LEEDS Institute for Data Analytics

UNIVERSITY OF LEEDS

Collecting & Analyzing Twitter data – an Introduction

Viktoria Spaiser

UAF in Political Science Informatics,

School of Politics and International Studies

Accessing Twitter data

1) **Twitter Streaming API** (**A**pplication **P**rogramming **I**nterface)
   - Real-time Twitter data collection of tweets
   - Spritzer sample is free (1% of all public tweets)
   - Other samples or full data (e.g. Firehose) are subject to a charge

   https://dev.twitter.com/streaming/overview

2) **Twitter REST APIs (in particular Twitter Search API)**
   - Historic (past 7 days!) data collection of tweets (e.g. based on hashtags)
   - Collection of tweets by location (place operator of the Search API)
   - Collection of followers & friends data for specified Twitter user(s)
   - API Rate limits apply

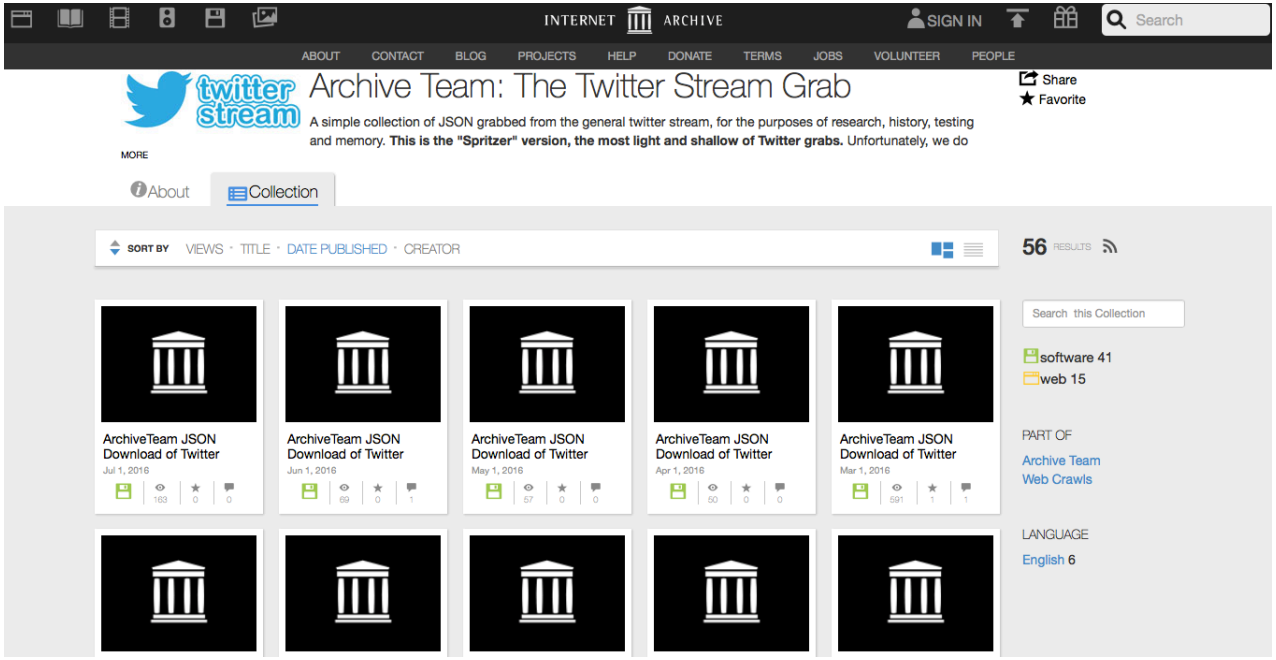
   https://dev.twitter.com/rest/public

Accessing Twitter data

Missed the date?

- No panic, there is an archive for Streamed Twitter data

https://archive.org/details/twitterstream



Here you can download historic Twitter Streaming API data in JSON format

Accessing Twitter data

What you need to access Twitter data via Twitter APIs

1. Twitter account
2. Obtain Authentication & Authorization (OAuth):
   - this requires registration as a developer (developing an app, even if you will not) with Twitter, register here: https://apps.twitter.com
   - you will get: Consumer Key, Consumer Secret, Access token, Acess token secret

WITHOUT THESE YOU WILL NOT BE ABLE TO ACCESS DATA VIA TWITTER APIS!!!

