

**Streaming Data Collection and Deserialization
on Twitter using Python
- First Report -**

Big Data Hands-on Lab

Monday, June 6th 2016

Katharina Drechsler

Matrikelnumber: 5996281

Patrick Klose

Matrikelnumber: 5152067

Victoria Reibenspieß

Matrikelnumber: 4171151

Sergej Schetle

Matrikelnumber: 5996267

Contents

1	Introduction	3
2	Python Installation and Configuration	4
2.1	Python Installation - Anaconda for Windows	4
2.2	Python Packages	7
3	Twitter	7
3.1	Twitter API - Tweepy	7
3.2	Twitter Developer Account	8
4	Server environment	10
5	Data collection and Python code	10
6	Problems	12
7	Conclusion	12

List of Figures

1	Website Continuum Downloads	4
2	Installation - Welcome to Anaconda	5
3	Installation type	6
4	Installing process	6
5	Tweepy installation	7
6	Twitter Account	8
7	Twitter Application Creation	9
8	Twitter Application Settings	9
9	Raspberry Pi	10
10	Raspberry Pi Screenshot	10

1 Introduction

Big Data involve complex, large-volume and increasing data sets with multiple as well as autonomous sources. With the rapid development of data storage, networking and the data collection volume, the usage of Big Data is at the moment quickly expanding in all science and engineering domains, including biological, physical and biomedical sciences.¹ Public health is only one of many disciplines where Big Data is making a big difference.

In addition, public health surveillance applications increasingly exploit data from social media platforms such as Twitter as key resources. Twitter offers multiple key benefits to public health surveillance. First, the dataset is extensive and immediately accessible. Large amounts of user-generated, freely available online content, which permit efficient and partly or fully automated, real-time monitoring of public sentiment, allow for bottom-up detection of emergent patterns. Such patterns may not be readily discovered by traditional surveillance methodologies such as pre-formulated surveys. Second, the dataset from Twitter is user-centric, thereby reflecting trends that surveys may not be able to capture or that are not discussed by users in a more formal context. Finally, Twitter demographics include difficult-to-reach and therefore underrepresented groups.²

In our project on the topic "Streaming Data Collection and Deserialization on Twitter using Python" we decided to use these advantages of twitter to analyze a public health topic. In particular, while writing this report, we are collecting a large corpus of smoker-related Twitter posts, filtering posts according to their relevance to smoker-related content, and conducting content analyses of these posts. In addition, we are blending this data with data from other sources. We believe that such an analysis offers insights for public health stakeholders, like insurance companies and government institutions. Since smoking is the leading cause of death among preventable diseases and kills around 6 million people per year worldwide³, the costs of smoking burden public health systems and obtaining deeper insights into smoking behavior could help to prevent smoking and smoke-related causes of death in the future. We will be creating two use cases in the context of public health, which will be discussed and presented in

¹ Wu, Xindong, et al. "Data mining with big data." *Knowledge and Data Engineering, IEEE Transactions on* 26.1 (2014): 97-107

² Myslín, Mark, et al. "Using twitter to examine smoking behavior and perceptions of emerging tobacco products." *Journal of medical Internet research* 15.8 (2013): e174

³ <http://www.who.int/mediacentre/factsheets/fs339/en/>

detail in the second phase of our project. Because of data availability and language issues, we restrict both use cases to the US.

Since the project is cutted into two phases, this report will only document the results of the first phase, including our project assignment, the technical setup and used technologies, as well as the used tools and our programming code for the data collection. Moreover, we will discuss problems we faced during the first phase of the project. Thus, our first report is based on observations of the installing and configuration process between May 16 and June 6.

2 Python Installation and Configuration

We decided to install the Python distribution "Anaconda", which contains Python and a collection of over 720 open source packages for data analytics, data science and scientific computing. We chose the version 4.0 due to the high number of with this version compatible packages available, and due to the extensive documentation and easy usability. In addition, with Anaconda, a huge amount of further Python packages can easily be installed, updated or deleted using the "conda" command. The other compatible Python packages can be managed using the well-known "pip" command.

