Data Processing

SS 2020

Project Racial Sentiment Map

A Production of

Wolfgang Riegler 01553839

Robin Kulha 11838239

Kerstin Scharinger 11813383

Contents

1. Project description 3
   1.1. Big data 3
   1.2. Libraries and Analysis 4
   1.3. Datasets and licenses 5
      1.3.1. Census 5
      1.3.2. Tweets from Twitter 6
   1.4. Data protection concerns 6
2. Project 9
   2.1. Architecture 9
   2.2. Data preparation 10
   2.2.1. Twitter Producer 10
   2.2.2. Stream Processor 11
   2.2.3. Spark 12
   2.2.3.1. Census 12
   2.2.3.2. Twitter 12
   2.2.4. Visualization 15
      2.2.4.1. Conclusion 19
   2.3. Sentiment Analysis 20
   2.4. K-Means Clustering 21
   2.5. Discussion on the challenges encountered 24
   2.6. Summary of experienced gained by each team member 25
   2.7. Recommendations for future work 25

1. Project description

While we were searching for a subject for our project, we heard the breaking news about protests in America on the media. We got inspired by them to have a deeper look at the current situation going on in most of the states in America. The current unrest where young people and minorities express grievances over both racial inequality and the relationship with their government just started some days ago. These tensions had been triggered by a young member of the black community in Minneapolis who died during a police operation.

Our goal is to find data sources that give us more insights and new findings by combining them. We are interested in the further ongoing unrest developments and whether they are supported by most of the American civilian population. Are there any differences regarding the mood of the population between the areas? Does the diversity of the population (the amount of white, black people living in an area) influence the agreement to these protests?

To obtain data about the people's opinions about these protests, we are going to use data from Twitter restricted to the location of America. We are restricting the location to a list of selected cities to keep the workload for the server doable. Furthermore, we are going to use sentiment analysis on these data to determine the general mood about the tensions from the tweets. We are going to combine these data with a data set retrieved from census showing the diversity of the American population. As an outcome, we want to create a meaningful visualization using maps.

1.1. Big data

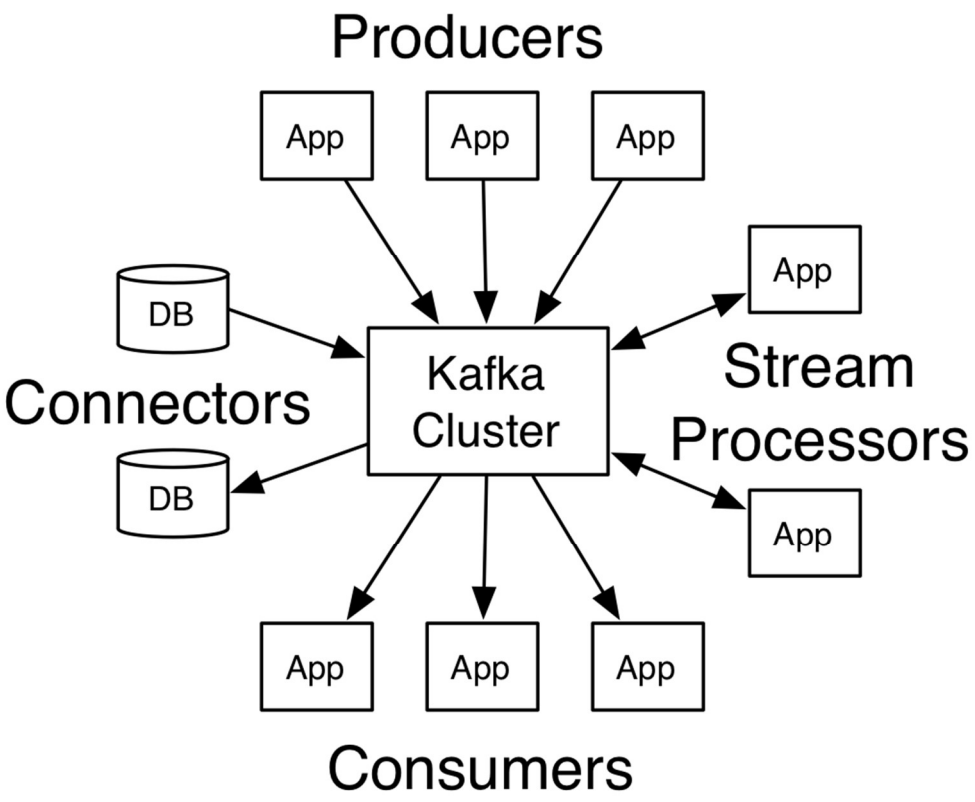
According to Gartner big data is "high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation."[1] Gartner mentions the volume, velocity, and variety of big data. The volume means the size of the data, big data projects have to deal with an increasing amount of data and therefore have to scale the systems used to handle the data. Velocity refers to the increasing speed of data generated today. Variety describes the diversity of the data generated. To sum up big data refers to the accelerated, unstructured mass of data generated every day by an increasing number of participants. Furthermore, regarding the aspects mentioned above the trust in data quality referred to as veracity is an important point to consider when dealing with big data. [2]

Taking these 4 V's of big data into consideration, we have to consider these aspects regarding our project in order to qualify it as a big data project. As mentioned above, we are going to obtain tweets from Twitter which are considered being unstructured data generated from a huge number of users. We expect to receive a high number of tweets during a short period of time. As we do not know the people generate the tweets and depending on the API provided by twitter to obtain the data, we have limited access to proof the data quality. This is also true for the census data, which is gathered by a survey and depend on the accuracy of the information provided by the survey participants.