Accessing Twitter data

1. **Python**
   (Python 2.7 + Anaconda for Python 2.7 recommended)
   useful packages: *tweepy*, *Twython*, *simplejson*, *nltk (Natural Language Toolkit)*
   install Python 2.7: https://www.python.org/downloads/
   install Anaconda: https://www.continuum.io/downloads
   install pacakges: e.g. type "`pip install tweepy`" in terminal/shell

2. **R** (packages *twitteR* and *ROAuth*):
   https://www.r-bloggers.com/setting-up-the-twitter-r-package-for-text-analytics/

3. **Other programming languages like Java etc.**

4. **NodeXL** (no coding, Windows only, for Social Network Analyses only):
   http://www.pewinternet.org/files/2014/02/How-we-analyzed-Twitter-social-media-networks.pdf

5. **Mecodify** (new, free software for extracting & visualizing Twitter data, no coding, soon available from: http://www.mecodem.eu, developed by **Walid Al-Safaq:** walid.al-saqaf@ims.su.se)

6. **LIDA** seems to have developed some software to collect tweets data, contact **David Batty**: d.batty@leeds.ac.uk

Twitter data, unprocessed

JSON (**J**ava**S**cript **O**bject **N**otation) format

one tweet!


```
{"in_reply_to_status_id_str":null,"geo":null,"text":"\u041c\u0438\u0440\u043e\u0432\u044b\u0435 \u0446\u0435\u043d\u044b \u043d\u0430 \u043d
\u0435\u0444\u0442\u044c \u0438\u0437\u043c\u0435\u043d\u0438\u043b\u0438\u0441\u044c \u043c\u0430\u043b\u043e\u0432\u0430\u0436\u043d\u043e:
\u0426\u0435\u043d\u044b \u043d\u0430 \u043d\u0435\u0444\u0442\u044c \u043d\u0430 \u043c
\u0438\u0440\u043e\u0432\u044b\u0445 \u0440\u044b\u043d\u043a\u0430\u0445 \u0432 \u0441\u0440\u0435\u0434\u0443 \u0438\u0437\u043c\u0435\u043d
\u0438\u043b\u0438\u0441\u044c \u0440\u0430\u0437\u043d\u043e\u043d\u0430\u043f\u0440\u0430\u0432\u043b\u0435\u043d\u043d\u043e... http:\/\/t.co\/
54CrdQ8k","in_reply_to_status_id":null,"in_reply_to_user_id_str":null,"id_str":"136960060445306880","contributors":null,"coordinates":null,"trunc
ated":false,"entities":{"hashtags":[],"urls":[{"indices":[116,136],"expanded_url":"http:\/\/bit.ly\/rQr6EU","display_url":"bit.ly\/
rQr6EU","url":"http:\/\/t.co\/54CrdQ8k"}],"user_mentions":[]},"source":"\u003Ca href=\"http:\/\/twitterfeed.com\" rel=\"nofollow
\"\u003Etwitterfeed\u003C\/a\u003E","place":null,"in_reply_to_user_id":null,"favorited":false,"created_at":"Thu Nov 17 00:13:17 +0000
2011","possibly_sensitive":false,"retweet_count":0,"in_reply_to_screen_name":null,"user":{"default_profile_image":false,"statuses_count":
49831,"profile_image_url_https":"https:\/\/si0.twimg.com\/profile_images\/1115168193\/
rss2_512_normal.png","screen_name":"rendytob","contributors_enabled":false,"geo_enabled":false,"profile_use_background_image":true,"id_str":"1856
19687","profile_text_color":"333333","location":"\u041c\u043e\u0441\u043a
\u0432\u0430","show_all_inline_media":false,"profile_background_image_url":"http:\/\/a3.twimg.com\/profile_background_images\/142967743\/
rainbow_bright_dl.jpg","favourites_count":0,"description":"\u041d\u043e\u0432\u043e\u0441\u0442\u0438: \u0412 \u043c\u0438\u0440\u0435, \u041e
\u0431\u0449\u0435\u0441\u0442\u0432\u043e, \u0421\u0428\u0410, \u0422\u0435\u043c\u0430 \u0434\u043d\u044f, \u041a\u0438\u043d\u043e, \u041f
\u0440\u043e\u0438\u0441\u0448\u0435\u0441\u0442\u0432\u0438\u044f, \u0424\u0443\u0442\u0431\r\n\u041e\u0441\u0432\u0435\u0434\u043e
\u043c\u043b\u0435\u043d - \u0437\u043d\u0430\u0447\u0438\u0442\u0442 \u0432\u043e\u043e
\u0440\u0443\u0436\u0435\u043d!","lang":"en","profile_link_color":"0084B4","verified":false,"notifications":null,"time_zone":null,"is_translator"
:false,"follow_request_sent":null,"profile_background_image_url_https":"https:\/\/si0.twimg.com\/profile_background_images\/142967743\/
rainbow_bright_dl.jpg","friends_count":1619,"profile_background_color":"C0DEED","listed_count":
94,"following":null,"profile_background_tile":false,"created_at":"Wed Sep 01 12:41:01 +0000
2010","protected":false,"profile_sidebar_fill_color":"DDEEF6","followers_count":1909,"profile_image_url":"http:\/\/a0.twimg.com\/profile_images\/
1115168193\/rss2_512_normal.png","name":"\u041d\u043e\u0432\u043e\u0441\u0442\u0438","id":
185619687,"default_profile":false,"utc_offset":null,"profile_sidebar_border_color":"C0DEED","url":null},"id":
136960060445306880,"retweeted":false}
{"in_reply_to_status_id_str":null,"geo":null,"text":"\u0413\u043e\u0441\u0434\u043e\u043b\u0433 \u0421\u0428\u0410 \u043f\u0440\u0435\u0432\u044b
\u0441\u0438\u043b 15 \u0442\u0440\u043b\u043d \u0434\u043e\u043b\u043b\u0430\u0440\u043e\u0432: \u0412 \u0441\u0435\u043d\u0442\u044f
\u0431\u0440\u0435 \u0433\u043e\u0441\u0434\u043e\u043b\u0433 \u0421\u0428\u0410 \u043e\u0446\u0435\u043d\u0438\u0432\u0430\u043b\u0441\u044f
\u0432 14,9 \u0442\u0440\u043b\u043d \u0434\u043e\u043b\u043b\u0430\u0440\u043e\u0432. \"\u041e\u0431\u0449\u0435\u0439 \u043e\u0431\u044a
\u0435\u043c \u0444\u0435\u0434\u0435\u0440... http:\/\/t.co\/
```