2.1 Python Installation - Anaconda for Windows

Firstly, we downloaded the Anaconda installer for Python 3.5 from the website (Fig.1):

<https://www.continuum.io/downloads>

Anaconda for Windows

PYTHON 2.7	PYTHON 3.5
<div>WINDOWS 64-BIT GRAPHICAL INSTALLER</div> <div>335M</div>	<div>WINDOWS 64-BIT GRAPHICAL INSTALLER</div> <div>345M</div>
<div>Windows 32-bit Graphical Installer</div> <div>281M</div>	<div>Windows 32-bit Graphical Installer</div> <div>283M</div>
Behind a firewall? Use these zipped Windows installers .	

Figure 1: Website Continuum Downloads

Secondly, by double-clicking on the downloaded .exe file we started the installation process (Fig.2). We followed the prompts in order to install the application to the default location.

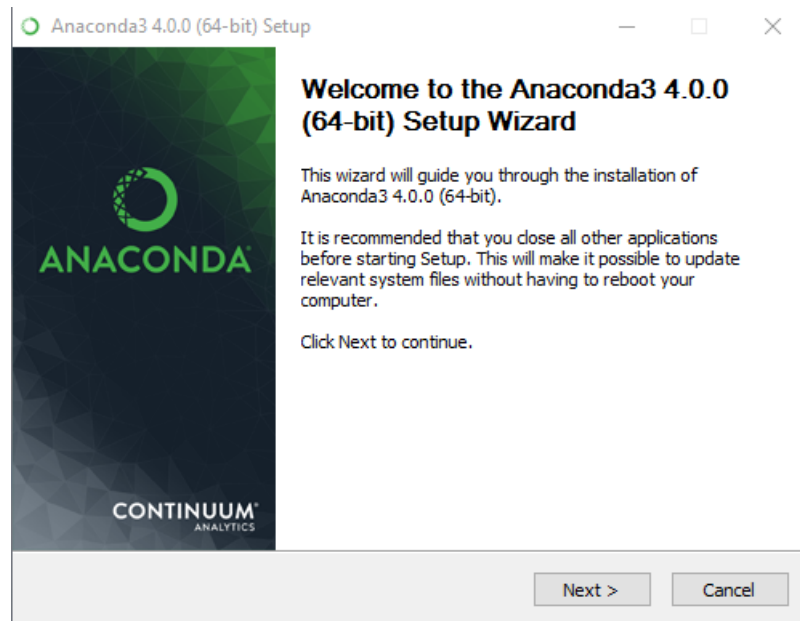


Figure 2: Installation - Welcome to Anaconda

Even though Anaconda may be installed system-wide, which does require administrator permissions, on Microsoft Windows it is best to install Anaconda for the local user. This does not require administrator permissions and delivers the most robust type of installation. Thus, we chose the option "Just Me" during the installation process (Fig.3).