[1] https://www.gartner.com/en/information-technology/glossary/big-data

[2] https://www.ibmbigdatahub.com/infographic/extracting-business-value-4-vs-big-data
https://bigdatapath.wordpress.com/2019/11/13/understanding-the-7-vs-of-big-data/

1.2. Libraries and Analysis



[3]

We use Kafka for our twitter data to handle the huge amount of data. We built a Producer to push tweets through our Twitter API into Kafka. Therefore, we specify a Kafka topic were the tweets are stored in Kafka. For the transformation of our tweets for example the performance of the sentiment analysis we use a stream processor where we consume an input stream from the topic specified and produce a desired output stream. And finally make use of a consumer to obtain our processed data and provide visualizations by using Spark Streaming.
To receive our Twitter data, we use the Tweepy Python API which allow us to connect to Twitter and stream real time tweets into Kafka.

All Tweets obtained through the Twitter API are stored in Kafka. Kafka writes data to disk and replicate data for fault-tolerance. The storage system of Kafka guarantees the preservation of data because of the full replicate system even if one server fails. The storage structure used by Kafka can scale and perform the same within an increasing amount of data.

To classify the mood of the people who publish a tweet on tweeter we need to use sentiment analysis. Sentiment analysis is a text classification tool that analyses incoming messages and tells the underlying sentiment from a range of -1 for a negative to 1 for a positive sentiment.[4]

[3] https://kafka.apache.org/intro
[4] https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17

With this, text is analyzed by combining natural language processing and machine learning to assign sentiment scores to sentences or phrases. First the text is broken into tokens
To perform the sentiment analysis in python we used the textblob package which offers a simple API to access its methods and perform natural language tasks. By using the sentiment function of textblob we obtain two properties, polarity, and subjectivity. Polarity lies between -1 for a negative statement and 1 for a positive statement. Subjectivity lies between 0 for factual information and 1 for a subjective statement such as a personal opinion, emotion for judgment. [5]

In addition to that, VADER, a library created for getting the sentiment of social media expressions, is used to get a comparison between the sentiments. With that we reduce the bias of only relying on one analysis tool to be correct.

We are going to combine the Tweets with data from Census. Therefore, we store and process the Census data in Spark using pandas. Before doing so, we joined the census data with the census blocks shapes to later plot it as a map.

Faust is a stream processing library, which is pure python. It can survive network problems and node failures. It is distributed because you can start more instances of your Faust app. The instances communicate with each other over Kafka. One single-core Faust worker instance is enough to process thousands of events.

1.3. Datasets and licenses

1.3.1. Census

We use the 2010 US Census dataset required from the National Historical Geographic Information System from the U.S. Census Bureau. From it we can get the racial diversity of a census block. To combine the two datasets the coordinates of a tweet will be matched with a census block.

To get the 2010 US Census dataset you need to register and agree to the NHGIS Usage License. This License consist of two main points:

- Do not redistribute the data without permission. It is only allowed to publish a subset of the data in a particular publication to meet the requirements of an academic journal. You need additional permission for any other redistribution.

- Cite the NHGIS data appropriately.

When you know the address of a person you can join information about the census block, like the income of households in that area.

Link: https://www.nhgis.org

[5] https://www.analyticsvidhya.com/blog/2018/02/natural-language-processing-for-beginners-using-textblob/

1.3.2. Tweets from Twitter

Twitter will be one of our sources to get information from when and where contents about the ‚BlackLivesMatter' movement / the protests against police violence were posted. Important attributes will be location attributes as ‚location' or the longitude and latitude if we can extract that because that will be used as a key attribute.

As mentioned above we will obtain the data with the Twitter API. Therefore, we have applied for a developer account and received keys to access to Twitter content.

Twitter grants us a n-exclusive, royalty free, non-transferable, non-sublicensable, revocable license if we stay within their terms and conditions and the Developer Policy. This allows us to use the Twitter API to:

- Use the Twitter API to integrate Twitter Content into your Services or conduct analysis of such Twitter Content;
- Copy a reasonable amount of and display the Twitter Content on and through your Services to End Users, as permitted by this Agreement;
- Modify Twitter Content only to format it for display on your Services; and
- Use and display Twitter Marks, solely to attribute Twitter's offerings as the source of the Twitter Content, as set forth herein.

1.4. Data protection concerns

We are not going to use the data in another way then described and planned in our project proposal (Data Repurposing). Moreover, we do not give data to third parties or share our data with others. We do not look at individual tweets or tweets from specific people to find out more about them. We do not analyze the sentiment of an individual, we interpret the overall or average sentiment in a specified area. Therefore, it should not be possible to identify a person from the data used (Personal Data).

