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

Task 1: Using the Mastodon Streaming API

The goal of this task is to get you familiar with using Streaming APIs on the example of Mastodon, an open-source social media platform akin to Twitter. To this end, you will create a data-retrieving application. Next, you will (1) monitor the public stream of Status – analogous to tweets – data; and (2) filter Statuses related to the Israel-Palestine conflict by writing some code. Besides, we encourage you to try the Mastodon REST API (but that is not mandatory).

1.1 Create an Account

Mastodon's Streaming API provides developers with access to the global stream of Status data. If properly used, you can continuously receive status data being pushed to you, possibly with some filters (which you will try in Subsection 1.3). You can refer to the official documentation[1,2] for more details.

Mastodon enforces OAuth authentication to use its Streaming API. That means you need to create an account and register an application with Mastodon before you can use it. Mastodon works in a decentralized manner (there are many independent servers), but we will use mastodon.social, its original server.

1. Create an account:
`https://mastodon.social/auth/sign_up`

2. Create an app:
You can do this in two ways:

- With `cURL`: `https://docs.joinmastodon.org/client/token/#app`
- By writing some code (see the code example in the following section).

[1] `https://docs.joinmastodon.org/methods/streaming/`
[2] `https://docs.joinmastodon.org/methods/filters/#v1`

1.2 Accessing the Public Streaming API

This sub-task requires you to write your own code to crawl the Statuses. The main advantage of using code is that you can directly add some event processing code, e.g., to extract interesting fields (such as Status content) or to report on the crawling progress. You can write the crawling code from scratch or use an existing library. Here are two recommended libraries for Python and Java:

- **Python:** You can use `Mastodon.py` package to create an app and monitor the public stream. The documentation can be found via following link: https://mastodonpy.readthedocs.io/en/stable/10_streaming.html
- **Java:** You can make use of the `Mastodon4J` library. An example is available at https://github.com/sys1yagi/mastodon4j#streaming-api

To help you get started, you can make use of the following code snippet to create an app and crawl the Mastodon Public Stream with `Mastodon.py`.

```python
from mastodon import Mastodon, StreamListener
# Create an app and obtain the client credentials.
# Substitute the name of your app and name of the credentials file.
Mastodon.create_app("<your_app_name>",
    api_base_url="https://mastodon.social",
    to_file="<your_client_credential_filename>.secret"
)
# Substitute the name of the credentials file.
mastodon = Mastodon(
    client_id="<your_client_credential_filename>.secret"
)
# Log in with your account.
# Substitute your e-mail and password
# and the name of the credentials file.
mastodon.log_in("<your_email>", "<your_password>",
    to_file="<your_user_credential_filename>.secret"
)
class StdOutListener(StreamListener):
    """
    A listener handles Statuses received from the stream.
    This basic listener simply prints received statuses to stdout.
    You need to modify the code to solve the tasks.
    """
    def on_update(self, status):
        print(status)
        return True

    def on_abort(self, err):
        print(err)

l = StdOutListener()
mastodon.stream_public(listener=l)
```

Listing 1: Boilerplate code for `Mastodon.py`

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

1. **What is the start and end time of the data that you have crawled?**
2. **What is the ID of the first status you got? And the last one?**
3. **How many Statuses did you get in total?**
4. **How large is the uncompressed results file in JSON format?**

1.3 Filtering Statuses about the Israel-Palestine Conflict

In this sub-task, you will filter the public Status stream to find Statuses related to the Israel-Palestine Conflict. Make use of the following list of keywords for filtering: `israel`, `palestine`, `hamas`, `gaza`, `israel-palestine conflict`.

For this part you need to write your own code to crawl the statuses and monitor the Public Stream for **2 hours**. Please answer the following question:

5. **How many Statuses did you get in total?**
6. **How many of them are related to the Israel-Palestine Conflict?**

1.4 Mastodon REST API

Besides the Streaming API, Mastodon provides a number of REST API endpoints for various of its functionalities. For example, given an ID of a Status, you can use the REST API to retrieve the Status including its metadata. Again, to access the API you need to make use of OAuth for authentication.

Note that there is usually a rate limit for REST APIs, defining how frequently you can call the API endpoints. For example, the endpoint to retrieve a Status with a given ID allows you to make at most 300 calls per every 5 minutes[3].

We encourage you to try a few of these API endpoints using an API wrapper: either `Mastodon.py` for Python, or `Mastodon4j` for Java.

[3]https://docs.joinmastodon.org/api/rate-limits/

Task 2: Exploratory and Confirmatory Analysis

In the second task, we will look at Tweets instead. You are going to conduct exploratory and confirmatory data analysis for 5 features in a Tweet Relevance Judgment problem. Before you start with the analysis, you need to have a working environment and the data prepared.

2.5 Working Environment

We recommend you to set up your environment for this assignment. We encourage you to make use of `anaconda` or `venv` to create a virtual environment.

Then, you need to have `Python 3`, `pandas`, `scipy`, `numpy` and `matplotlib` installed. `pandas` and `numpy` are easy-to-use and powerful data analysis tool, `matplotlib` is a `Python` plotting library. Installing `Python` should be straightforward, for the packages you have the following options:

- for Unix-like OS, an easy way to install these packages is by using `pip`, which is automatically installed if you are working in a virtual environment or have downloaded Python from `python.org`. If, for some reason, you don't have `pip` installed, you can follow the instructions at the project website. Then, you can install a package `X` (`pandas` / `scipy` / `numpy` / `matplotlib`) by typing `pip install X` in your console.
- for Windows you can also install the packages from binaries, for instance, obtained from Christoph Gohlke's Python Extension Packages for Windows repository. **Note:** With `Python 2.7.9+` and `3.4+` the `pip` functionality is also supported (by default) on Windows operating systems.

