data.py)
5. open the map.html (write in the web browser address: localhost:5000/map.html). You have to be connected to the Internet during the execution time.
6. A google map will be shown in the page and the data will be appear on the map gradually.



2. PerCapita HeatMap

The PerCapita HeatMap will follow identically the user documentation of the Interactive Map feature, up to step 5. Depending on how we plan to implement both, either the Per Capita HeatMap will be featured on a separate webpage, or, it will be an option that is embedded in the Interactive Map itself.

3. CalorieMeter

This part of the document explains to the user how to compile and run the Python code of the CalorieMeter.

For the first time, the following packages must be downloaded using the following command:

```
pip install twitter
pip install prettytable
pip install json
pip install Counters
```

Then simply the code could be compiled using the following command:

```
python CalorieMeter.py
```

```
import twitter
import json
def oauth_login():
    # XXX: Go to http://twitter.com/apps/new to create an app and get values
    # for these credentials that you'll need to provide in place of these
    # empty string values that are defined as placeholders.
    # See https://dev.twitter.com/docs/auth/oauth for more information
    # on Twitter's OAuth implementation.

    CONSUMER_KEY = 'XXX'
    CONSUMER_SECRET = 'XXX'
    OAUTH_TOKEN = 'XXX'
    OAUTH_TOKEN_SECRET = 'XXX'
    auth = twitter.oauth.OAuth(OAUTH_TOKEN, OAUTH_TOKEN_SECRET,
                               CONSUMER_KEY, CONSUMER_SECRET)
    twitter_api = twitter.Twitter(auth=auth)
    return twitter_api
# Sample usage
twitter_api = oauth_login()
# Nothing to see by displaying twitter_api except that it's now a
# defined variable
print twitter_api
###############
q = 'spent #LoseIt' # CalorieMeter intresting keywords and hashtags
count = 100
# See https://dev.twitter.com/docs/api/1.1/get/search/tweets for more details
search_results = twitter_api.search.tweets(q=q, count=count)
statuses = search_results['statuses']
# Iterate through 5 more batches of results by following the cursor
for _ in range(5):
    print "Length of statuses", len(statuses)
    try:
        next_results = search_results['search_metadata']['next_results']
    except KeyError, e: # No more results when next_results doesn't exist
        break
    # Create a dictionary from next_results, which has the following form:
    # ?max_id=313519052523986943&q=NCAA&include_entities=1
    kwargs = dict([ kv.split('=') for kv in next_results[1:].split("&") ])
```

```
    search_results = twitter_api.search.tweets(**kwargs)
    statuses += search_results['statuses']
####################
status_texts = [ status['text']
                 for status in statuses ]
screen_names = [ user_mention['screen_name']
                 for status in statuses
                     for user_mention in
status['entities']['user_mentions'] ]
hashtags = [ hashtag['text']
             for status in statuses
                 for hashtag in status['entities']['hashtags'] ]
# Compute a collection of all words from all tweets
words = [ w
          for t in status_texts
              for w in t.split() ]

# Explore the first 5 items for each...
print json.dumps(status_texts[0:5], indent=1)
print json.dumps(screen_names[0:5], indent=1)
print json.dumps(hashtags[0:5], indent=1)
print json.dumps(words[0:5], indent=1)
##########################

from collections import Counter
for item in [words, screen_names, hashtags]:
    c = Counter(item)

##########################

from prettytable import PrettyTable

for label, data in (('Word', words),
                    ('Screen Name', screen_names),
                    ('Hashtag', hashtags)):
    pt = PrettyTable(field_names=[label, 'Count'])
    c = Counter(data)
    [ pt.add_row(kv) for kv in c.most_common()[:10] ]
```

```
    pt.align[label], pt.align['Count'] = 'l', 'r' # Set column alignment
    print pt
```

The results of this code should simply display the following:

```
<twitter.api.Twitter object at 0x7fc81d2f50d0>
Length of statuses 100
Length of statuses 200
Length of statuses 200
[
 "I spent 15 minutes boxing. 248 calories burned. #LoseIt",
 "I spent 2 hours cleaning. 331 calories burned. #LoseIt",
 "I spent 2 hours cleaning. 315 calories burned. #LoseIt",
 "I spent 20 minutes rowing a rowing machine. 155 calories burned. #LoseIt",
 "I spent 35 minutes riding a stationary bike. 263 calories burned. #LoseIt"
]

+----------+-------+
| Word     | Count |
+----------+-------+
| calories |   200 |
| burned.  |   200 |
| I        |   200 |
| spent    |   200 |
| #LoseIt  |   200 |
| minutes  |   163 |
| hour     |    57 |
| 1        |    57 |
| doing    |    50 |
| and      |    42 |
+----------+-------+
```

4. Play

In order for the user to play the game, the following instructions should be followed:

1) Go to the folder where the game.py file exists
2) Open a Python Shell

3) Type the following command to import the Game class: `from game import Game`
4) Type the following command to import the Game class: `from datetime import date`
5) Create a date of the last update : `lastUpdate = date.today()`
6) Create a new area by typing:
   `AreaTest = Game('County','Essex',249,500,150,lastUpdate,15000,'NJ')`
   and you have create an object for Essex County,NJ
7) Type : `AreaTest.NewPoints()` and the new points for this area will be calculated.
8) To see the points type `AreaTest.__get__('points')`

To run it for a new area here is what its input of step 6 is and their valid values:

i. 'County': Is the type of the area. Can take either 'County' or 'State'
ii. 'Essex': The name of the Area. Can take any name
iii. 249 : the points of that area. Can take any non-negative value
iv. 500: The number of new Tweets. Any non-negative value
v. 150 : The number of Tweet of the last update. Any non-negative value
vi. lastUpdate : The date of the last update. Object of python class date.
vii. 15000: Total number of tweets for this area. Any non-negative number
viii. 'NJ' : The initial of the State the County is in. If the Area is a state leave it blank.

As an example the user should see file gameTest.py