In general, we use as little data as possible but as much as we need, and we just store it as long as we need it for our project (Data Minimization). Our project should be transparent due to our explanation what we do with the data (Transparency). [6]

6 https://rm.coe.int/big-data-and-data-protection-ico-information-commissioner-s-office/1680591220

1.5. Biases

In our project there are many biases which can occur. We probably didn't even think about all of them but here are some, quite important ones:

The average of a Census block does not usually fit to an individual person and its tweet.

Another bias is that we can only use tweets which contain location data. It is also not guaranteed that a person lives in the area if he/she tweets from a certain location.

Using biased keywords like 'blacklifesmatters' which is rater used of one side. This could lead to a bias doing our sentiment analysis. Therefore, we have decided to not filter the data using certain keywords.

While preparing our kafka producer to receive tweets through twitter API we realized that we were not able to filter our kafka stream for more than one criterion. Originally, we planned to filter the tweets which are located within a specified latitude and longitude and uses some keywords in the text. Unfortunately, the filter only supports either a filter for location or a filter for text, this means we only received data within a location, or a keyword mentioned in the text. As a result, we decided to retain the location filter and obtain all tweets within a specified area.

So, we are dealing with our concern of using wrong keywords since we can't know all related words to the protests. Moreover, using all tweets within a certain location and, then, comparing locations with now ongoing protests may result in a better outcome.

Regarding our algorithms, we may face some biases at implementing a sentiment analysis on our tweets. The tweets could be categorized with a wrong sentiment – e.g. ironical statements will not be considered as ironic in our analysis.

Furthermore, we have to keep in mind that our interpretation regarding the connection between the tweet sentiment and ongoing protests in America can be biased. There is a possibility that we retrieve wrong conclusions from our outcome because of any circumstances we haven't consider before. Moreover, some ongoing situations like the lockdown and the coronavirus situation may have a major impact on the average tweet sentiment and this aspect may lead to misinterpretation of our outcome.

To sum up, we are aware that we can hardly exclude any biases from our project. Also, we are aware that all steps, from procedures with the data itself, to the algorithms used and to our final interpretation of our project could cause biases.

1.6. Legal and Ethical Issues

Since we analyze tweets from twitter, we use the free of charge twitter API to retrieve twitter content. Hence, we must follow the Developer Policy provided by Twitter. While using data from twitter we must respect people's privacy and provide transparency in our analysis. Furthermore, by building on the Twitter API or accessing Twitter content we must comply with all Twitter policies like the Automation Rules, the Display Requirements, the

API Restricted Rules, the Twitter Rules…. Therefore, we restricted in our data processing to comply with all these polices. For instance, the Twitter API has rate limits which help to ensure fair data usage.

The tweets we obtain from the API are public data since Twitters' data policy classifies every tweet as public data by default. This means that every user must be aware that his data can be used by a third party. However, Twitter also provides protected tweets in his settings to classify tweets as non-public.[7]

Twitter does not allow to store, derive or infer following information about twitter users.

- Health (including pregnancy)
- Negative financial status or condition
- Political affiliation or beliefs
- Racial or ethnic origin
- Religious or philosophical affiliation or beliefs
- Sex life or sexual orientation
- Trade union membership
- Alleged or actual commission of a crime

Inherently, our project requires to extract such information as the political affiliation using the sentiment analysis. One legal way of getting and working with that information is not to store any personal data (for example, user IDs, usernames, and other identifiers) as Twitter suggests. In our project we followed this suggestion. Therefore, we are not be able to identify the user and the associated information. Another way would have been to look for the keyword, to get the attributes and information we need while not deriving any sensitive information.

Next, it is also prohibited to track, alert or monitor sensitive events such as protests. As we understood, this means we must not i.e. create a live map of active protests. Therefore, using keywords like ‚blacklifematters' is acceptable as long as we do not filter for ongoing protests.

Furthermore, location data must be in conjunction with the Twitter content to which it is attached. So that kind of data must not be used on a standalone basis. Twitter permits to use maps or other related tools that show aggregated geo activities (e.g. the number of people in a city using a hashtag).

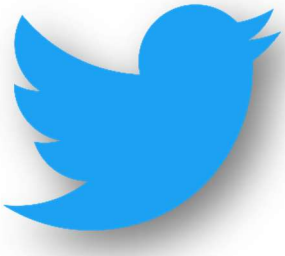
There are a lot more of restrictions like the prohibition of spam or spammy behavior if you automatically create content. However, those things probably will not interfere with our project. [8]

[7] https://help.twitter.com/de/safety-and-security/public-and-protected-tweets
[8] https://developer.twitter.com/en/developer-terms/agreement-and-policy

2. Project

2.1. Architecture

We obtain our source data from twitter by using the twitter API



We use a kafka producer to stream the data into kafka

We use faust as a stream processor and push the data back to kafka and to the SQLite database

We use the SQLite database to store are processed data and push it to spark

We create a pandas dataframe to push data from SQLite into Spark

Finally, we load the pandas dataframe into Spark

We obtain the percentage of white people in a census block from the census bureau.