foreign languages (here Russian) or special characters encoded in *unicode*

Twitter data, unprocessed


```
{
    "created_at": "Thu Apr 30 21:53:11 +0000 2015",
    "id": 593895901623496700,
    "id_str": "593895901623496704",
    "text": "This is a #test tweet @LoveforTestingT with an image. http://t.co/ZvgHovKZq4",
    "source": "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web Client</a>",
    "truncated": false,
    "in_reply_to_status_id": null,
    "in_reply_to_status_id_str": null,
    "in_reply_to_user_id": null,
    "in_reply_to_user_id_str": null,
    "in_reply_to_screen_name": null,
    "user": {
        "id": 2993982541,
        "id_str": "2993982541",
        "name": "Test Demo",
        "screen_name": "jondee_test",
        "location": "Denver, CO",
        "url": null,
        "description": "this is a test account.",
        "protected": false,
        "verified": false,
        "followers_count": 2,
        "friends_count": 43,
        "listed_count": 0,
        "favourites_count": 0,
        "statuses_count": 30,
        "created_at": "Sat Jan 24 00:12:53 +0000 2015",
```

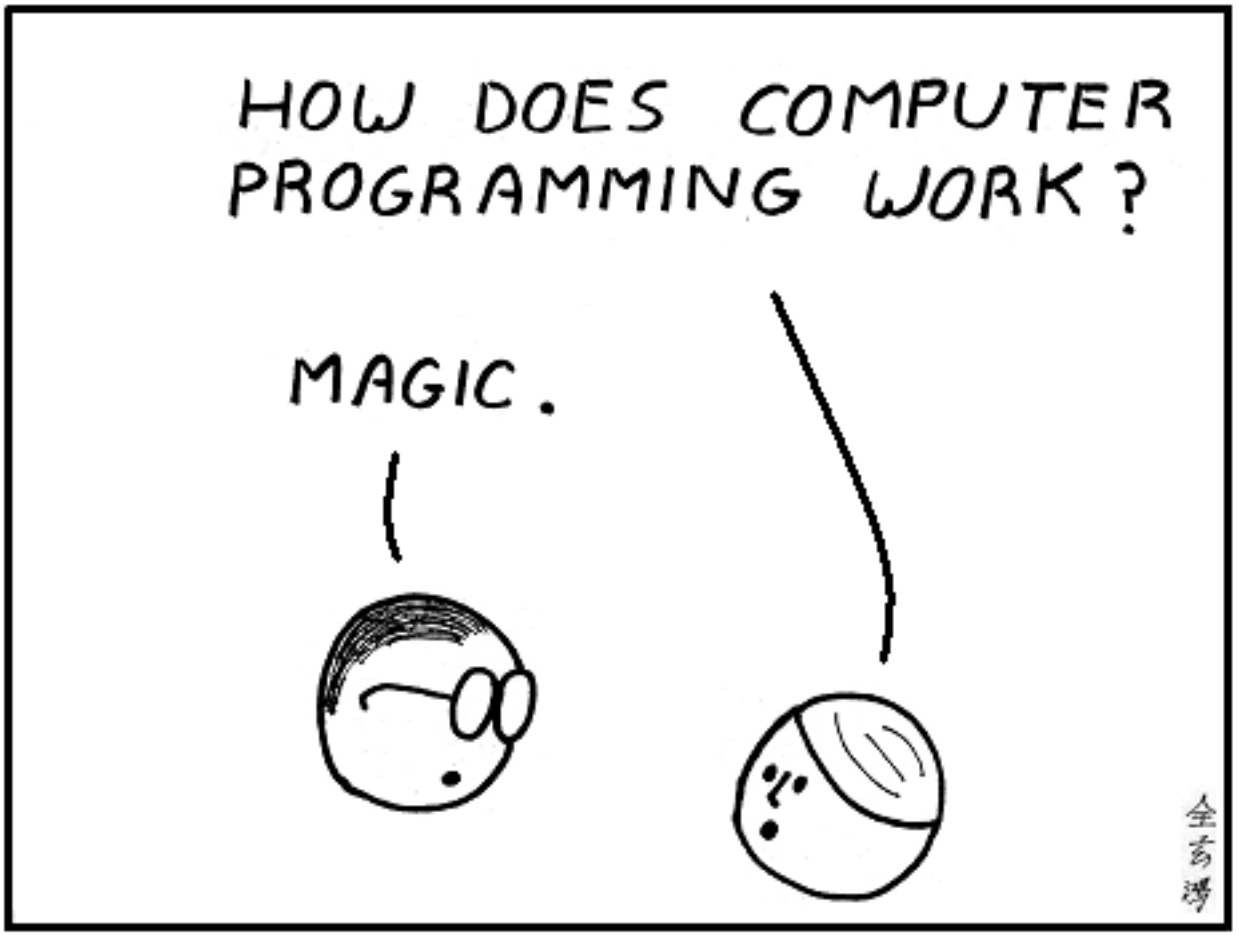

...

