Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig

Mitglied der SUPSI

4.Semesterarbeit

Von Twitter-Trends über KMeans zu GoogleMaps

EDS, Einführung in Data Science, INF-P-IN010, BE2, HS20/21, Dr. Paoli Beatrice

Inhaltsverzeichnis

1. Abstract

Mit Tweepy werden Trends von Twitter abgerufen. Die Geocodierung zu den angegebenen Locations werden durch die GoogleMaps API abgefragt. Diese können anschliessend mit den KMeans zu Clustern zusammengefasst werden. Damit wird eine KML-Datei erstellt und via GoogleMaps visualisiert.

2. Einleitung

In dieser Arbeit werden trendige Twitter Topics abgefragt und durch Ermittlung des Entstehungsorts eine Landkarte dargestellt. Welche Erkenntnisse können daraus gezogen werden?

3. Material

Um Twitter Trends abzurufen wird ein Twitter Developer Account [2] benötigt. Die Geodaten werden über die GoogleMaps API verarbeitet, hierzu ist ebenfalls ein Google Developer Account [3] nötig. Für die Google API muss eine Kreditkarte hinterlegt werden. Bei der Benutzung zu Testzwecken sollten keine Kosten anfallen, die Verrechnung wird aufgrund der eingegangenen Requests erstellt. Programmiert wurde in Python [6] mit der IDE PyCharm [7]. Für die Twitterabfragen wurde Tweepy [1] genutzt.

```
pip3 install tweepy
pip3 install pandas
pip3 install simplekml
```

4. Methoden

Twitter:
Um Requests tätigen zu können wird eine bei Twitter registrierte Applikation benötigt. Diese wird mit einem Twitter Developer Account [2] erstellt. Mit dem API Key und Secret wird es ermöglicht als registrierter Benutzer auf die Twitter API zuzugreifen. In dieser Arbeit wird dazu Tweepy verwendet.

Tweepy [1]:
Der Vorgang zur Autorisierung mit OAuth 1a ist ein hin und her:

1. Benötigt man ein Request Token von Twitter
2. Redirect zu twitter.com um die Applikation zu autorisieren
3. Austausch des Request Token mit dem Access Token

Um Abfragen aufgrund der begrenzten Requestrate abbrechen zu lassen empfiehlt es sich den Parameter «wait_on_rate_limit=True» mitzugeben. Die Abfrage wartet somit auf ein allenfalls aufgebrauchtes Limit. Wartezeit beträgt 15 Minuten für weitere Requests.

Dies könnte so aussehen:

```
def tweepy_authenticate(self):
    auth = tweepy.OAuthHandler(self.consumer_key,
self.consumer_secret, self.callback_uri)
    redirect_url = auth.get_authorization_url()
    print(redirect_url)
    user_print_input = input("What's the pin value? ")
    auth.get_access_token(user_print_input)
    t_api = tweepy.API(auth, wait_on_rate_limit=True)
    return t_api
```

Trends [1]:
trends_available()
Bietet eine Möglichkeit alle Locations die Trends beinhalten abzufragen. Wichtige Informationen zur Weiterverarbeitung ist die WOEID «Where on Earth ID» ehemals von Yahoo! betreitgestellt.

Beispiel Ausgabe:

```
{'name': 'Zurich',
  'placeType': {'code': 7, 'name': 'Town'},
  'url': 'http://where.yahooapis.com/v1/place/784794',
  'parentid': 23424957,
  'country': 'Switzerland',
  'woeid': 784794,
  'countryCode': 'CH'}
```

trends_place(place['woeid'])
Aufgrund der WOEID kann nach Trends in gegebener Location gesucht werden. Mit dem Namen können anschliessend Tweets gesucht werden, die unter diesen Trend laufen.