We load the csv file into Apache Spark and the shape file into geopandas

2.2. Data preparation

2.2.1. Twitter Producer

First, we loaded the required libraries for our kafka producer. To obtain the data from twitter we used the python tweepy package.

```
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
from kafka import SimpleProducer, KafkaClient
```

We created a new kafka topic called “blacklifematters” and filtered our tweets by location, using coordinates for the whole United States.

```
class StdOutListener(StreamListener):
    def on_data(self, data):
        producer.send_messages("blacklifematters", data.encode('utf-8'))
        #print (data)
        return True
    def on_error(self, status):
        print (status)
```

```
kafka = KafkaClient("localhost:9092")
producer = SimpleProducer(kafka)

l = StdOutListener()

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

stream = Stream(auth, l)
stream.filter(locations=[-125.170127, 25.827821, -65.355876, 49.233598]) #Location =  USA
```

2.2.2. Stream Processor

The Stream processor subscribes to the topic “blacklifesmatters” and gets the raw output of Twitter API from Kafka. It the serializes the message automatically with JSON. Then it performs sentiment analysis with the VADER and TextbBob library and stores it into the dictionary of the tweet. It checks if the attributes are defined before extracting them. The coordinates and bounding_box attributes are containing also a dictionary, which is then converted to a json string. A challenge was that the name of the place where sometimes written in French and contained a single quote. Therefore, the single quotes got replaced with two single quotes. The SQL insert statement wasn’t a problem anymore for the SQLite database engine. Since the Faust workers are executed asynchrony it was a challenge that the database isn't locked. Finally, the modified dictionary of a tweet is sent to the “tweets” topic in Kafka.

```
import faust
from textblob import TextBlob
import re
import aiosqlite
#import nltk
#nltk.download('vader_lexicon')
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import json

app = faust.App(
    'stream_processor',
    broker='kafka://localhost:9092',
    value_serializer='json',
)

twitterAPI_topic = app.topic('blacklifematters')

tweets_topic = app.topic('tweets')
```

```
@app.agent(twitterAPI_topic)
async def getTweet(stream):
    db = await aiosqlite.connect('data/tweets.db')
    async for line in stream.filter(lambda p: 'id' in p):
        clean_tweet = ' '.join(re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z \t])|(\w+:\/\/\S+)", " ", line['text']).split())
        # create TextBlob object of passed tweet text
        analysis = TextBlob(clean_tweet)
        line['sentiment'] = analysis.sentiment.polarity
        #uses VADER for sentiment
        analyzer = SentimentIntensityAnalyzer()
        vs = analyzer.polarity_scores(clean_tweet)
        line['VADER'] = vs['compound']
        #extracts coordinates, bounding_box, full_name
        coordinates = "NULL"
        bounding_box = "NULL"
        full_name = "NULL"
        if line['coordinates']:
            coordinates = "'"+json.dumps(line['coordinates'])+"'"
        if line['place']:
            if line['place']['bounding_box']:
                bounding_box = "'"+json.dumps(line['place']['bounding_box'])+"'"
            if line['place']['full_name']:
                full_name =line['place']['full_name']
                full_name = full_name.replace("'","''")
        if line['id']:
            #inserts into sqlite database
            await db.execute("INSERT INTO tweets VALUES ("+str(line['id'])+", "+str(line['sentiment'])+", "+str(line['VADER'])+",
"+coordinates+", "+bounding_box+", '"+full_name+"')")
            await db.commit()

        #sends new message to tweets topic
        await tweets_topic.send(value=line)
    await db.close()
# enter in terminal:
#faust -A stream_processor worker -l info
```

2.2.3. Spark

2.2.3.1. Census

The columns from the census csv are renamed to get a more meaningful name for the columns. Additionally,

```
from pyspark.sql import SparkSession
import geopandas as gpd

# Build the SparkSession
spark = SparkSession.builder \
   .master("local") \
   .appName("rsm") \
   .config("spark.executor.memory", "1gb") \
   .getOrCreate()

sc = spark.sparkContext

# reads data from csv

race = spark.read.csv('./data/nhgis0005_csv/nhgis0005_ds172_2010_tract.csv', header=True, inferSchema=True)

race = race.withColumnRenamed("H7X001","TOTAL")
race = race.withColumnRenamed("H7X002","WHITE")
race = race.withColumnRenamed("H7X003","BLACK")
race = race.withColumnRenamed("H7X004","AMERCIAN_INDIAN_ALASKA_NATIVE")
race = race.withColumnRenamed("H7X005","ASIAN")
race = race.withColumnRenamed("H7X006","NATIVE_HAWAIIAN_OTHER_PACIFIC_ISLANDER")
race = race.withColumnRenamed("H7X007","OTHER")
race = race.withColumnRenamed("H7X008","TWO_OR_MORE")
```

