Präsenz von Schlagworten in Twitter

Viele von uns verwenden Twitter täglich, um sich über die neuesten Ereignisse und das aktuelle Weltgeschehen zu informieren. Ich persönlich nutze Twitter hauptsächlich dazu, mein Wissen in der Softwarebranche aktuell zu halten und schnellstmöglich an neue Mitteilungen von Technik-Giganten zu gelangen. Darüber hinaus ist ein grosser Teil der Twitter-Nutzer selbst aktiv. Jeden Tag werden rund 500 Millionen Tweets versendet (Smith, 2019). Diese Tweets stellen eine umfangreiche Quelle dar, aus der Informationen zu fast jedem erdenklichen Thema geschöpft werden können. Anhand von diesen Informationen versuche ich herauszufinden, welche meiner beiden verwendeten Programmiersprachen Java und Python mehr Bühnenpräsenz im Web geniesst. Da die Zeit fehlt Terrabyte von Daten zu untersuchen, wird ein Twitter Stream aufgebaut, der über 48 Stunden in Echtzeit alle Tweets zu den Schlagworten Java und Python liefert. Diese Tweets werden danach analysiert, welche der beiden Programmiersprachen heutzutage die grössere Aufmerksamkeit geniesst. Meine Vermutung ist, dass mehr über Python getweetet wird.

Twitter bietet mit dem Twitter Streaming API einen Service an, mit dem es möglich ist, Tweets in Echtzeit zu filtern und zu erhalten. In welcher Form und Menge Daten empfangen werden, hängt von den jeweiligen Filtereinstellungen des Streams ab. Hierbei werden nicht durch eine wiederholte Aufforderung neue Daten geladen, sondern durch den einmalig geöffneten Stream neue Daten empfangen. Sobald es ein Tweet gibt, bei dem ein Schlagwort übereinstimmt, teilt der Server dies dem Client mit. Somit ist es möglich, mit nur einer Anfrage an den Server, mehrere Tausend Tweets als Antwort zu erhalten. Dies natürlich über einen gewissen Zeitraum hinweg (Twitter Inc., 2019).
Ein Stream kann auf mehrere Arten aufgebaut werden. Direkt über die Twitter Streaming API oder über ein Dritt-Bibliothek, die die Twitter Streaming API unterstützt. Eine sehr einfach zu verwendete Python Bibliothek ist Tweepy. Sie vereinfacht die Interaktion mit der Twitter Stream API durch die Kapslung der Requests enorm und liefert alle benötigten Werkzeuge direkt mit (Roesslein, 2019). Deshalb wurde die im Folgenden beschriebene Analyse mittels Tweepy durchgeführt.
Um eine Verbindung zum Twitter Stream herzustellen, wird zuerst eine Authentifizierung eingerichtet. Die Authentifizierung erfolgt über das OAuth 2.0 Protokoll (Parecki, 2019). Dazu muss bei Twitter die Applikation, die den Twitter Stream aufbaut, registriert werden. Das Snippet 1 zeigt auf, wie die Authentifizierung durchgeführt wird.

```
import tweepy

auth = OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_TOKEN_SECRET)
```

Snippet 1: Authentifizierung

Mittels der Authentifizierung und einem Streamlistener-Objekt kann ein Stream zur Twitter Streaming API erstellt werden. Im Streamlistener wird definiert, was mit den empfangenen Tweets passieren soll. Im Snippet 2 werden die empfangen Tweets in eine Textdatei geschrieben, um sie später für die Analyse wiederzuverwenden. Für eine längere Auswertung sollte die Textdatei durch eine geeignete Datenbank ersetzt werden. Da dies für eine 48 Stunden Analyse nicht notwendig ist, kann der Stream ohne eine Datenbank gestartet werden. Zusätzlich

ist es möglich beim Starten eines Sreams Filterargumente mitzugeben. Im nachfolgenden Snippet 2 ist ein Filter für die zwei Schlagworte Java und Python eingestellt.

```
tweets_file = open('/data/twitter_tweets.txt', 'w+')

class MyStreamListener(tweepy.StreamListener):

    def on_data(self, raw_data):
        tweets_file.write(raw_data)

    def on_error(self, status_code):
        # Does not stop the stream.
        return True

myStream = tweepy.Stream(auth, listener=MyStreamListener())
myStream.filter(track='java', 'python')
```

