Fachhochschule Kiel
Hochschule für Angewandte Wissenschaften

IB.SH
Ihre Förderbank

Bewertung von Stadtquartieren über Soziale Medien

JULIAN BERGER & CHRISTINA HÜBERS

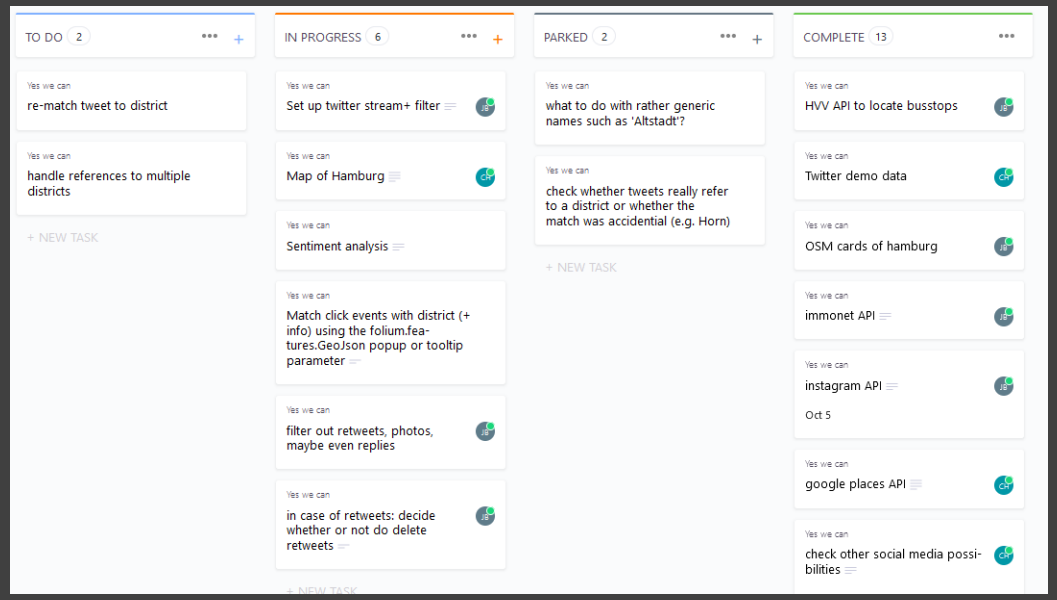
21.12.2021

Gliederung

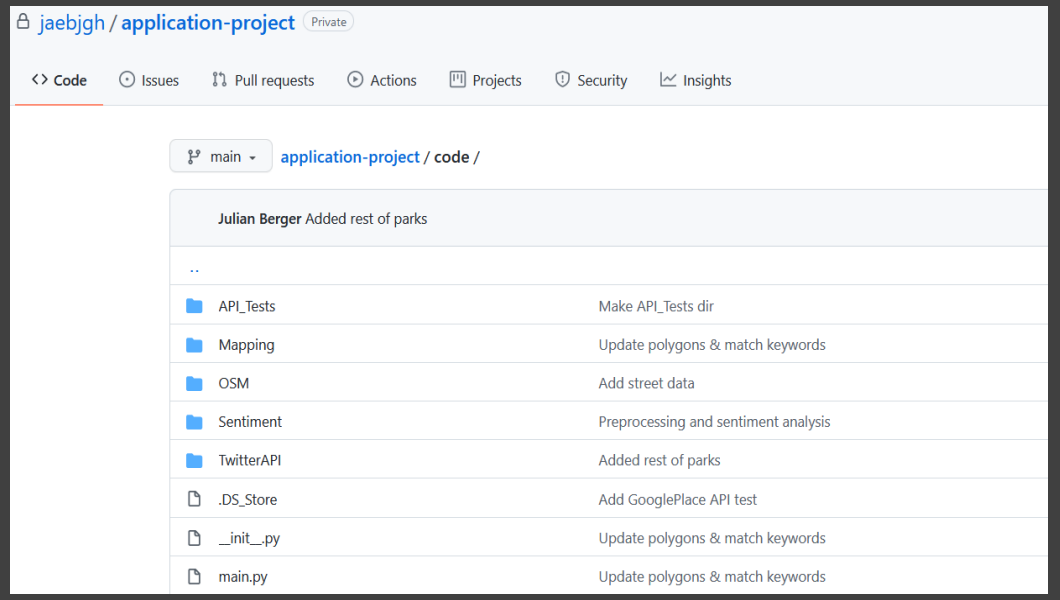
- Tools & Workflow
- Warum Twitter?
  - .. because it‘s all there is
  - Tweepy
- Tech-Stack
  - *Daten sammeln.* Tweepy Stream auf VM
  - *Datenbank.* MongoDB (via pymongo)
  - *Stimmungsanalyse.* BERT. Bidirectional Encoder Representations from Transformers.
  - *Visualisierung*. Leaflet (via Folium)
- Ergebnisse
- Herausforderungen

Tools

Projektplanung: clickup.com



Versionskontrolle: github.com



Workflow

WÖCHENTLICH

- Wöchentliche Teammeetings
  - Anfangs stark mit *clickup* gearbeitet
    → Aufgabenverteilung für den nächsten „Sprint“
  - Später Zweiteilung in
    1. Datenbeschaffung/Stimmungsanalyse
    2. Visualisierung
       → informellere Absprache

ZWEIWÖCHENTLICH

- Meeting mit Dozenten
- Meeting mit Vertretern der IB.SH:
  Dr. Jochen Heimann & Hilmar Müller-Teut

Warum Twitter?