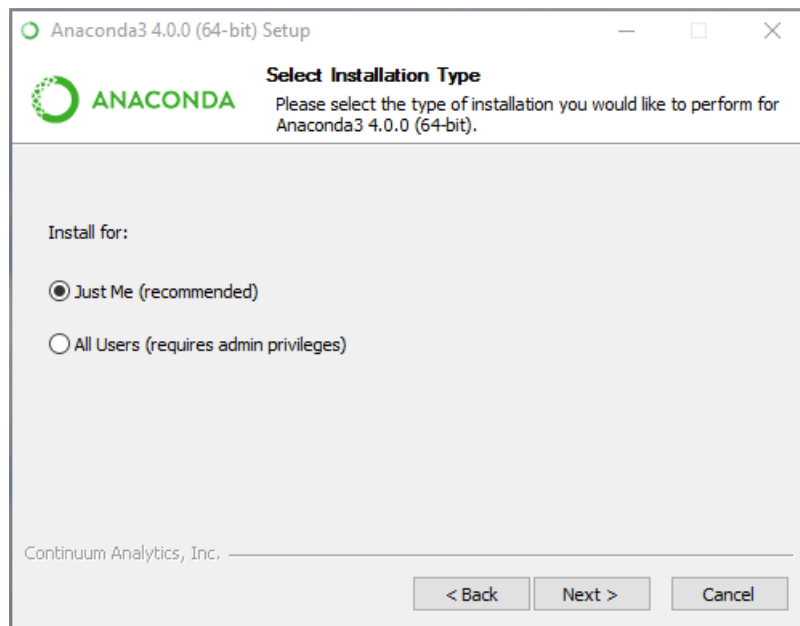


Figure 3: Installation type

Thirdly, after following each upcoming instruction on the screen, the installer was installing Anaconda (Fig.4). Therefore, we were able to successfully install and use Anaconda for our project.