Beispiel Ausgabe:

```
{'name': '#EhefürAlle',
  'url': 'http://twitter.com/search?q=%23Ehef%C3%BCrAlle',
  'promoted_content': None,
  'query': '%23Ehef%C3%BCrAlle',
  'tweet_volume': None}
```

Tweets [1]:
search(user_print_input)
Die Suche nach Tweets benötigt ein String nachdem gesucht wird der bis zu 500 Zeichen lang sein kann. Interessante Values für die weitere Verarbeitung zu Geolocations sind «coordinates» und «place». Leider sind diese standardmässig deaktiviert und können nur durch den User eingeschaltet werden.

Beispiel Ausgabe (Nach Formatierung):

```
{'user': 'HugentoblerF',
  'coordinates': None,
  'place': None,
  'id': 1334055997678231555,
  'id_str': '1334055997678231555',
  'text': '@Blickch Es geht auch darum, Spital Betten frei zu halten.
\nSolange die #REGA die verletzten Skisportler im Minuten-… https://t
.co/KXKWVf28He',
  'source': 'Twitter for iPhone',
  'source_url': 'http://twitter.com/download/iphone',
  'in_reply_to_status_id': 1334048250857476096,
  'in_reply_to_status_id_str': '1334048250857476096',
  'in_reply_to_user_id': 22781017,
  'in_reply_to_user_id_str': '22781017',
  'in_reply_to_screen_name': 'Blickch',
  'retweet_count': 0,
  'favorite_count': 1,
  'lang': 'de'}
```

Google:
endpoint = f"https://maps.googleapis.com/maps/api/geocode/{self.data_type}"
params = {"address": loc_query, "key": self.g_api_key}
Einem GoogleMapsAPI Request liegt ein URL zugrunde in der auch der API.Key metgeschickt werden muss. Hierzu wird ein GoogleMaps Developer Account benötigt. Die erstellte Applikation soll mit der Geocoding API [] ausgestattet werden.

Beispiel URL:

```
https://maps.googleapis.com/maps/api/geocode/json?address=1600+Amphit
heatre+Parkway,+Mountain+View,+CA&key=YOUR_API_KEY
```

Bei der Erstellung dieser URL in Python hilft die Library «urllib» [] verwendet wurde im Speziellen urllib.parse. Dies vereinfachert die codierung eines http Request.

Beispiel Endpoint:

```
endpoint =
f"https://maps.googleapis.com/maps/api/geocode/{self.data_type}"
params = {"address": loc_query, "key": self.g_api_key}
url_params = urlencode(params)
url = f"{endpoint}?{url_params}"
```

KML [5]:
Wird verwendet, um Koordinaten und Topics auf Google Maps ersichtlich zu machen.

Datenbereitstellung:
Um die Respone richtig verarbeiten zu können, müssen die Daten in brauchbarer weise zurecht gemacht werden. Hierfür werden diverse Variationen von Dict, List und Klassen (tweepy.models.SearchResults) zurückgegeben. Hier ist höchste Vorsicht geboten beim Umgang mit diesen Daten.

```
[{'Switzerland': [{'trends': [{'name': '#2020Wrapped',
       'url': 'http://twitter.com/search?q=%232020Wrapped',
       'promoted_content': None,
       'query': '%232020Wrapped',
       'tweet_volume': 400879},
      {…Weiter Einträge…}
      {'name': 'Earth',
       'url': 'http://twitter.com/search?q=Earth',
       'promoted_content': None,
       'query': 'Earth',
       'tweet_volume': 158614}],
     'as_of': '2020-12-02T09:49:37Z',
     'created_at': '2020-11-07T17:32:39Z',
     'locations': [{'name': 'Switzerland', 'woeid': 23424957}]}]}]
```