2.6 Dataset

The dataset in a `.csv` format is available via the following link: `https://goo.gl/BFHE66`. The dataset allows us to investigate what makes a Tweet relevant to a given topic. Each instance (line in the data file) lists the values of 25 features (25 columns) and the relevance judgment (the last column) for a Tweet + topic pair. We explain the meanings of all features in Table 2.6.

This task requires you to (1) analyze a total of 5 features – 4 listed below and an additional one of your interests – and (2) present the results of your analysis and the plots for these 5 features.

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

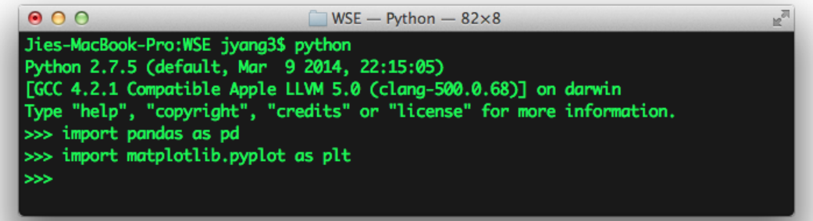
Column name	**Description**
`text_score`	Retrieval Score (based on Information Retrieval models)
`text_score_expansion`	Retrieval Score using expansion on topic
`hashtag`	If the Tweet contains hashtag(s)
`hasURL`	If the Tweet contains URL(s)
`isReply`	If the Tweet is a reply to another Tweet
`length`	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 Organization 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`	#follow*er*s of the author
`nFriends`	#follow*ee*s of the author
`nFavorties`	#times the Tweet has been marked as favorite by others
`nListed`	#lists where the author has been listed in
`isVerified`	Is the Tweet is posted by a verified account?
`isGeoEnabled`	Is there a geo-location attached to this Tweet?
`twitterAge`	Years since the author signed up on Twitter
`#tweetsPosted`	Number of Tweets posted by the author on Twitter

Table 1: Description of all features in the Tweet dataset

2.7 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, ...). If you do not have any experience with this, we encourage you to follow this tutorial using `Python` for data visualization (and for hypothesis testing later).

Assuming that you have the tools of preference ready, you can start by reading the data file and visualizing 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 the console. In this Figure `pandas` and `matplotlib` are renamed to `pd` and `plt`, respectively.



```
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
>>>
```

Figure 1: Start Python and import packages

Now, you can start reading the data file to a data structure, and visualize the feature you are investigating. For instance, in the following listing, 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()
```

Listing 2: Visualizing the `#entities` feature

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 the `pandas` documentation. Make use of appropriate visualization techniques to explore the given features. Note that if the values of a feature conform to a power law distribution[4], you can use a log-scaled plot.

[4]`http://en.wikipedia.org/wiki/Power_law`

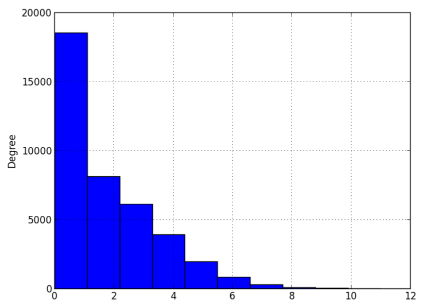


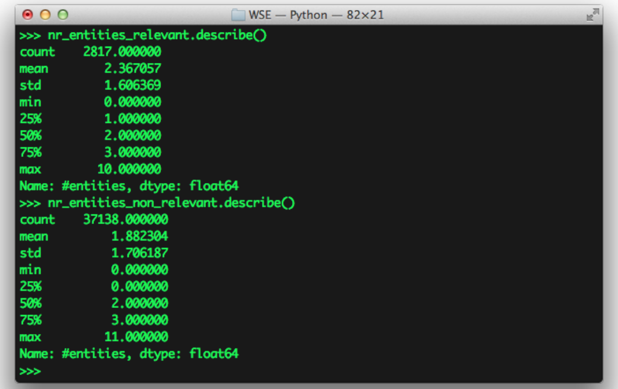
Figure 2: Histogram of `#entities`.

Obtaining descriptive statistics Again, any tool that supports the calculation of the *mean* and *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()
```

Listing 3: Describing the `#entities` feature



```
>>> 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
>>>
```

Figure 3: Statistics on `#entities` for relevant and non-relevant tweets.

2.8 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
)
```

Listing 4: Mann-Whitney U test

To be delivered

You are expected to compress the following files into a zip file. Please name this file as "`[Lastname].[Firstname].zip`", e.g. `Yang.Jie.zip`, and upload it via the Brightspace platform. Please add your submission in the *Practical 2 - Social Web Data Analytics* assignment. Your deliverable should contain:

- A report (in PDF) which includes: (1) answers to all questions in bold font of Task 1; (2) plots, descriptive statistics, and results (either "discriminative" or "non-discriminative") of hypothesis testing for the 4 stated features and 1 feature selected by you in Task 2; (3) the problems that you encounter while doing the homework (if there are any).
- Your code for crawling the Mastodon Public Stream filtering Statuses related to the Israel-Palestine Conflict, as used in Sections 1.2 and 1.3.

Deadline

Please submit your homework by December 15, 2023

Q&A

For questions you can contact Jie Yang: `J.Yang-3@tudelft.nl`