# IN4252 Web Science & Engineering
# Hands-on Assignment
# Social Web Data Analytics

## 1 Task 1: Retrieving via Twitter API

The objective of this task is to get you familiar with Twitter Streaming API. To achieve this, you will first create a data retrieving application. Next, you will: 1) monitor the public stream of tweet data; 2) filter tweets sent from Amsterdam and those related to COVID-19, by writing your code. Besides, you are also encouraged to try Twitter REST API, though it is not mandatory.

### 1.1 Create an Application

Twitter Streaming API provides developers with Twitter's global stream of Tweet data. If properly used, you can continuously receive Tweet data being pushed to you, possibly with some supported filters (which you will try in subsection 1.3). You can refer to the official documentation[1] for more details.

As in March 2013, Twitter enforced OAuth authentication for using Streaming API. That means you need to register an application on Twitter Developers before you can use it. You can access the application page via the following link (a Twitter account needed):

`https://dev.twitter.com/apps`

By clicking on **Create an app**, you need to provide some basic information about the application. After that, you will be able to get the following parameters:

- Consumer key;
- Consumer secret;
- Access token;
- Access token secret.

You will need them to get your application authenticated with OAuth.

[1] `https://developer.twitter.com/en/docs/tweets/filter-realtime/guides/connecting.html`

## 1.2 Accessing Public Streaming API

This sub-task requires you to write your code to crawl the tweets. The advantage of using code is that you can directly add some event processing code, e.g., extracting interesting fields (e.g., tweet text), or reporting on crawling progress. You can either write the crawling code from scratch or use existing libraries. Here are two recommended libraries for Python and Java:

- **Python:** You can use *tweepy* package for monitoring public stream. The code example can be found via following link: `https://github.com/tweepy/tweepy/blob/master/examples/streaming.py`
- **Java:** You can make use of *twitter4j* library. Code example: `http://twitter4j.org/en/code-examples.html`

As an example, if you choose to use *Tweepy*, then you may use the following lines of code to crawl the Twitter Stream:

```
from __future__ import absolute_import, print_function

from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream

# Go to http://apps.twitter.com and create an app.
# The consumer key and secret will be generated for you after
consumer_key="[your_consumer_key]"
consumer_secret="[your_consumer_secret]"

# After the step above, you will be redirected to your app's page.
# Create an access token under the the "Your access token" section
access_token="[your_access_token]"
access_token_secret="[your_access_token_secret]"

class StdOutListener(StreamListener):
    """ A listener handles tweets that are received from the stream.
    This is a basic listener that just prints received tweets to stdout.
    """
    def on_data(self, data):
        print(data)
        return True

    def on_error(self, status):
        print(status)

if __name__ == '__main__':
    l = StdOutListener()
    auth = OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_token, access_token_secret)


stream = Stream(auth, l)
stream.sample()
```

For this part, you need to monitor the Public Stream for 10 mins and answer the following questions:

1. **What is the starting and ending time of the data that you have crawled?**
2. **What is the id of the first tweet you got? And the last one?**
3. **How many tweets did you get?**
4. **How large is the result file (uncompressed file in JSON format)?**

## 1.3 Filtering Tweets sent from Amsterdam

In this sub-task, you will crawl the following two types of tweets by filtering the public tweet stream: 1) tweets sent from Amsterdam, which has the following geo-coordinates "[4.61, 52.27, 5.07, 52.50]"; and 2) tweets related to COVID-19. Use the following list of keywords for filtering tweets related to COVID-19: covid, covid-19, corona, coronavirus, lockdown.

This sub-task requires you to write your own code to crawl the tweets. You need to monitor the Public Stream for 2 hours. Please answer the following questions:

1. **How many tweets sent from Amsterdam did you get?**
2. **How many tweets are related to COVID-19?**

## 1.4 Twitter REST API

Besides Streaming API, there is a set of REST APIs that provide most of Twitter functionality. For example, given an id of the Tweet, you can use Twitter REST API to retrieve the tweet including the metadata. Still, you need to use OAuth for authentication. However, there is usually a rate-limit for REST APIs. That means there is a certain limit on how frequent you can use the APIs. For example, the API call of retrieving a tweet with a given tweet id have a rate-limit of 180 times per 15 minutes[2].

You may try a couple of these APIs by utilizing the API wrapper, either *tweepy* if you are using Python or *twitter4j* for Java.

[2]`https://developer.twitter.com/en/docs/basics/rate-limiting`

## 2 Task 2: Exploratory and Confirmatory Data Analysis

In this task, you are going to conduct exploratory and confirmatory data analysis for 5 features in our Tweet Relevance Judgment problem. Before starting the analysis, however, you need to have your working environment and the data prepared.