Snippet 2: Twitter Stream

Während der Erarbeitung stellte ich keine Grenzwerte für ein Abfragelimit fest. Jedoch wurde mein Stream einmalig für eine gewisse Zeit blockiert. Dies passierte während ich mit den Filterargumenten experimentierte. Hier muss beachtet werden, auf welche Schlagworte gefiltert wird. Heiss diskutierte Schlagworte können einen grossen Traffic verursachen, was wiederum zu einer Blockung des Streams führen kann. Die Twitter Streaming API gibt nur für Filterargumente Maximalwerte bekannt. Diese wären 400 Tracks, 25 Standorte und 5000 Benutzer (Twitter Inc., 2019). Bezüglich einer Limite des Endpunktes wird nichts veröffentlicht.

Die gesammelten Tweets werden für die Datenanalyse wieder aus der Textdatei gelesen und in einer Liste zwischengespeichert. Innert zwei Tagen konnten rund 30’000 Tweets gesammelt werden. Die Tweets werden anhand der in ihrer Textnachricht enthaltenen Schlagworte Java oder Python in unterschiedliche Stapel sortiert. Bei Tweets mit beiden Schlagworten im Text, befinden sich somit in beiden Stapeln. Durch das Python Paket Matplotlib wird die Aufteilung visualisiert. Die Abbildung 1 ist die daraus entstandene Grafik. Sie zeigt auf, dass beinahe ein Drittel mehr über Python getweetet wurde als über Java.

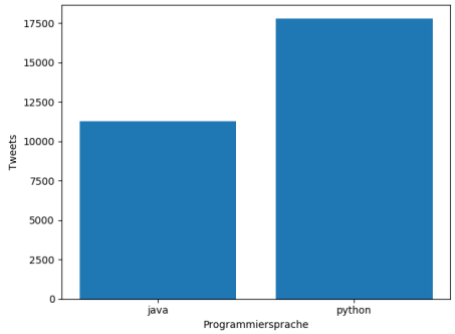

Abbildung 1: Anzahl Tweet pro Schlagwort

Nun stellt sich die Frage: «Warum wird mehr über Python getweetet als über Java?»
Um diese Frage zu beantworten, müssen die zwei Stapel von Tweets genauer untersucht werden. Dazu teilte ich die Textnachrichten der Tweets in einzelne Wörter auf und generierte über das Python Paket Wordcloud eine Grafik mit diesen Wörtern. Je grösser ein Wort im Bild erscheint, desto mehr wurde dieses in den Tweets verwendet. Die Abbildung 2 und zeigen dieAbbildung 3 generierten Grafiken. Es ist auf den ersten Blick klar, welche Grafik aus welchem Stapel generiert wurde, da die jeweilige Programmiersprache als grösstes Wort heraussticht.