2.2.3.2. Twitter

Load the tweets from sqlite database into spark. Therefore, we use pandas to read data from sqlite into pandas push it into spark.

```
#load the tweets into spark
from shapely.geometry import Point, LineString, Polygon
from shapely import geometry
import pyspark
from pyspark.sql.functions import *
import pyspark.sql.functions as F
from pyspark.sql import SparkSession
from pyspark.sql.functions import lit
from pyspark.sql.functions import exp


# Build the SparkSession
spark = SparkSession.builder \
   .master("local") \
   .appName("Tweets") \
   .config("spark.executor.memory", "1gb") \
   .getOrCreate()

sc = spark.sparkContext

#connect to sqlite
connection_uri = 'sqlite:///data/tweets.db'
db_engine = sqlalchemy.create_engine(connection_uri)

df_pd = pd.read_sql("SELECT * FROM tweets", db_engine)
df_pd
```

	id	textblob	vader	coordinates	bounding_box	full_name
0	1279841981611618310	-0.4	0.5267	None	{"type": "Polygon", "coordinates": [[[-89.5715...	Kentucky, USA
1	1279841964146462723	0.0	-0.1757	None	{"type": "Polygon", "coordinates": [[[-83.2880...	Detroit, MI
2	1279841982312046592	-0.1	-0.6369	None	{"type": "Polygon", "coordinates": [[[-95.8685...	Broken Arrow, OK
3	1279841982672564224	0.0	0.0000	None	{"type": "Polygon", "coordinates": [[[-122.326...	SeaTac, WA
4	1279841982601465856	0.8	-0.2732	None	{"type": "Polygon", "coordinates": [[[-87.6346...	Florida, USA
...	...	...	...	...	...	...
65962	1279862127344463872	0.0	0.0000	None	{"type": "Polygon", "coordinates": [[[-77.2652...	Tysons Corner, VA
65963	1279862127335964676	0.0	-0.4767	None	{"type": "Polygon", "coordinates": [[[-95.8232...	Houston, TX
65964	1279862127440695296	0.0	0.6369	{"type": "Point", "coordinates": [-111.9666355...	{"type": "Polygon", "coordinates": [[[-111.984...	Ammon, ID
65965	1279862127034015745	0.0	-0.2732	None	{"type": "Polygon", "coordinates": [[[-95.4107...	Neartown - Montrose, Houston
65966	1279862128816590853	-0.2	0.1593	None	{"type": "Polygon", "coordinates": [[[-106.645...	Texas, USA

65967 rows × 6 columns

We get the data into a pandas dataframe to inspect our data. Before we start to plot our data, we have to do some cleaning.

We create a spark dataframe:

```
df = spark.createDataFrame(df_pd)
df
```

```
DataFrame[id: bigint, textblob: double, vader: double, coordinates: string, bounding_box: string, full_name: string]
```

```
df.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- textblob: double (nullable = true)
 |-- vader: double (nullable = true)
 |-- coordinates: string (nullable = true)
 |-- bounding_box: string (nullable = true)
 |-- full_name: string (nullable = true)
```

The functions clean the column “bounding_box” and calculates a Longitude of the polygon center from the polygon inside the column.

```
import json
from pyspark.sql.functions import udf
def getCenterLong(line):
    point=None
    if line:

        coor = json.loads(line)
        sumlo = 0
        lo_count = 0
        for elem in coor['coordinates'][0]:
            sumlo += elem[0]
            lo_count += 1

        point = sumlo / lo_count


    return  point

clong_udf = udf(getCenterLong)
df= df.withColumn('CenterLo',clong_udf(df.bounding_box).cast("float"))

df.select('CenterLo').show()
```

```
+-----------+
|   CenterLo|
+-----------+
|  -85.76824|
|  -83.09929|
| -95.770195|
|-122.296486|
| -83.804474|
|-122.630905|
|  -118.4119|
|  -95.44649|
| -80.369156|
|  -79.27257|
| -118.15671|
| -73.955986|
|  -91.40101|
|  -97.39446|
| -121.87373|
|   -77.0144|
| -113.27558|
| -73.968544|
| -121.54538|
| -100.07689|
+-----------+
only showing top 20 rows
```