### 2.1 Working Environment

We recommend you to set up your own environment for this assignment. First of all, you need to have *Python*[3], *pandas*[4], *scipy*[5], *numpy*[6] and *matplotlib*[7] installed. *pandas* is a easy-to-use and powerful data analysis tool and *matplotlib* is a python 2D plotting library. Installing *Python* should be straightforward. For *pandas*, *scipy*, *numpy* and *matplotlib*, according to your operating system, you have the following recommended options:

- for Unix-like OS, a easy way to install these package is by using Python tool `pip`, which can be installed by following the instructions at `https://pip.pypa.io/en/latest/installing.html`. Then, the installation of *pandas* can be done by the following command in your console: `pip install pandas`. Similarly, *matplotlib*, *scipy* and *numpy* can be installed by `pip install matplotlib/scipy/numpy`.
- for Windows, the best resource for third-party python packages is Christoph Gohlke's Python Extension Packages for Windows repository: `http://www.lfd.uci.edu/%7Egohlke/pythonlibs/`.
  **Note:** With Python 2.7.9+ and 3.4+ the PIP functionality is also supported (by default) on Windows operating systems.

### 2.2 Dataset

The dataset is available via the following link: `https://goo.gl/BFHE66`. This is the dataset that we used for investigating what makes a tweet relevant to a given topic. Each instance (= each line in the data file) in this dataset lists the 25 features (25 columns) and the relevance judgment (the last field) for a (tweet, topic)-pair. Please refer to Table 1 if you want to know the meaning of each feature/column.

**This task requires you to (i) analyze a total of 5 features (4 listed below and you need to select another one of your interests) and (ii) present the analysis results and plots of the 5 features**.

[3]`https://www.python.org`
[4]`http://pandas.pydata.org`
[5]`http://www.scipy.org`
[6]`http://www.numpy.org`
[7]`http://matplotlib.org`

1. #entities;
2. #entityTypes;
3. #tweetsPosted;
4. sentiment.

Table 1: Description of Columns in the Data File

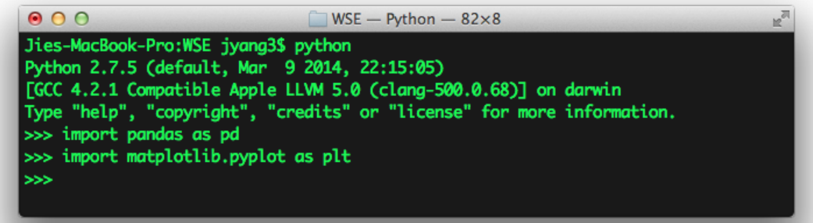
| **Column name** | **Description** |
|---|---|
| text_score | Retrieval Score (based on some IR models) |
| text_score_expansion | Retrieval Score using expansion on topic |
| hashtag | If the tweet contains hashtag(s) |
| hasURL | If the tweet contains URL(s) |
| isReply | If the tweet is a reply to another tweet |
| length | The length (in characters) of the tweet |
| sentiment | The polarity of the sentiment of this tweet |
| tweet_topic_time_diff | The time difference between the tweet and the query |
| semantic_overlap | Overlap of named entities between the topic the tweet |
| #entityTypes | #types of Named-Entities (NE) extracted from tweet |
| #entities | #NEs extracted from tweet |
| organization_entities | #NEs with type of Orgranization extracted from tweet |
| person_entities | #NEs with type of Person extracted from tweet |
| work_entities | #NEs with type of Work extracted from tweet |
| event_entities | #NEs with type of Event extracted from tweet |
| species_entities | #NEs with type of Species extracted from tweet |
| places_entities | #NEs with type of Places extracted from tweet |
| nFollowers | #followers that the author has |
| nFriends | #followees that the author has |
| nFavorties | How many times has the tweets been marked as favorite by others? |
| nListed | How many lists has the author been listed in? |
| isVerified | Whether the tweet is posted by a verified account or not? |
| isGeoEnabled | Is there a geolocation attached to this tweet? |
| twitterAge | How many years has been since the author signed up on Twitter? |
| #tweetsPosted | How many tweets has the author posted on Twitter? |

## 2.3 Explore the Features

In this part, you will visualize the features and obtain the descriptive statistics. You can use any data visualization tools (R, gnuplot, Matlab, etc.) you like. If you do not have any experience with this, you can follow this tutorial of using Python for data visualization (and for hypothesis testing later).

Assuming that you have all the tools ready, you can start reading the data file and visualize the features. First of all, you can enter Python interactive environment by typing `python` in your console, and import *pandas* and *matplotlib*. Figure 1 shows the corresponding actions in console. In this Figure *pandas* and *matplotlib* are renamed to `pd` and `plt`, respectively.

Figure 1: Start Python and import packages.



```
Jies-MacBook-Pro:WSE jyang3$ python
Python 2.7.5 (default, Mar  9 2014, 22:15:05)
[GCC 4.2.1 Compatible Apple LLVM 5.0 (clang-500.0.68)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>>
```

Now, you can start reading the data file to a data structure, and visualize the feature you are investigating. For instance in the following example, the original csv data is loaded to `df`, from which the feature #entities is extracted to `nr_entities`, whose histogram is then visualized.