.. because it‘s all there is

Es gibt viele Social Media Plattformen mit einer API, aber …

Instagram:

- Schwieriger Bewerbungsprozess

Tumblr:

- Sehr wenig Inhalt

Pinterest:

- Verweist meistens auf Inhalt von anderen Seiten

Facebook:

- Eingeschränkter Zugang zu Gruppen und Seiten

Twitter:

- Uneingeschränkter Zugang zu Tweets
- Große Anzahl von Tweets
- Einfacher Zugang zu Developer Account
- Es gibt ein Python Package für die API

Warum Twitter?

Tweepy

- Python Paket, um auf die Twitter API zuzugreifen
- Benötigt einen Twitter Developer Account mit Authentifizierungstoken
- Stream-Klasse gibt Tweets zurück, die bestimmte Anforderungen erfüllen, wie:
  - Sprache (z. B. Deutsch)
  - Keywords (z. B. Bergedorf, Altstadt)
  - Locations (Geokoordinaten)
- Minimalbeispiel:

```
import tweepy
stream = tweepy.Stream(Credentials)
stream.filter(track=['IB.SH'], languages = ['de'])
```

Tech-Stack

Tweets sammeln

- Python Skript, das rund um die Uhr auf einer virtuellen Maschine in der FH Kiel läuft
- Liste mit Namen von Stadtteilen, Parks und Plätzen
  - Blankenese, Ochsenwerder, Altstadt, Billstedt, Hamm, usw.
  - Marco-Polo-Terrassen, Hohe Bleichen, Lotsekai, usw.
- Jeder Tweet bekommt eine einzigartige ID und wird in einer MongoDB gespeichert
- Etwa 1500 Tweets pro Tag

Tech-Stack

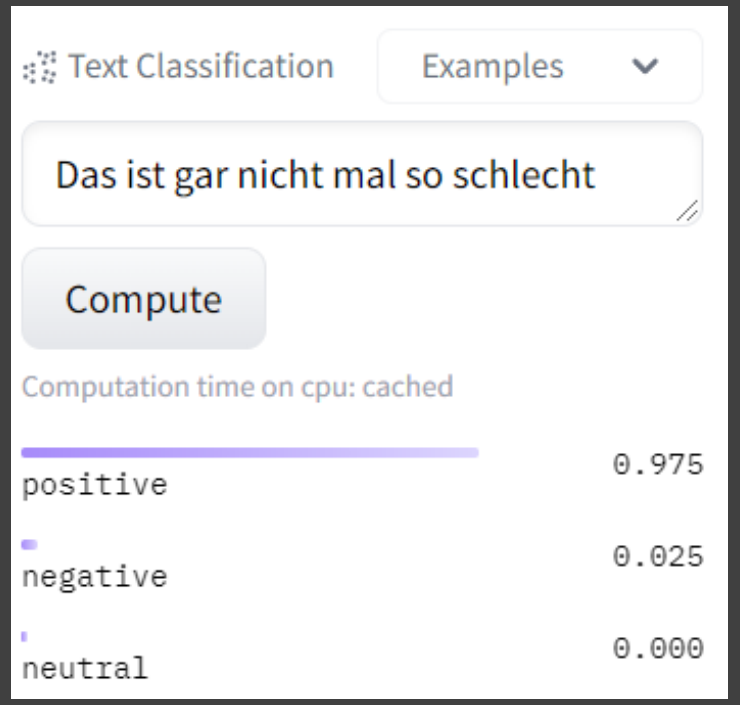
Tweets aufbereiten

- Export als CSV-Datei aus der MongoDB
  - Etwa 40.000 Tweets nach 4 Wochen
- Tweets enthalten häufig zwar Filterwort, aber beziehen sich nicht auf Bezirk
  - Filtern von Zuginformationen
  - Filtern von Tweets, die nicht eindeutige Bezirke enthalten und ‚Hamburg' nicht erwähnen
  - Filtern von Bezirken, die in dem Tweet keinen Ort darstellen (NLP-Modell)
- Tokenisierung der Tweets:
  - Entfernung von URLs
  - Löschen von Sonderzeichen wie ‚@'
  - Ersetzung von Zahlen durch Text
  - Kleinschreibung

Tech-Stack

Tweets analysieren (BERT)