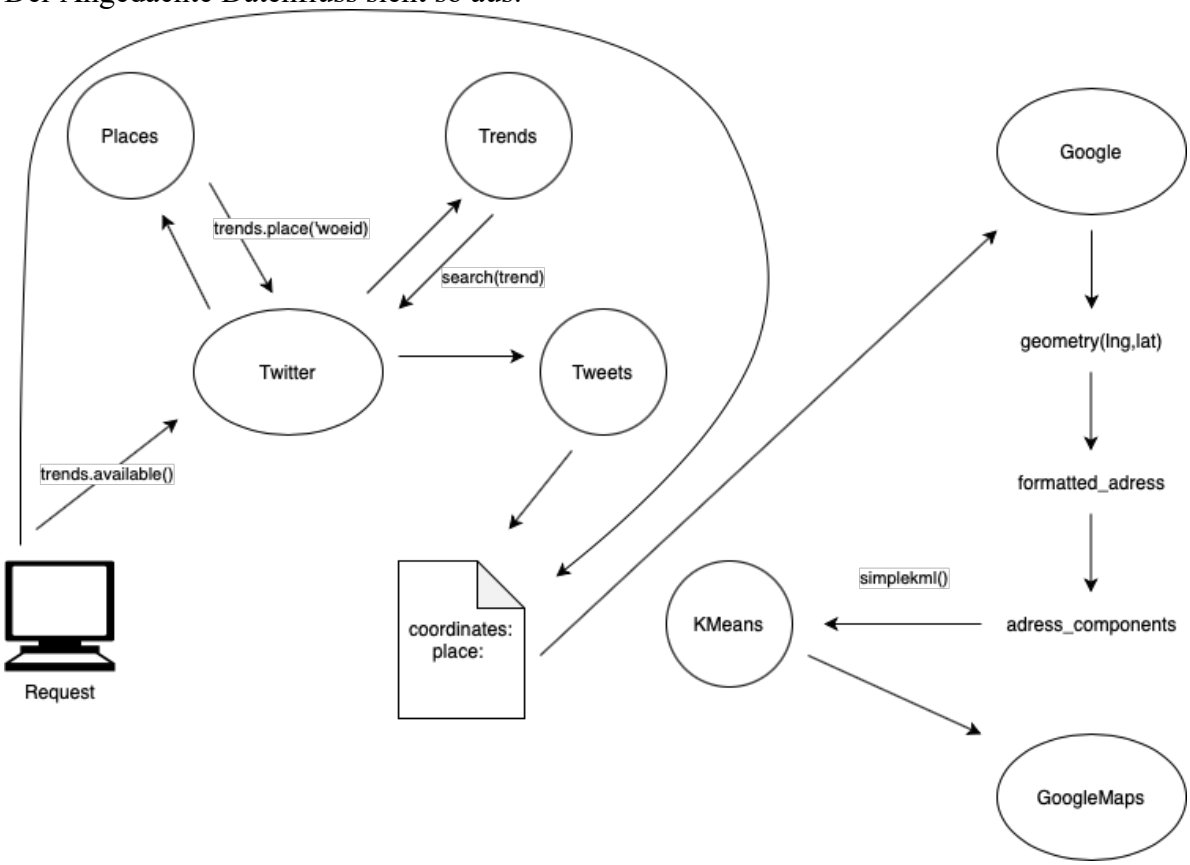
Um dies zu verdeutlichen, hier eine Zugriffsdarstellung:

```
print(type(t_client.trends_per_place))
print(len(t_client.trends_per_place))

print(type(t_client.trends_per_place[0]))
print(t_client.trends_per_place[0].keys())
print(len(t_client.trends_per_place[0]))

print(type(t_client.trends_per_place[0]['Switzerland']))
print(len(t_client.trends_per_place[0]['Switzerland']))

print(type(t_client.trends_per_place[0]['Switzerland'][0]))
print(t_client.trends_per_place[0]['Switzerland'][0].keys())
print(len(t_client.trends_per_place[0]['Switzerland'][0]))

print(type(t_client.trends_per_place[0]['Switzerland'][0]['trends']))
print(len(t_client.trends_per_place[0]['Switzerland'][0]['trends']))

print(type(t_client.trends_per_place[0]['Switzerland'][0]['trends'][0]))
print(t_client.trends_per_place[0]['Switzerland'][0]['trends'][0].keys())
print(t_client.trends_per_place[0]['Switzerland'][0]['trends'][0]['name'])

Output:
<class 'list'>
1
<class 'dict'>
dict_keys(['Switzerland'])
1
<class 'list'>
1
<class 'dict'>
dict_keys(['trends', 'as_of', 'created_at', 'locations'])
4
<class 'list'>
50
<class 'dict'>
dict_keys(['name', 'url', 'promoted_content', 'query', 'tweet_volume'])
#2020Wrapped
```