Twitter data, key variables

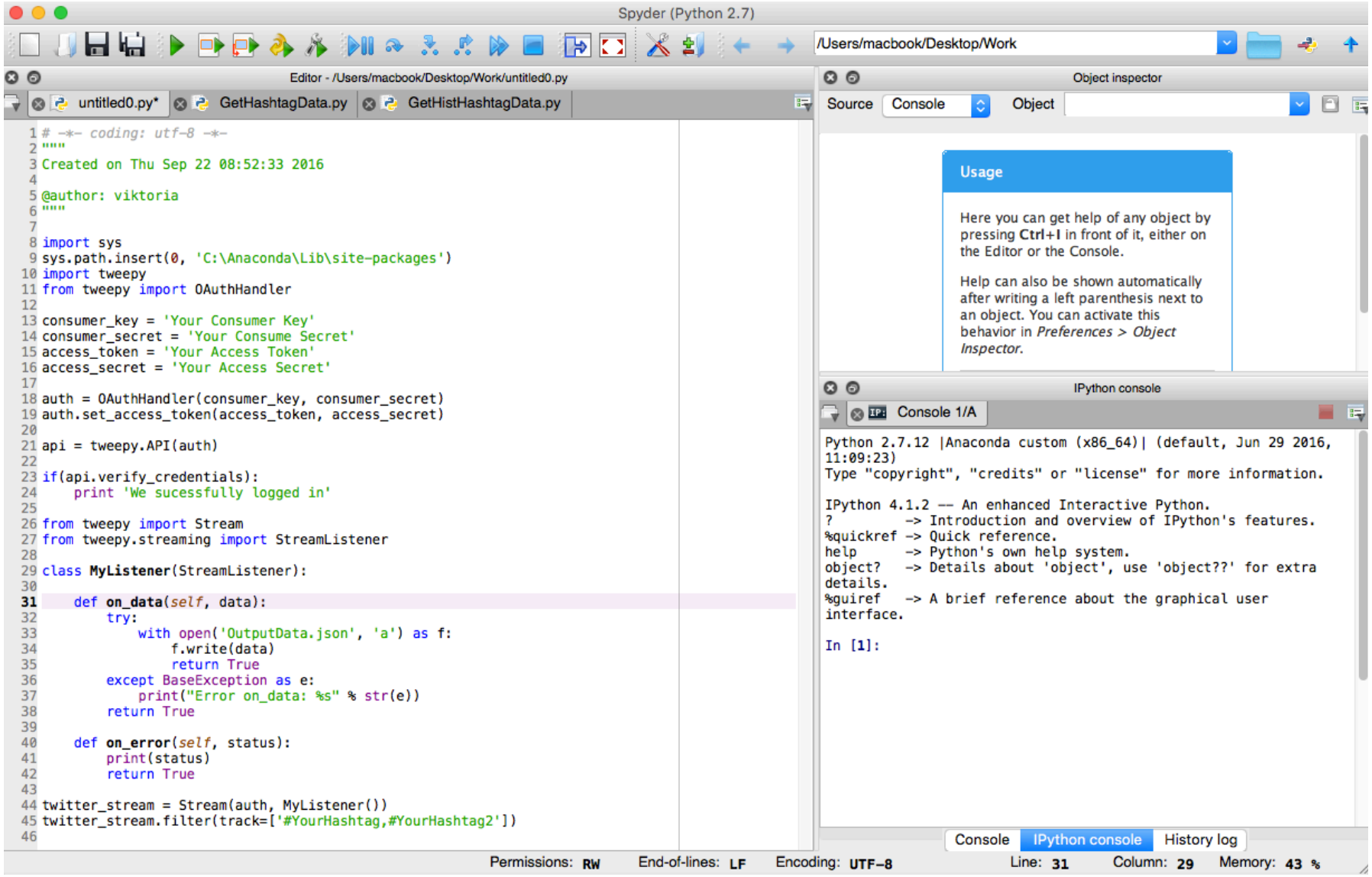
Field	Description
`id`	Unique tweet ID number
`text`	Tweet text, if retweet then starts with `RT @screen_name:`
`created_at`	Timing of tweet creation, or of Twitter account creation if nested within the Twitter `user` field
`place/coordinates`	Latitude, longitude coordinates, if `geo-enabled` set to "`true`" (has to be activated by user, per default deactivated (value "`false`")
`user_mentions/ screen_name`	Indicates whether and which Twitter user is mentioned (@) in the tweet
`in_reply_to_screen_ name`	Indicates whether the twitter was a reply and in that case to which Twitter user (if not a reply value "`null`")
`user/screen_name`	User name of Twitter user
`user/location`	Location information (e.g. name of town) as provided by Twitter user
`user/name`	Full name of Twitter user as provided by Twitter user
`user/description`	Profile description of Twitter user

and many more variables…: http://support.gnip.com/sources/twitter/data_format.html

Ok, let's start coding then…



Getting data from the Streaming API



Getting data from the Search API

```
import sys
sys.path.insert(0, 'C:\Anaconda\Lib\site-packages')
import tweepy
from tweepy import OAuthHandler

consumer_key = 'Your Consumer Key'
consumer_secret = 'Your Consumer Secret'
access_token = 'Your Access Token'
access_secret = 'Your Access Secret'

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)

api = tweepy.API(auth, wait_on_rate_limit=True, wait_on_rate_limit_notify=True)

if (not api):
    print ("Can't Authenticate")