Abbildung 2: Wordcloud Python-Tweets

Das über Python mehr diskutiert wird, könnte daran liegen, dass Python fast in jedem Teilgebiet der Informatik eingesetzt wird. Sei es für Analysen grosser Datenmengen, Entwicklung von künstlicher Intelligenz oder für Erstellung von Applikationen. Auch in der Webentwicklung kam Python in den letzten Jahren auf: Dank Frameworks, die zusammen mit JavaScript die Entwicklung im Web mittels Python stark vereinfachten und für jedermann ermöglichten. So enthält die Wordcloud Abbildung 2 Wörter wie Learning, Data, Machine-Learning, Javascript und AI (Artifizielle Interlligenz zu deutsch Künstliche Intelligenz). Aufgrund der Systemunabhängigkeit und einer einfachen Syntax ist Python eine beliebte Programmiersprache für Einsteiger. Diese Überlegungen werden von mehreren Technikblogs und Zeitschriften belegt, die bestätigen, dass Python eine universelle Allzweck-Programmiersprache ist. Dabei bleibt der Python-Code relativ einfach lesbar, was beim Einstieg ungemein hilft (Lang, 2018). Durch dieses breite Spektrum an Einsatzmöglichkeiten ergeben sich mehr Themengebiete über die diskutiert und getweetet werden kann als bei Java.
Ein kurzer Blick auf alle Hashtags bei den Python Tweets bejahen die oben aufgeführten Argumente. In den Top zehn Hashtags befinden sich erneut Schlagworte wie: #javascript, #MachineLearning, #AI und #DataScience. Die Tabelle 1 ist aus den Pythen-Tweets Hashtags erstellt worden und zeigt die Top zehn der verwendeten Hashtags auf.

Hashtag	Anzahl
python	2814
Python	1842
programming	918
javascript	676
reactjs	420
MachineLearning	383
angular	372
100DaysOfCode	366
DataScience	350
AI	350

Tabelle 1: Top zehn Python Tweets Hashtags

Dies soll nicht bedeutet, das Java eine unbeliebte Programmiersprache ist. Java ist heute noch die am meist verbreitete Programmiersprache (Richters, 2018). Java eignet sich jedoch nicht so gut für die Analyse von grossen Datenmengen und für die Entwicklung künstlicher Intelligenz, welche aktuell Topthemen in der Informatik-Branche sind (Fincher, 2018). Der Haupteinsatzort von Java ist in der Programmierung von Business Applikationen, die Daten verwalten oder in der App Entwicklung für Android-Smartphone. Durch das mächtige Framework Spring ist Java auch in der Webentwicklung vertreten. Die Wordcloud Abbildung 3 stimmt jedoch nur teilweise mit den obigen Erklärungen überein. Wörter wie Police und East stechen stärker heraus als das Framework Spring oder Android. Dies liegt daran, dass während der zweitägigen Analyse in East Java die Polizei weitere Verdächtige benannte, die an einem gewaltsamen Protest beteiligt waren (Boediwardhana, 2019).

Abbildung 3: Wordcloud Java-Tweets

Eine weitere Wordcloud Abbildung 4, ohne die Wörter die einen direkten Zusammenhang mit dem Vorfall in East Java zu tun haben, ergeben ein erwartetes Bild. So treten zum Beispiel das Framework Spring als auch Android stärker hervor. Eine interessante Entdeckung ist, dass Minecraft nun in der Wordcloud Abbildung 4 hervorsticht. Dies ist insofern keine Überraschung da, Minecraft ein Java geschriebenes Computerspiel ist.

Abbildung 4: Wordcloud Java-Tweets überarbeitet

Die Analyse hat meine Vermutung bestätigt, dass Python zurzeit mehr Beachtung geschenkt wird. Dies liegt hauptsächlich daran, dass der Einsatz Ort von Python die aktuellen Topthemen der Informatik abdeckt und zugleich für Neueinsteiger in die Programmierung ideal geeignet ist. Die Datenanalyse an sich ist sehr interessant und mächtig. Durch eine relativ kurze Datenanalyse von zwei Tagen konnten sogar Rückschlüsse auf das aktuelle Weltgeschehen am anderen Ende der Welt gemacht werden. Die aufgezeigte Analyse kann beliebig auf zwei andere Schlagworte angewendet werden, so können anderen Themen auf einfache Art und Weise auf ihre Bühnenpräsenz im Web untersucht werden. Wird die Analyse gar über eine längere Zeit durchgeführt kann dies noch detaillierte Erkenntnisse liefern.
Schon meine kleine Datenanalyse hat aufgezeigt, dass unscheinbar wirkende Tweets sehr viel mehr Informationen liefern, als auf den ersten Blick ersichtlich.