5. Resultate

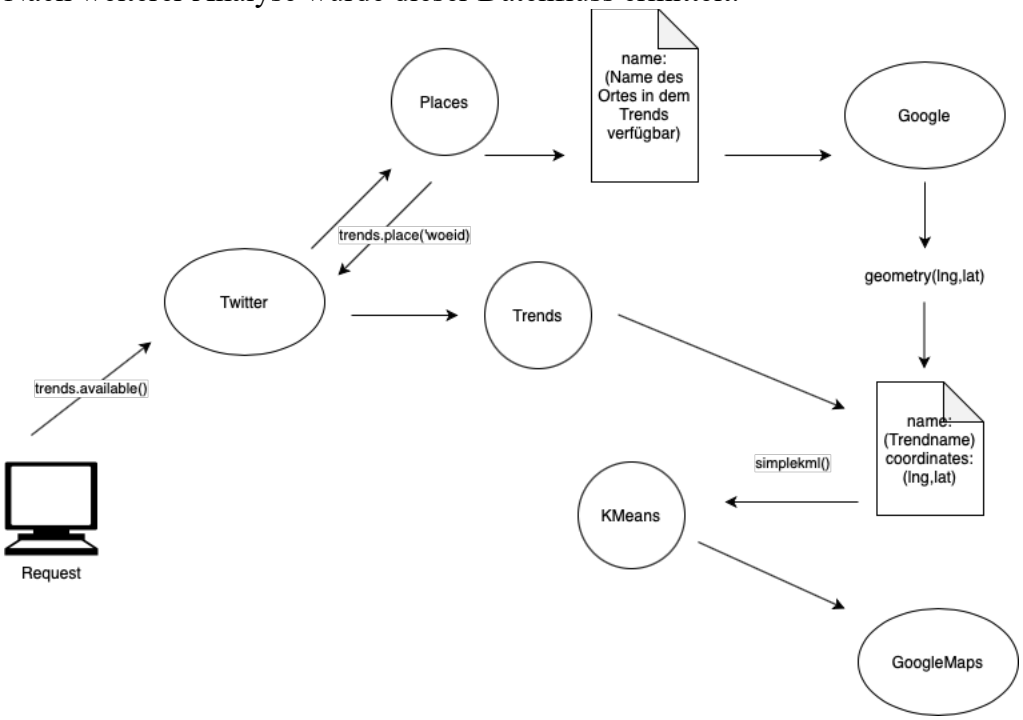
Twitter Topics können einfach abgerufen werden und weiterverarbeitet in Tweets oder zu Geolokalisierungszwecken verwendet werden. Der Umgang mit verschiedenen und ineinander verschachtelten Datentypen bereitet einen extrem erhöhten Aufwand, der nicht unterschätzt werden darf. Richtige Resultate konnten aus dieser Arbeit nicht gezogen werden. Der Zeitliche Aspekt hat ein Strich durch die Arbeit gezogen. Weiteres dazu in der Diskussion.

Der Angedachte Datenfluss sieht so aus:



Nach einiger Zeit wurde festgestellt, dass die Values in «coordinates:» und «place:» meistens leer sind. Die Standardeinstellung der Twitterapp lässt den Standort nicht mitschicken. Dies führt dazu, dass die Koordinaten anderweitig besorgt werden mussten.