    sys.exit(-1)

searchQuery = '#YourHashtag'
maxTweets = 10000000
tweetsPerQry = 100
fName = 'OutputData.json'

sinceId = None

max_id = -1L

tweetCount = 0
print("Downloading max {0} tweets".format(maxTweets))
with open(fName, 'w') as f:
    while tweetCount < maxTweets:
        try:
            if (max_id <= 0):
                if (not sinceId):
                    new_tweets = api.search(q=searchQuery, count=tweetsPerQry)
                else:
                    new_tweets = api.search(q=searchQuery, count=tweetsPerQry,
                                            since_id=sinceId)
            else:
                if (not sinceId):
                    new_tweets = api.search(q=searchQuery, count=tweetsPerQry,
                                            max_id=str(max_id - 1))
                else:
                    new_tweets = api.search(q=searchQuery, count=tweetsPerQry,
                                            max_id=str(max_id - 1),
                                            since_id=sinceId)
            if not new_tweets:
                print("No more tweets found")
                break
            for tweet in new_tweets:
                s = str(tweet)
                f.write(s.encode("ascii"))
            tweetCount += len(new_tweets)
            print("Downloaded {0} tweets".format(tweetCount))
            max_id = new_tweets[-1].id
        except tweepy.TweepError as e:
            print("some error : " + str(e))
            break

print ("Downloaded {0} tweets, Saved to {1}".format(tweetCount, fName))
```