Literaturverzeichnis

Boediwardhana, W., 2019. *The Jakarta Post.* [Online]
Available at: https://www.thejakartapost.com/news/2019/08/31/east-java-police-name-another-suspect-related-to-racist-abuse-against-papuan-students.html
[Zugriff am 04 09 2019].
Fincher, M., 2018. *Hackernoon.* [Online]
Available at: https://hackernoon.com/ai-and-big-data-two-major-parts-of-the-digital-future-2f9c7c5e813a
[Zugriff am 04 09 2019].
John Hunter, D. D. E. F. M. D., 2019. *Matplotlib.* [Online]
Available at: https://matplotlib.org/3.1.1/tutorials/introductory/pyplot.html
[Zugriff am 04 09 2019].
Lang, M., 2018. *Heise.* [Online]
Available at: https://www.heise.de/tipps-tricks/Programmieren-fuer-Anfaenger-Welche-Programmiersprache-eignet-sich-am-besten-3974883.html#anchor_2
[Zugriff am 03 09 2019].
Parecki, A., 2019. *OAuth 2.0.* [Online]
Available at: https://oauth.net/2/
[Zugriff am 03 09 2019].
Pivotal Software, 2019. *Spring.* [Online]
Available at: https://spring.io/
[Zugriff am 04 09 2019].
Python Software Foundation, 2019. *Pypi.* [Online]
Available at: https://pypi.org/project/wordcloud/
[Zugriff am 04 09 2019].
Richters, M., 2018. *Jaxenter.* [Online]
Available at: https://jaxenter.de/redmonk-pypl-tiobe-rankings-vergleich-69108
[Zugriff am 04 09 2019].
Roesslein, J., 2019. *Tweepy.* [Online]
Available at: https://tweepy.readthedocs.io/en/latest/streaming_how_to.html
[Zugriff am 03 09 2019].
Smith, K., 2019. *Brandwatch.* [Online]
Available at: https://www.brandwatch.com/de/blog/twitter-statistiken/
[Zugriff am 03 09 2019].
Twitter Inc., 2019. *Twitter API.* [Online]
Available at: https://developer.twitter.com/en/docs/tweets/filter-realtime/overview
[Zugriff am 03 09 2019].