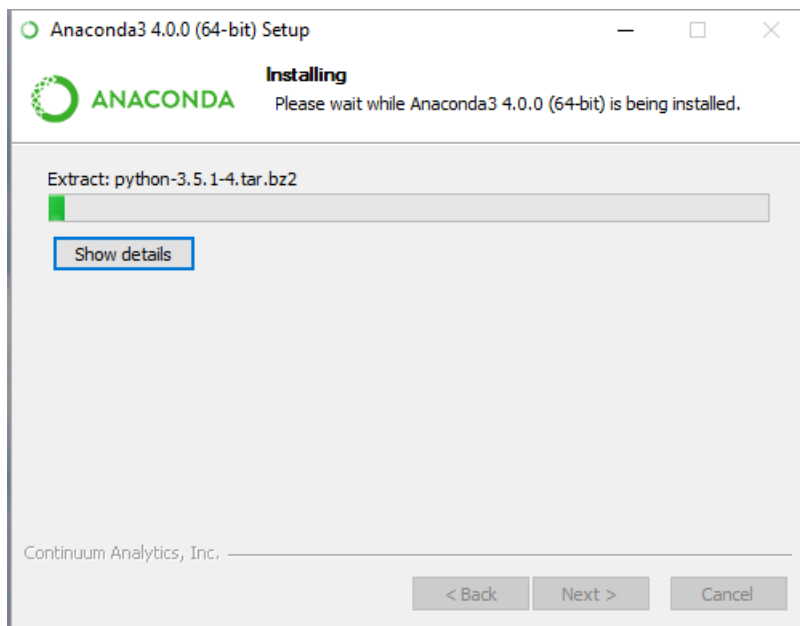


Figure 4: Installing process

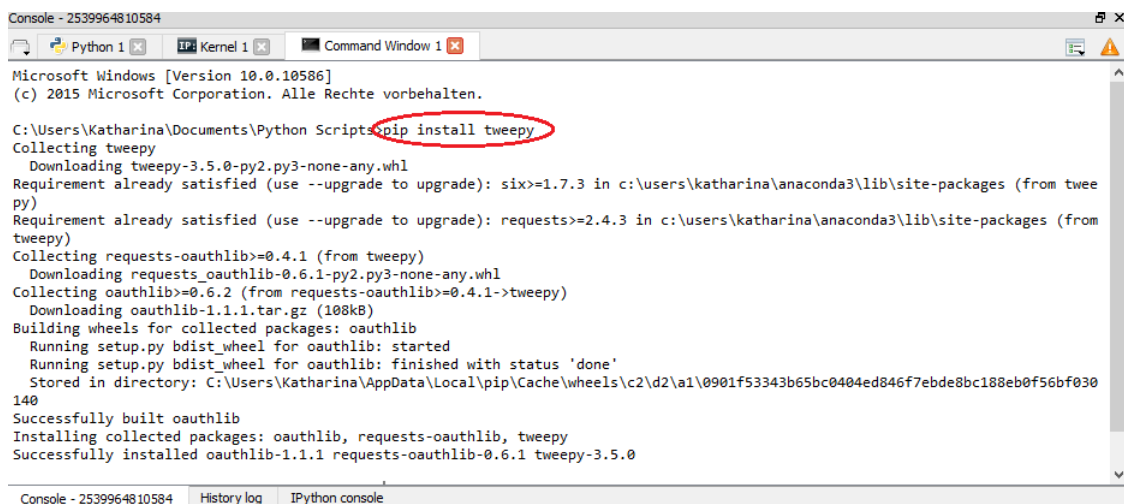
2.2 Python Packages

The installation of Anaconda automatically includes, among others, the packages *jsonschema* and *NLTK*. While *Jsonschema* offers an implementation of JSON Schema validation for Python, *NLTK* is a Natural Language Toolkit. We may use these Python packages (and further ones) for the second part of our project (analysis) and therefore refer to it in the second report. For the first part of our project, we only needed the "Tweepy" package to get connected to the Twitter API and retrieve the relevant data. Since this package is not pre-installed and cannot be installed using the "conda" command, we had to use the "pip" command (see below).

3 Twitter

3.1 Twitter API - Tweepy

In order to access the Twitter API, we additionally installed the Python package *Tweepy* using the "pip" command (Fig. 5).



```
Console - 2539964810584
Python 1
IP: Kernel 1
Command Window 1

Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Katharina\Documents\Python Scripts>pip install tweepy
Collecting tweepy
  Downloading tweepy-3.5.0-py2.py3-none-any.whl
Requirement already satisfied (use --upgrade to upgrade): six>=1.7.3 in c:\users\katharina\anaconda3\lib\site-packages (from tweepy)
Requirement already satisfied (use --upgrade to upgrade): requests>=2.4.3 in c:\users\katharina\anaconda3\lib\site-packages (from tweepy)
Collecting requests-oauthlib>=0.4.1 (from tweepy)
  Downloading requests_oauthlib-0.6.1-py2.py3-none-any.whl
Collecting oauthlib>=0.6.2 (from requests-oauthlib>=0.4.1->tweepy)
  Downloading oauthlib-1.1.1.tar.gz (108kB)
Building wheels for collected packages: oauthlib
  Running setup.py bdist_wheel for oauthlib: started
  Running setup.py bdist_wheel for oauthlib: finished with status 'done'
  Stored in directory: C:\Users\Katharina\AppData\Local\pip\Cache\wheels\c2\d2\1\0901f53343b65bc0404ed846f7ebde8bc188eb0f56bf030140
Successfully built oauthlib
Installing collected packages: oauthlib, requests-oauthlib, tweepy
Successfully installed oauthlib-1.1.1 requests-oauthlib-0.6.1 tweepy-3.5.0
```

Figure 5: Tweepy installation

The package provides access to the Twitter APIs out of a Python program. Worth to mention, Tweepy also supports OAuth authentication, as BasicAuth is no longer supported by the Twitter APIs. Both the Twitter Search API (in general not the firehose!) and the Streaming API can be accessed with Tweepy.

3.2 Twitter Developer Account

To use one of the Twitter APIs, we first needed to create a Twitter account on <http://www.twitter.com>. The sign up process included different steps, like entering information about name, email address and preferred password as well as choosing a username. The username, BigData Goethe, would be used every time we post a message or engage with someone on Twitter (Fig 6).