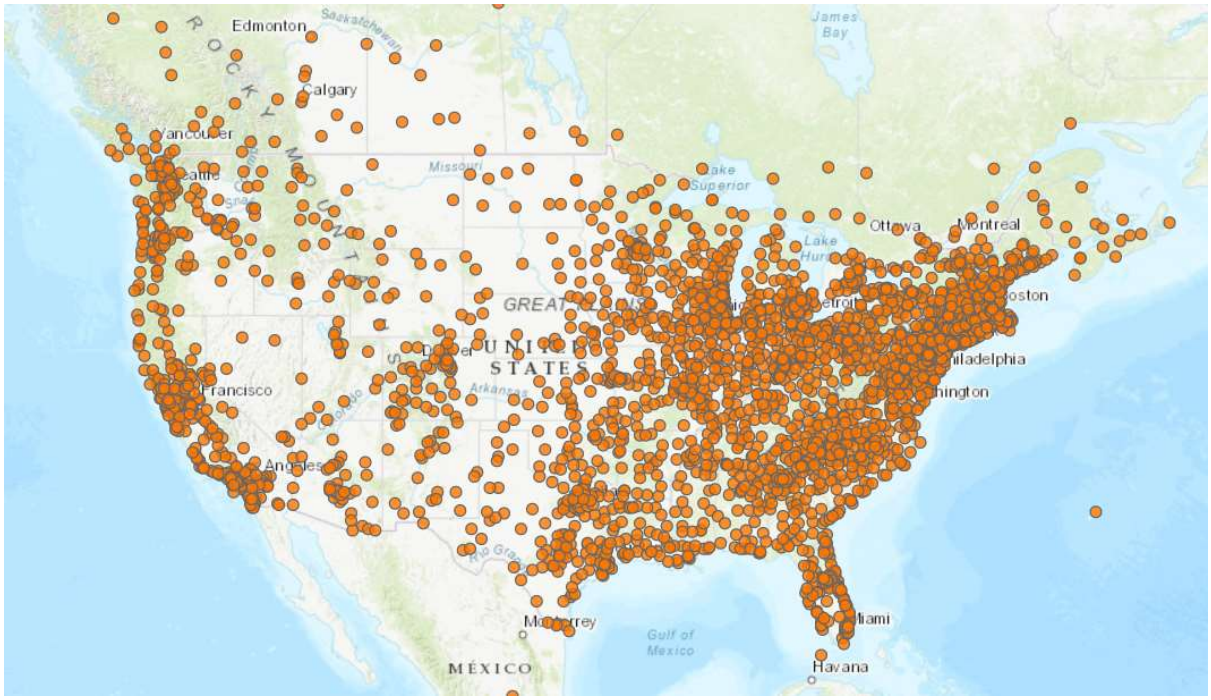
Finally, we calculated a latitude center and extracted the longitude and latitude from the coordinates column to receive coordinates to plot.

```
df.printSchema()
```

```
root
 |-- id: long (nullable = true)
 |-- textblob: double (nullable = true)
 |-- vader: double (nullable = true)
 |-- coordinates: string (nullable = true)
 |-- bounding_box: string (nullable = true)
 |-- full_name: string (nullable = true)
 |-- pointLo: float (nullable = true)
 |-- pointLat: float (nullable = true)
 |-- CenterLo: float (nullable = true)
 |-- CenterLat: float (nullable = true)
```

2.2.4. Visualization

The map shows cities or towns with black lives matter protests in the USA since May 25, 2020. There is a high density of protests on the east and west coast and particularly on the north east side. Therefore, we expect a higher number of Tweets in those areas.



[9]

We used geopandas and mathplotlib to plot our tweets on a map. First, we loaded a shapefile of the United States and remove Alaska and Hawaii. Afterwards we created a geometry variable with the columns longitude and latitude and added it to the twitter sentiment data frame.

```
#import a shape file with geopandas and remobe Alaska and Hawaii for the plot
import geopandas as gpd
import matplotlib.pyplot as plt
gdf = gpd.read_file('./data/UScountries/UScounties.shp')
gdf = gdf[gdf.STATE_NAME != "Alaska"]
gdf = gdf[gdf.STATE_NAME != "Hawaii"]
gdf.head()
```

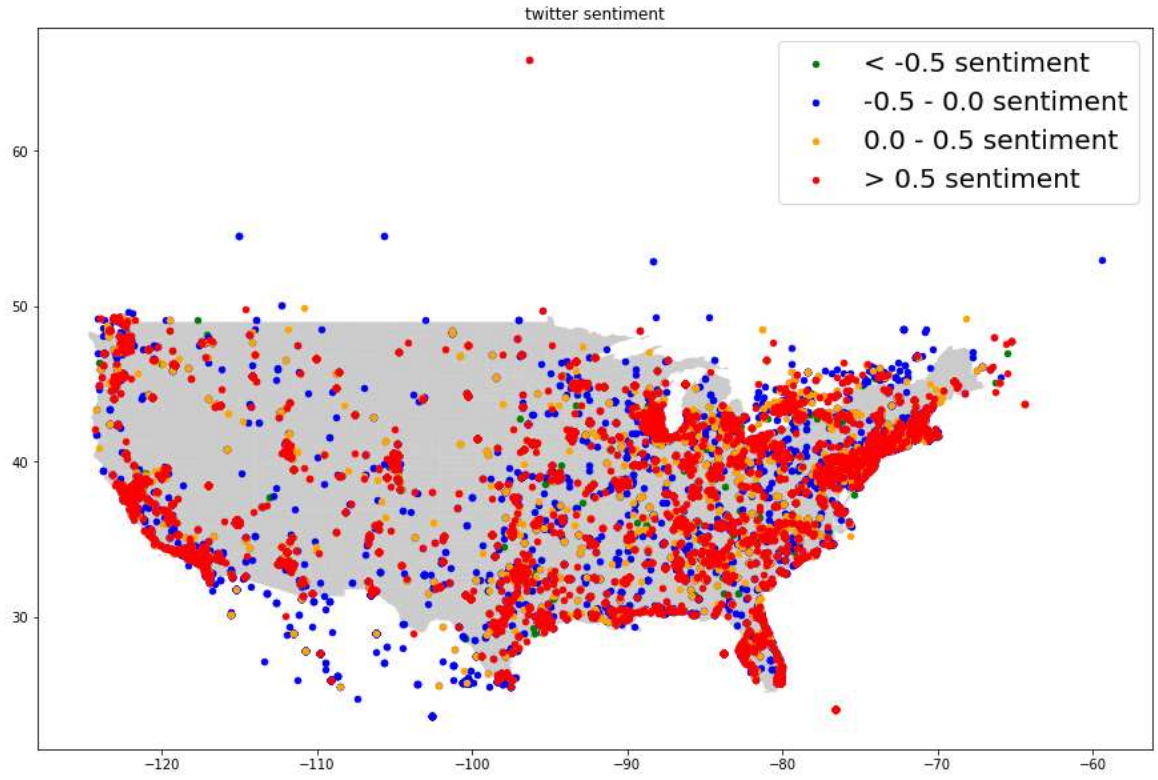
	NAME	STATE_NAME	STATE_FIPS	CNTY_FIPS	FIPS	geometry
0	Lake of the Woods	Minnesota	27	077	27077	POLYGON ((-95.34283 48.54668, -95.34105 48.715...
1	Ferry	Washington	53	019	53019	POLYGON ((-118.85163 47.94956, -118.84846 48.4...
2	Stevens	Washington	53	065	53065	POLYGON ((-117.43883 48.04412, -117.54219 48.0...
3	Okanogan	Washington	53	047	53047	POLYGON ((-118.97209 47.93915, -118.97406 47.9...
4	Pend Oreille	Washington	53	051	53051	POLYGON ((-117.43858 48.99992, -117.03205 48.9...

```
#Using the Longitude and Latitude of the Twitter Data Set to locate the different listings
geometry = [Point(xy) for xy in zip(sentiment_pd['CenterLo'], sentiment_pd['CenterLat'])]
geometry[:3]
sentiment_pd = gpd.GeoDataFrame(sentiment_pd, geometry = geometry)
```