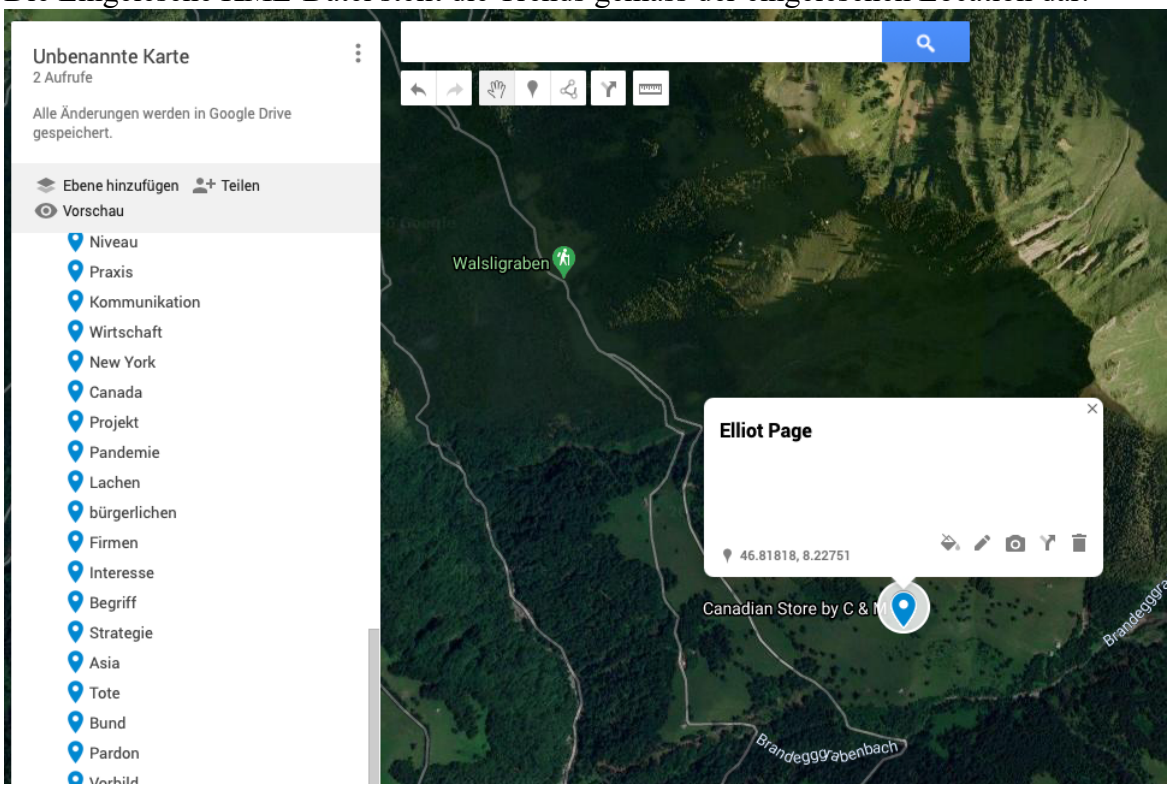
Nach weiterer Analyse wurde dieser Datenfluss ermittelt:



Die Abfrage nach aktuellen Trends geht allem voran. Zurück werden alle Ortsnamen inklusive woeid gegeben. Mit Hilfe der Ortsnamen kann über die Google API eine Anfrage gestellt werden, um die Geocodierung zu erhalten. Mit der woeid werden via Twitter die jeweiligen Trends zum Ortsnamen abgerufen. Anschliessend werden diese zusammengefügt und durch den KMeans Algorithmus geclustert und via KML-Datei in GoogleMaps sichtbar gemacht.