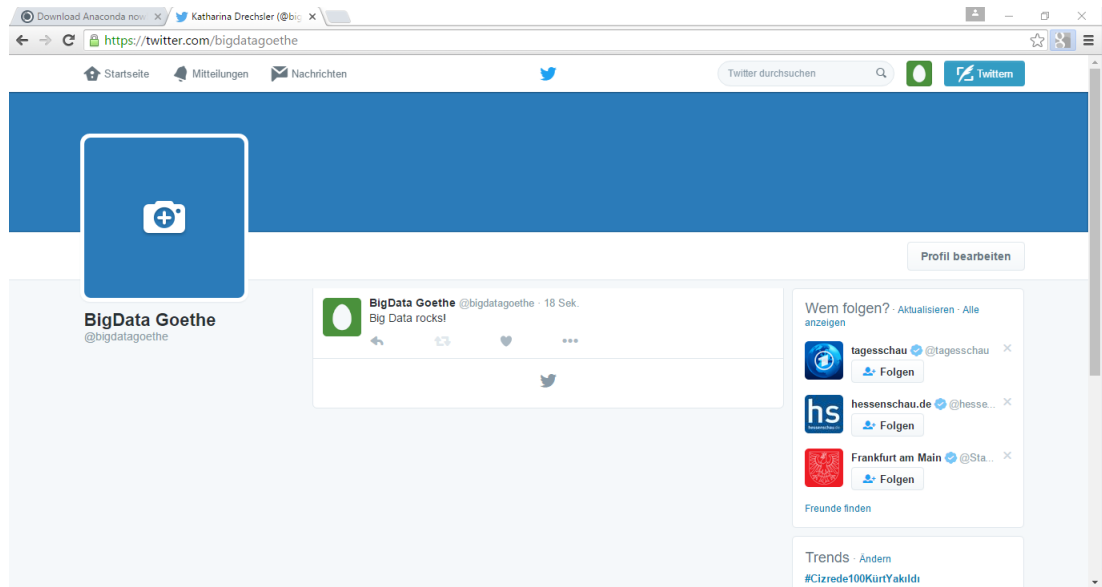


Figure 6: Twitter Account

Next, we visited <http://dev.twitter.com> and signed in with our Twitter credentials. Then, we created a new application and added information about the name, description and website (Fig. 7). The name of the application is presented to users when they are prompted to authorize our application to access their Twitter information. The website refers to the URL, which points the user back to our application, where they can download it or find out more information.

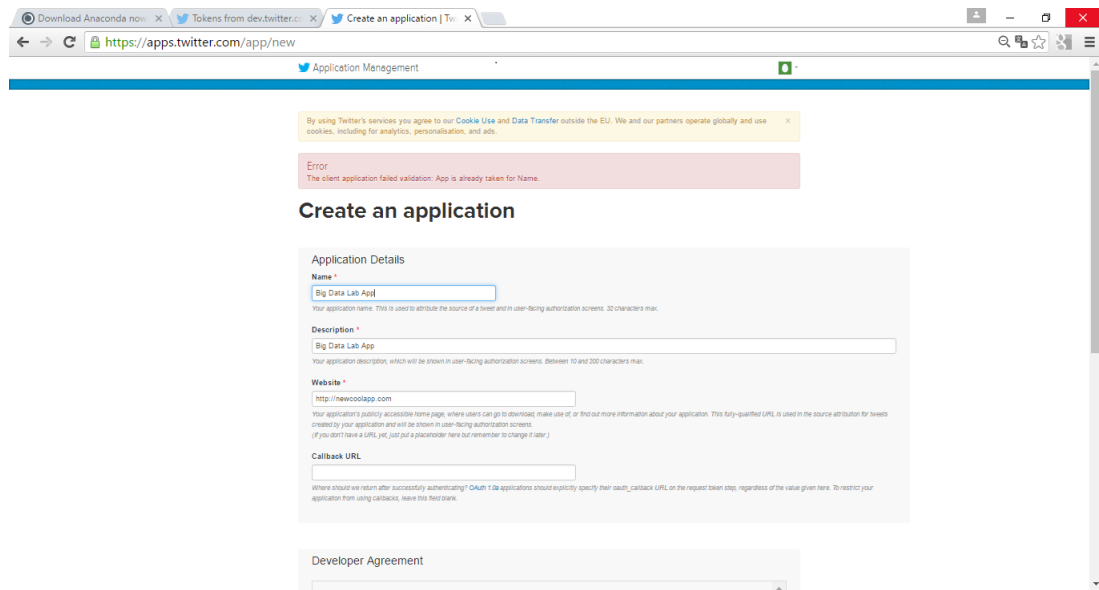


Figure 7: Twitter Application Creation

After following each upcoming instruction on the screen and completing the form, we specified our application settings. In the application settings we then accessed the tab "Key and Access Token". Here the consumer key and consumer secret as well as the access token and access token secret are provided. These values are the application's credentials, which are needed for almost all transactions with Twitter, including going through the OAuth authorization flow and working with the Twitter APIs. How we used these values in our Python program is described in the section "Data collection and Python code".