Processing JSON data

```
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 22 15:00:59 2016

@author: viktoria
"""

import string
import json

# READ IN JSON FILE
path = 'OutputData.json'
infile = open(path, 'rU')

# NAMES FOR HEADER ROW IN OUTPUT FILE
fields = "id screen_name name location".split()

# PREPARE YOUR OUTPUTFILE
outfn = "users_location.txt"
outfp = open(outfn, "w")
outfp.write(string.join(fields, "\t") + "\n")  # header

# READING IN AND WRITING OUT DATA
#for entry in infile:
for line in open(path,'r'):
    tweet = json.loads(line)
    # CREATE EMPTY DICTIONARY
    r = {}
    for f in fields:
        r[f] = ""
    # ASSIGN VALUE OF THE FIELDs IN JSON TO THE FIELDS IN OUR DICTIONARY
    r['id'] = tweet['id']
    r['screen_name'] = tweet['user']['screen_name']
    r['name'] = tweet['user']['name']
    r['location'] = tweet['user']['location']
    # CREATE EMPTY LIST
    lst = []
    # ADD DATA FOR EACH VARIABLE
    for f in fields:
        lst.append(unicode(r[f]).replace("\/", "/"))
    # WRITE ROW WITH DATA IN LIST
    outfp.write(string.join(lst, "\t").encode("utf-8") + "\n")
outfp.close()
```

Natural Language Processing

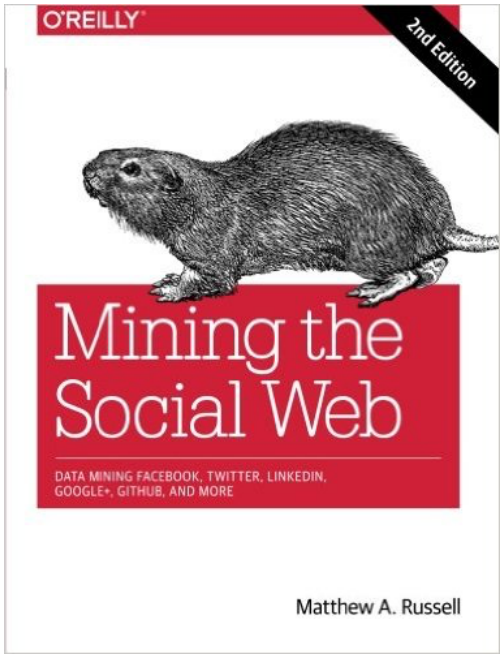
```
import json
import csv
import nltk
from nltk.corpus import stopwords
from nltk.collocations import BigramCollocationFinder

path = 'OutputData.json'
outpath = '/Users/macbook/Desktop/Work'
outfn = "ProcessedData.txt"

rutwout = list()
for line in open(path,'r'):
    block = json.loads(line)
    tweet = block["text"]
    rutwout.append(tweet + "\n")

stopset = stopwords.words('english')
filter_stops = lambda w: len(w)< 2 or w in stopset


def fdist(file):
    freq = nltk.FreqDist()
    words = nltk.tokenize.regexp_tokenize(str(rutwout), "[\w']+")
    words = [word.lower() for word in words]
    words = [word for word in words if len(word) > 2]
    if words not in stopset:
        freq = nltk.FreqDist(words)
        freqwords=freq.most_common(300)
        return freqwords

def saveFreqWord(freq):
    temp_dict = dict(freq)
    writer = csv.writer(open('Wordcounts.csv', 'wb'))
    for key, value in temp_dict.items():
        writer.writerow([key, value])

saveFreqWord(fdist(path))


def bicolloc(file):
    words = nltk.tokenize.regexp_tokenize(str(rutwout), "[\w']+") #tokenising words
    words = [word.lower() for word in words]
    words = [word for word in words if len(word) > 2]
    bgm = nltk.collocations.BigramAssocMeasures() #initialising the BigraAssociationMeasures
    bcf = BigramCollocationFinder.from_words(words) #initialising the BigramCollocationFinder
    bcf.apply_word_filter(filter_stops) #filtering words
    scored = bcf.score_ngrams(bgm.student_t)[:300] #computing 100 trigram-collocation with the highest scores based on T-Test
    open('/Users/macbook/Desktop/Work/Bigrams.txt','w').writelines(str(scored))
    temp_dict = dict(scored)
    csvData = []
    for (col1, col2), col3 in temp_dict.iteritems():
        csvData.append("%s, %s, %s" % (col1, col2, col3))
    f = open('Bigrams.csv', 'w')
    f.write("\n".join(csvData))
    f.close()

bicolloc(path)
```