- BERT ist ein vortrainiertes Sprachmodell für natural language processing (NLP) von Google
  - Nur für die englische Sprache
- Fein abgestimmtes deutsches Modell für Sentiment Klassifizierung
  - Trainiert mit Daten von Twitter, Facebook und Filmbewertungen
  - Gibt positiv, negativ oder neutral als Klassifizierung für einen Text zurück
  - Zusätzlich Wert für die Klassifizierung, je größer, desto sicherer die Klassifizierung



Tech-Stack

Visualisierung

Geographische Daten von openstreetmap.org (via Overpass API → REST-Schnittstelle)

⚡ Response in CSV oder JSON-Format

Umwandlung in GeoJSON (RFC #7946):

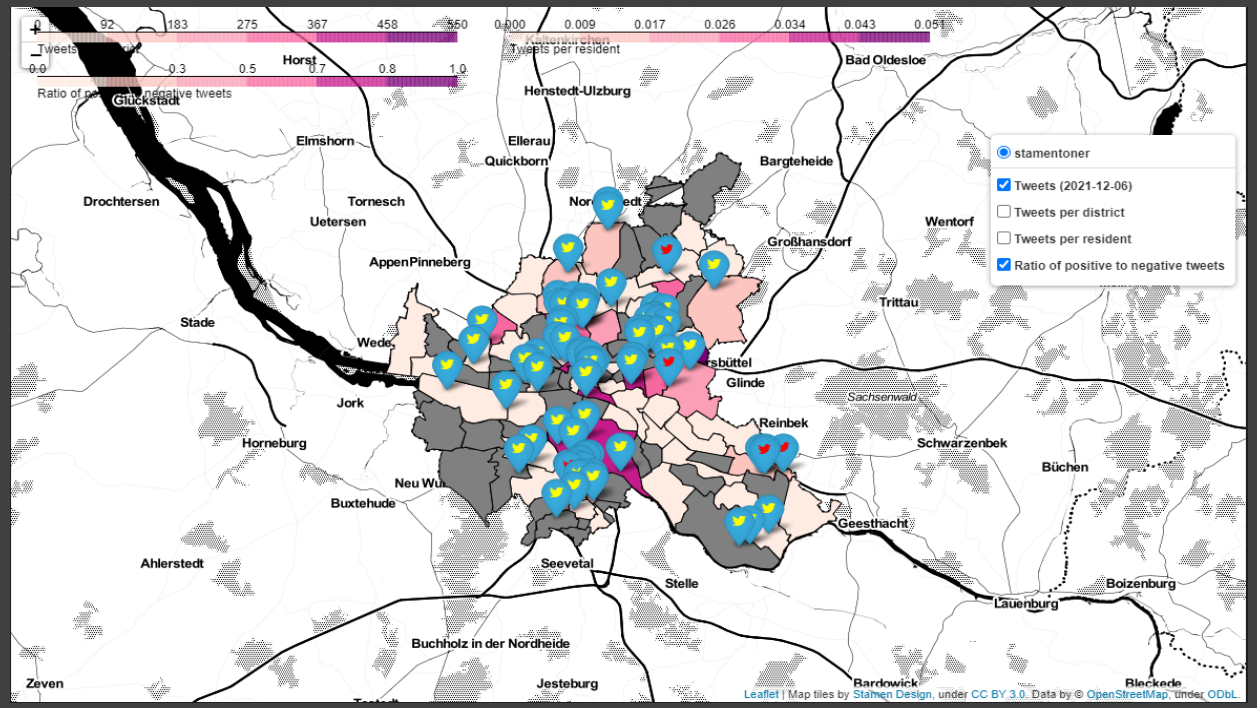
- Extra Zeile in Overpass-Query (`convert item …`)
- Weitere Verarbeitung der HTTP-Response notwendig

```
overpass_query_districts = """
[out:json];
area["ISO3166-2"="DE-HH"];
rel["boundary"="administrative"]["admin_level"=10](area);
convert item ::=::,::geom=geom(),_osm_type=type();
out geom;
"""
```

```
out = {"type": "FeatureCollection",
  "features": []}

for feature in response_data:
    feature["type"] = 'Feature'
    feature["name"] = feature["tags"]["name"]
    out["features"].append(feature)
```

Tech-Stack Visualisierung



Leaflet ermöglicht Erstellung interaktiver Karten mithilfe von JavaScript

- GeoJSON-Methoden, die `geometry`-Spalte eines Geopandas-Dataframes automatisch erkennen
- Interaktiv: Mausklicks, Hovern, Zoomen
- Anzeige basiert auf festem Datenstand, keine Live-Updates möglich

Herausforderungen

- Ernüchternde Auswahl an öffentlichen APIs

- Einstellungen für Twitter Stream finden
  - Vergleich von Städten / Bezirken / Stadtteilen?
  - Zuordnung über Inhalt oder Geo-Location?
  - Alternative ausprobiert: Konzentration auf einen Stadtteil – Tracking von Straßennamen / Plätzen

- Vorverarbeitung von Tweets
  - Uneindeutige Namen der Stadtteile
  - Bestätigung des Bezugs eines Tweets auf einen Bezirk

Vielen Dank!