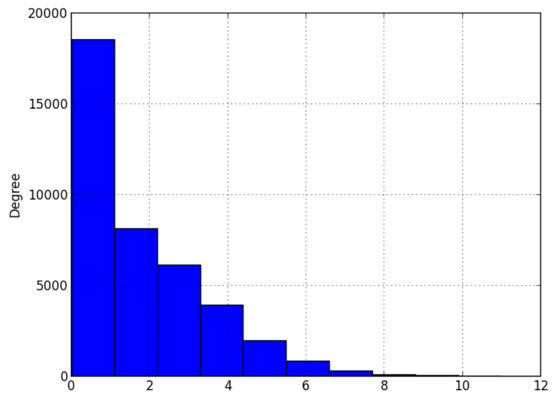
```
df=pd.read_csv("[path_to_your_data_file]")
nr_entities = df['#entities']
plt.figure();
nr_entities.plot(kind='hist')
plt.show()
```

The result of this code snippet is shown in Figure 2.

Plotting the histogram is sometimes necessary. For instance, it is useful for choosing a parametric hypothesis testing method (e.g., *t-test*), which requires a specific feature distribution. For other plots (e.g., boxplot, scatter plot), please refer to pandas documentation[8]. Please use appropriate visualization techniques to explore the given features. Note that if the values of a feature conform to a power-law distribution[9], you can use a log-scaled plot.

[8]`http://pandas.pydata.org/pandas-docs/version/0.15.0/visualization.html`
[9]`http://en.wikipedia.org/wiki/Power_law`

Figure 2: Histogram of #entities.



**Obtaining descriptive statistics** Again, any tool that supports the calculation of *mean*, *std* values can be used. Here we continue using Python (*pandas*) to get the result. Compared to visualization, obtaining descriptive statistics in *pandas* is easier: simply use the `describe` function. The following code returns the descriptive statistics of #entities of relevant and non-relevant tweets. The corresponding result is shown in Figure 3.

```
nr_entities_relevant = df[df['relevanceJudge']==1]['#entities']
nr_entities_non_relevant = df[df['relevanceJudge']==0]['#entities']
nr_entities_relevant.describe()
nr_entities_non_relevant.describe()
```

### 2.4 Confirm Which Features are Discriminative

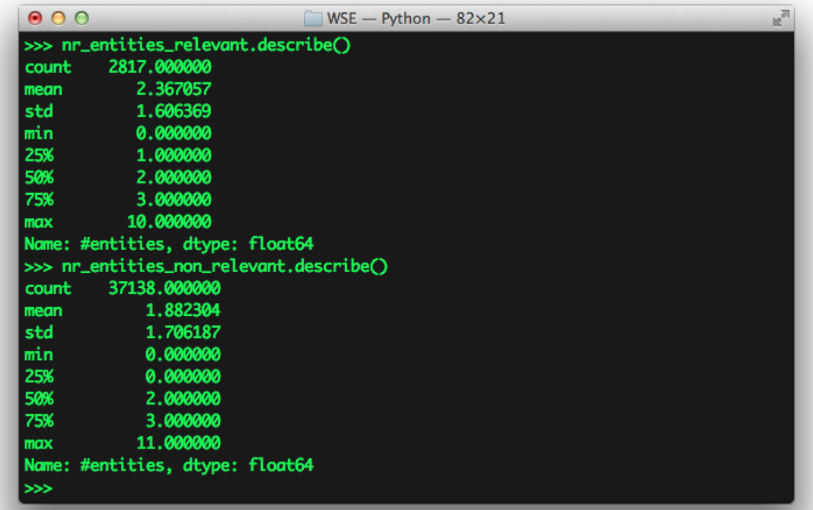
Here you will use hypothesis testing techniques to confirm whether a feature is useful in discriminating relevant tweets from non-relevant ones. In this tutorial, we continue using Python for hypothesis testing. The following code will directly output the result of Mann-Whitney *U* test.

```
import numpy as np
from scipy.stats import mannwhitneyu
u, p_value = mannwhitneyu(nr_entities_non_relevant, nr_entities_relevant)
```

## 3 To be delivered

You are expected to compress following files in a zip file. Please name this file as "**[Lastname].[Firstname].zip**", e.g. Jie.Yang.zip, and upload it via the

Figure 3: Descriptive statistics of #enttities of relevant and non-relevant tweets.



```
>>> nr_entities_relevant.describe()
count    2817.000000
mean        2.367057
std         1.606369
min         0.000000
25%         1.000000
50%         2.000000
75%         3.000000
max        10.000000
Name: #entities, dtype: float64
>>> nr_entities_non_relevant.describe()
count    37138.000000
mean         1.882304
std          1.706187
min          0.000000
25%          0.000000
50%          2.000000
75%          3.000000
max         11.000000
Name: #entities, dtype: float64
>>>
```

Brightspace platform. Please add a new submission in the *Practical 2 - Social Web Data Analytics* folder.

- A report (in PDF) which includes: 1) answers to the questions (in bold font) of Task 1; 2) visualizing plots, descriptive statistics, and results of hypothesis testing for the 4 stated features and 1 feature selected by you (either discriminate or non-discriminate) in Task 2; 3) the problems that you encounter while doing the homework (if there are any).
- Your code for crawling tweets sent from Amsterdam and those related to COVID-19, specified in Section 1.3.

## 4 Deadline

Please submit your homework before December 16, 2022.

## 5 Q&A

- Jie Yang: J.Yang-3@tudelft.nl