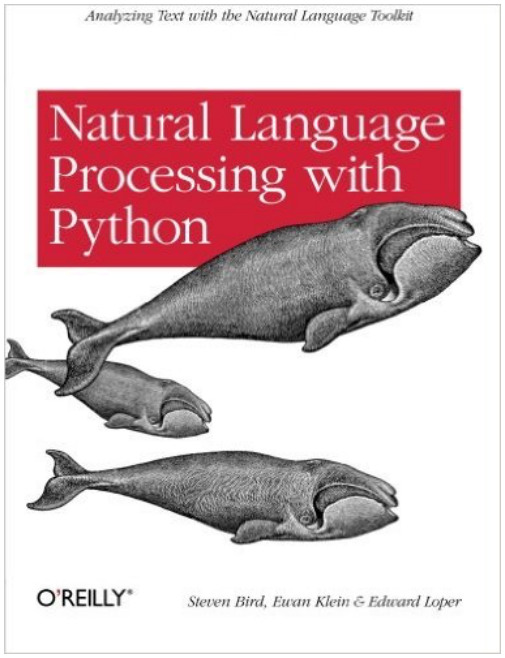
Geo-location Processing

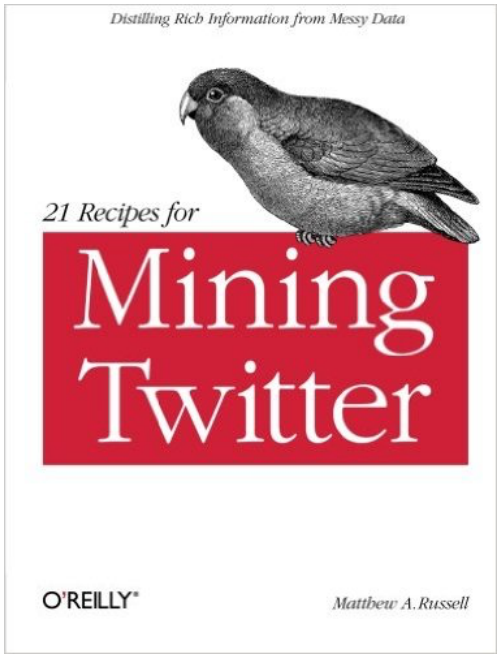
```python
# -*- coding: utf-8 -*-
"""
Created on Thu Sep 22 19:26:59 2016

@author: viktoria
"""
import json

path = 'OutputData.json'
with open(path, 'r') as f:
    geo_data = {
        "type": "FeatureCollection",
        "features": []
    }
    for line in f:
        tweet = json.loads(line)
        if tweet['coordinates']:
            geo_json_feature = {
                "type": "Feature",
                "geometry": tweet['coordinates'],
                "properties": {
                    "text": tweet['text'],
                    "created_at": tweet['created_at']
                }
            }
            geo_data['features'].append(geo_json_feature)

# Create GeoJSON
with open('geo_data.json', 'w') as fout:
    fout.write(json.dumps(geo_data, indent=4))

```

You can use GeoJSON for instance in QGIS or to create interactive maps with Leaflet http://leafletjs.com/examples/geojson.html

Recommended Further Reading







And many sources on the internet…