Erstellt wurden ein TwitterClient und ein GoogleMapsClient um die Anfragen an einem Punkt zu managen. Details dazu im Anhang – Programmcode.

Die Eingelesene KML-Datei stellt die Trends gemäss der eingelesenen Location dar.



Es wurde aus Zeitgründen lediglich eine Location ausgewertet. Alle Trends, die unter der woeid für Switzerland gefunden wurden, werden mit jeweiliger Geocodierung versehen und als KML-Datei in GoogleMaps [8] geladen. Für Auswertungen müssen weitere Trends und Locations hinzugefügt werden.

6. Diskussion

In der Fragestellen ist nicht genau geklärt, wie die Trends geclustert werden sollen. Dies sollte vor Beginn der Erarbeitung klar sein. Dies wurde bei dieser Arbeit leider vernachlässigt und machte sich spätestens beim Wechsel des Datenflusses bemerkbar, was die Zeitmanagement komplett durcheinanderbrachte.
Was ist der sinnvollste Weg verschiedene Trends zu clustern wenn nicht an die Geocodierung der beinhalteten Tweet zugegriffen werden kann? Ein Versuch von mehreren Locations die Trends abzurufen endete sehr schnell im Request Limit und zog alles in die Länge. Den Trends können so lediglich ein und dieselbe Geocodierung zugewiesen werden. Über die sinnvollste Herangehensweise sollte genauer nachgedacht werden.

7. Reflexion

Wo lagen die Probleme?

Analyse und Zielsetzung:
Allem voran kommt das Zeitmanagement. Es wurden zu wenig klare Ziele definiert, um an die Aufgabe effizient herangehen zu können. So ging viel Zeit damit verloren vor und zurück zu programmieren. Die Fragestellungen wurden im Vorfeld ungenügend analysiert, um daraus Ziele generieren zu können. Anfangs wurde deshalb ins «Blaue» gearbeitet und es musste festgestellt werden, dass der Ansatz nicht zum gewünschten Ziel führt.

Handling von Datentypen:
Erschwerend musste festgestellt werden, dass das Handling mit verschiedenen Datentypen grosse Probleme zu bereiten scheint. Im Nachgang zu dieser Arbeit müssen die Zugriffe auf die gängigsten Datentypen genauer studiert werden, um später nicht nochmal in die gleichen Probleme zu geraten. Das Handling mit verschiedenen Datentypen hat sehr grosse Probleme und Zeitverluste verursacht. Die Responses aus der Twitter API sind teilweise dict, dann wieder list und «SearchResults». Dies ist teilweise sehr verwirrend und unübersichtlich. Es mussten immer wieder Anpassungen vorgenommen werden da es Probleme mit den Abfragen gab.

8. Quellen

[1]	Tweepy Doc http://docs.tweepy.org/en/latest/# Auth: http://docs.tweepy.org/en/latest/auth_tutorial.html Trends: http://docs.tweepy.org/en/latest/api.html?highlight=trends#trends-methods Search: http://docs.tweepy.org/en/latest/api.html?highlight=trends#search-methods
[2]	Twitter Developer https://developer.twitter.com/en/portal/dashboard
[3]	GoogleDeveloper https://developers.google.com https://cloud.google.com/maps-platform Geocoding API https://developers.google.com/maps/documentation/geocoding/start?hl=de
[4]	Urllib https://docs.python.org/3/library/urllib.html urllib.parse https://docs.python.org/3/library/urllib.parse.html#module-urllib.parse
[5]	SimpleKML https://simplekml.readthedocs.io/en/latest/ kml upload to google maps: https://www.youtube.com/watch?v=1HqQuHeGa38
[6]	Python https://www.python.org
[7]	PyCharm https://www.jetbrains.com/pycharm/
[8]	GoogleMaps https://www.google.ch/maps/

9. Anhang

- Programmcode GitHub:
  https://git.ffhs.ch/christoph.grossmann/eds_4semesterarbeit.git