[9] https://www.creosotemaps.com/blm2020/

Finally, we created a plot with the tweets on the map colored by sentiment.

```
#polt a map with tweets colored by sentiment
fig, ax = plt.subplots(figsize = (15,15))
gdf.plot(ax = ax, alpha = 0.4, color = 'grey')
sentiment_pd[sentiment_pd['vader'] <= -0.5].plot
(ax = ax, markersize = 20, color = 'green', marker = 'o', label = '< -0.5 sentiment')
sentiment_pd[(sentiment_pd['vader'] > -0.5) & (sentiment_pd['vader'] <= 0.0)].plot
(ax = ax, markersize = 20, color = 'blue', marker = 'o', label = '-0.5 - 0.0 sentiment')
sentiment_pd[(sentiment_pd['vader'] > 0.0) & (sentiment_pd['vader'] <= 0.5)].plot
(ax = ax, markersize = 20, color = 'orange', marker = 'o', label = '0.0 - 0.5 sentiment')
sentiment_pd[sentiment_pd['vader'] > 0.5].plot
(ax = ax, markersize = 20, color = 'red', marker = 'o', label = '> 0.5 sentiment')
plt.legend(prop={'size':20})
plt.title('twitter sentiment')
```



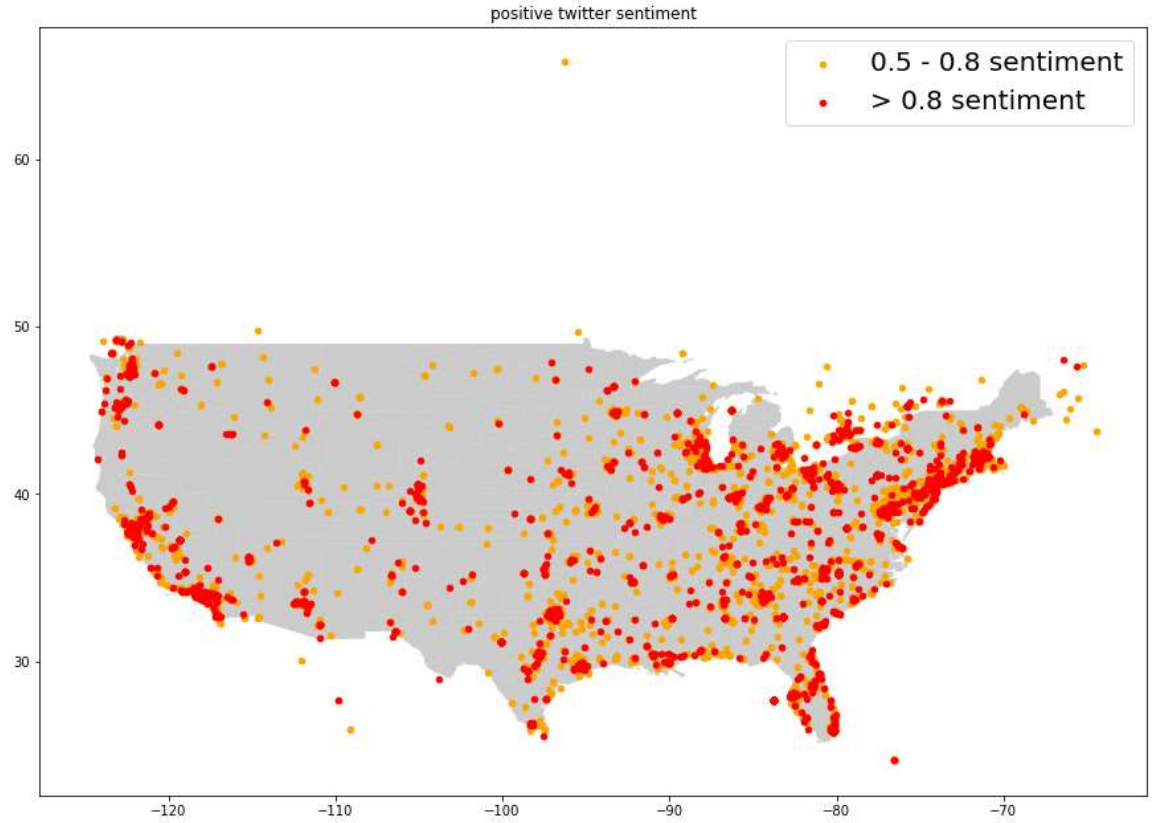
The plot shows tweets on the United Stated map categoriezed by sentiment. The rarely green points shows very negative sentiments below -0.5 on a scala from -1 for negative and 1 for a positiv sentiment. The blue points are showing a negative sentiment from -0.5 to 0.0, which can be considered to as lightly negative to neutral and dominate on the east coast. The lightly positive orange colored tweets are equally distributed over the map. Suprisingly the plot shows a domination of red colored very positive tweets.