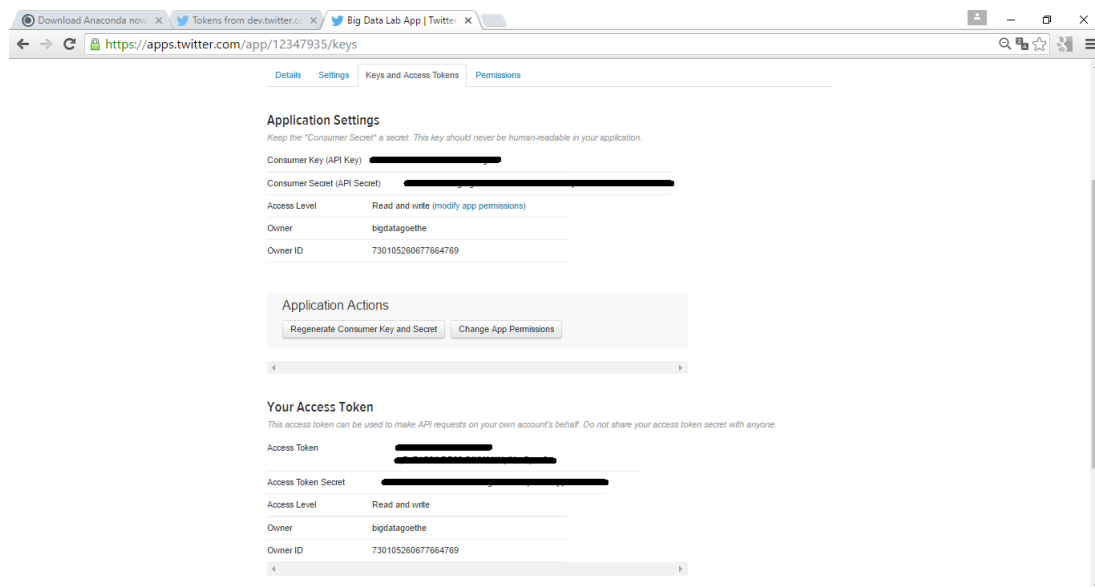


Figure 8: Twitter Application Settings

4 Server environment

We are using a Raspberry Pi 2 and an external storage device for the data collection platform. This hardware infrastructure offers us the possibility to collect the data 24 hours a day, 7 days a week.