Skript für die Tweets-Analyse

```
import json
import re
import matplotlib.pyplot as pyplot

from collections import Counter
from prettytable import PrettyTable
from wordcloud import WordCloud

class WordCloudWrapper():
    # Wörter die nicht verwendet werden.
    stopwords = ['RT', 're', 'https', 'http', 'gt', 'Retweet', 'suspect', 'incitement', 'police', 'breaking',
                 'separatism', 'spreading', 'East', 'crime', 'davidlipson', 'co']

    def create_word_cloud(self, words):
        text = (' ').join(words)

        # Grösse und Style definieren.
        wordcloud = WordCloud(width=720, height=720, margin=0, collocations=False, background_color='White')

        # Wörter filtern.
        for word in self.stopwords:
            wordcloud.stopwords.add(word)

        wordcloud.generate(text)

        # Wordcloud-Bild anzeigen.
        pyplot.imshow(wordcloud, interpolation='bilinear')
        pyplot.axis("off")
        pyplot.margins(x=0, y=0)
        pyplot.show()

class FileReader():
    tweets_data = []
    tweets_file = open('./tweets.txt', 'r')

    # Tweets aus der Datei lesen.
    def get_received_tweets(self):
        for line in self.tweets_file:
            try:
                tweet = json.loads(line)
```

```
            self.tweets_data.append(tweet)
        except:
            continue

    print('Count of Tweets: ', len(self.tweets_data))
    return self.tweets_data

class DataAnalyser():
    tweets = []
    # Dictionary für Schlagwort und Anzahl Tweets.
    tweets_count = {}

    def __init__(self, tweets):
        self.tweets = tweets

    # Text, Wörter aus dem Text, Hashtags und Land aus den Tweets entehmen und Tabelle der Top 10 erstellen.
    def analyse_tweets(self):
        tweets_text = self.get_text(self.tweets)
        tweets_hashtags = self.get_hashtags(self.tweets)
        tweets_words = self.get_words(tweets_text)
        tweets_counties = self.get_country(self.tweets)

        self.create_table(tweets_words, tweets_hashtags, tweets_counties)

    # Text, Wörter aus dem Text, Hashtags und Land aus den Tweets entehmen und Tabelle der Top 10 erstellen.
    # Werden nur soche Tweets beachtet in dem das 'word' im Text vorkommt.
    def analyse_tweets_for_word(self, word):
        tweets_for_specific_word = []
        for tweet in self.tweets:
            if self.word_in_text(word, tweet['text']):
                tweets_for_specific_word.append(tweet)

        tweets_text = self.get_text(tweets_for_specific_word)
        tweets_hashtags = self.get_hashtags(tweets_for_specific_word)
        tweets_words = self.get_words(tweets_text)
        tweets_county = self.get_country(tweets_for_specific_word)

        print(word, len(tweets_text))
        self.create_table(tweets_words, tweets_hashtags, tweets_county)
        self.tweets_count[word] = len(tweets_text)

        return tweets_words
```

```
# Text aus einem Tweet entnehmen.
def get_text(self, tweets):
    return [tweet['text'] for tweet in tweets]

# Hashtags aus einem Tweet entnehmen.
def get_hashtags(self, tweets):
    return [hashtag['text'] for tweet in tweets for hashtag in tweet['entities']['hashtags']]

# Wörter aus einem Tweet-Text entnehmen.
def get_words(self, text):
    return [word for tweet in text for word in tweet.split()]

# Land aus einem Tweet entnehmen falls vorhanden.
def get_country(self, tweets):
    countries = []
    for tweet in tweets:
        if tweet['place'] != None:
            countries.append(tweet['place']['country'])
        else:
            None
    return countries

# Tabelle für die Top 10 Wörter, Hashtags und Länder erstellen.
def create_table(self, words, hashtags, countries):
    for label, data in (('Wort', words), ('Hashtag', hashtags), ('Land', countries)):
        table = PrettyTable(field_names=[label, 'Anzahl'])
        c = Counter(data)
        [table.add_row(item) for item in c.most_common()[:10]]
        table.align[label], table.align['Anzahl'] = 'l', 'r'
        print(table)

# Grafik generieren für den Tweet-Count.
def create_tweets_count_diagramm(self):
    if not self.tweets_count:
        return

    pyplot.bar(range(len(self.tweets_count)), list(self.tweets_count.values()), align='center')
    pyplot.xticks(range(len(self.tweets_count)), list(self.tweets_count.keys()))
    pyplot.xlabel('Programmiersprache')
    pyplot.ylabel('Tweets')
    pyplot.show()
```

```
    # Ist ein Wort in einem Text vorhanden liefert die Methode 'True' zurück, ansonsten 'Flase'.
    def word_in_text(self, word, text):
        word = word.lower()
        text = text.lower()
        all_words = re.findall(r'\w+', text)
        if word in all_words:
            return True
        return False

if __name__ == '__main__':
    # Tweets aus Dateilesen.
    tweets_data = FileReader().get_received_tweets()

    data_analyser = DataAnalyser(tweets_data)
    # Alle Tweets analysieren.
    data_analyser.analyse_tweets()
    # Tweets für ein Schlagwort analysieren.
    tweets_words_for_java = data_analyser.analyse_tweets_for_word('java')
    tweets_words_for_python = data_analyser.analyse_tweets_for_word('python')

    # Diagramm für Wort-Tweet-Analyse erstellen.
    data_analyser.create_tweets_count_diagramm()

    # Wordcloud erstellen.
    wc_wrapper = WordCloudWrapper()
    wc_wrapper.create_word_cloud(tweets_words_for_java)
    wc_wrapper.create_word_cloud(tweets_words_for_python)
```

Snippet 3: Tweets-Analyse