Figure 9: Raspberry Pi

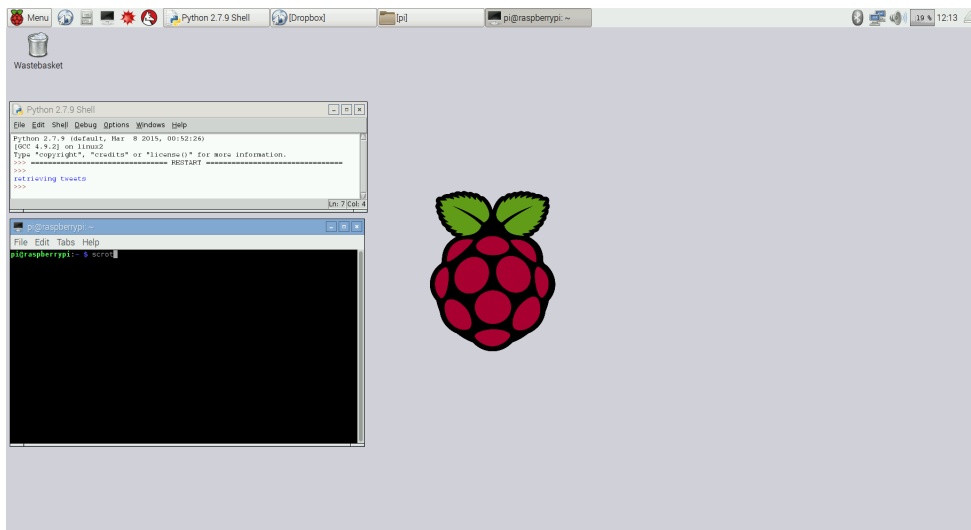


Figure 10: Raspberry Pi Screenshot

5 Data collection and Python code

In this section you will find the Python program, with which we are collecting the relevant data for our project. Please see the comments in the source code, which explain the function of each block of code.

```

1 # Necessary imports.
import tweepy

3

4 # The OAuth parameters.
5 consumer_key = "secret"
6 consumer_secret = "secret"
7 access_token = "secret"
8 access_token_secret = "secret"
9
10 # Overwrite the on_data method to specify the desired action.
11 class MyStreamListener(tweepy.StreamListener):
12     def on_data(self, data):
13         with open("../path/to/save/the/file.txt", "a") as fileobject:
14             fileobject.write(data)
15
16 # Authorize.
17 auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
18 auth.set_access_token(access_token, access_token_secret)
19 api = tweepy.API(auth)
20
21 # Build the necessary keyword list.
22 keyword_list = []
23 with open("../path/to/the/keywords.txt") as fileobject:
24     for line in fileobject:
25         keyword_list.append(line[:-1])
26
27 # Open the streaming api and listen.
28 myStreamListener = MyStreamListener()
29 myStream = tweepy.Stream(auth = api.auth, listener = myStreamListener())
30 myStream.filter(track = keyword_list, async = True, languages = ["en"])

```

This program is executed (also while writing this report) on our server infrastructure (see above) 24 hours a day and 7 days a week to get the most out of our available time to collect the data.

We started to collect smoking-related tweets on Friday, May 27 15:44:00. From Monday, May 30 onwards we collected tweets continuously. Before that, there were several interruptions. Until the current point in time (June 4, 5pm), we have collected almost 1 GB of tweets. As we

have extended our list of keywords since the start of the project, we expect to collect at least 4 GB of tweets by the end of June.

With the keywords we use, we try to retrieve tweets from smokers, people trying to quit smoking and companies advertising their cigarette brands. The diversity of smoking related tweets will allow us to conduct health related analytics of different kind in the second phase of the project.

6 Problems

This section provides a brief analysis of the problems we had in the first phase of our project "Streaming Data Collection and Deserialization on Twitter using Python". We did not face any technical issues in the setup process and the installation was performed without any problems or errors. However, to find a topic to focus on proved to be rather difficult. Here, the main obstacles were the necessary level of innovativeness of the solution, the short timeframe to execute the project as well as the planned use of open data as a data source (mainly Twitter).

After we had found our topic, we faced a problem concerning the restriction of both use cases to the US. Many of the gathered data don't carry a value for the geolocation of the post. Therefore, we can't filter out posts which are not coming from the US in the moment we gather them. Furthermore, we noticed that it is not possible to apply keyword filters and location filters simultaneously to tweets as "bounding boxes do not act as filters for other filter parameters"⁴. Therefore, we retrieve smoking-related tweets globally and then exclude non-US data in the data preparation step. In the second report there will be a detailed description of the data preparation and preprocessing steps we then have taken in the second phase of the project.

7 Conclusion

The aim of this first report was to provide a documentation of the process and problems we have had in the first phase of our project "Streaming Data Collection and Deserialization on Twitter using Python". Thus, we described the setup process, the data collection and the

⁴ Twitter Developers Documentation, Streaming API request parameters, available at: <https://dev.twitter.com/streaming/overview/request-parameters>, accessed on June 3rd.

problems we faced.

In the next step, we will further conduct our data collection and afterwards start with our analysis to work out both use cases. Thus, in our second report, we will document our experiences regarding the work on the two use cases and the final results of our project. The results will also be presented at the end of the semester.