Since there are so many tweets on one map, we decided to have a deeper look into both positive and negative tweets separately.

So first we looked at the positive sentiment of the tweets.

We used the same plot but filtered only positive sentiment higher than 0.5.

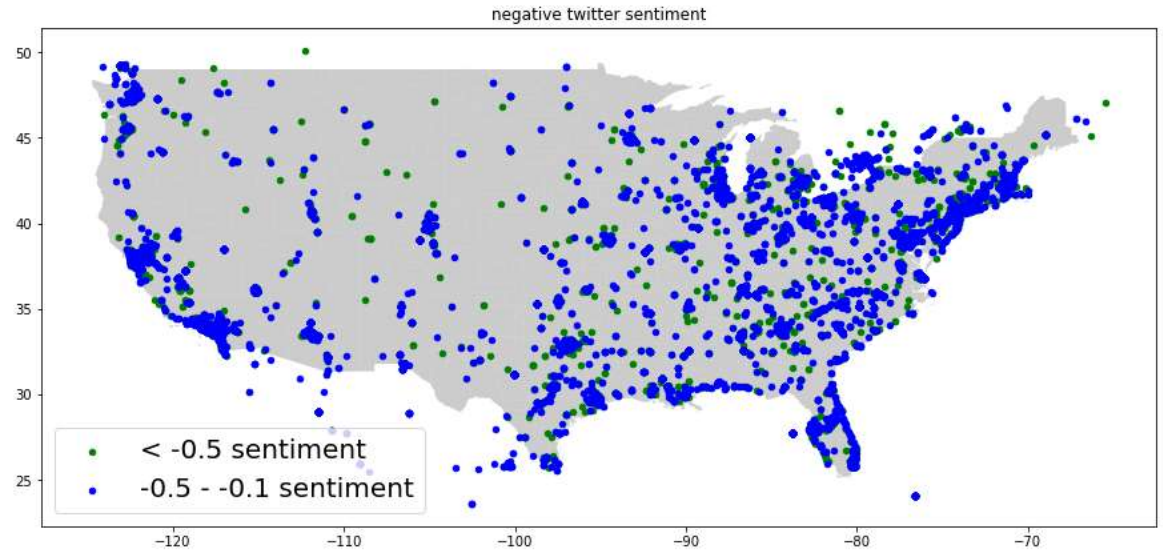
```
#have a deeper view into the very positive tweets
fig, ax = plt.subplots(figsize = (15,15))
gdf.plot(ax = ax, alpha = 0.4, color = 'grey')
sentiment_pd[(sentiment_pd['vader'] > 0.5) & (sentiment_pd['vader'] <= 0.8)].plot
(ax = ax, markersize = 20, color = 'orange', marker = 'o', label = '0.5 - 0.8 sentiment')
sentiment_pd[sentiment_pd['vader'] > 0.8].plot
(ax = ax, markersize = 20, color = 'red', marker = 'o', label = '> 0.8 sentiment')
plt.legend(prop={'size':20})
plt.title('positive twitter sentiment')
```



The plot shows approximately equal distributed orange twitter points which represents tweets with a sentiment classified from 0.5 to 0.8. The very positive sentiment above 0.8 are represented in collections particularly on the west coast in California and on the east coast around New York, Pennsylvania, and Florida.

Furthermore, we have inspected the negative tweets in particular. Therefore, we selected tweets with a sentiment below -0.1.

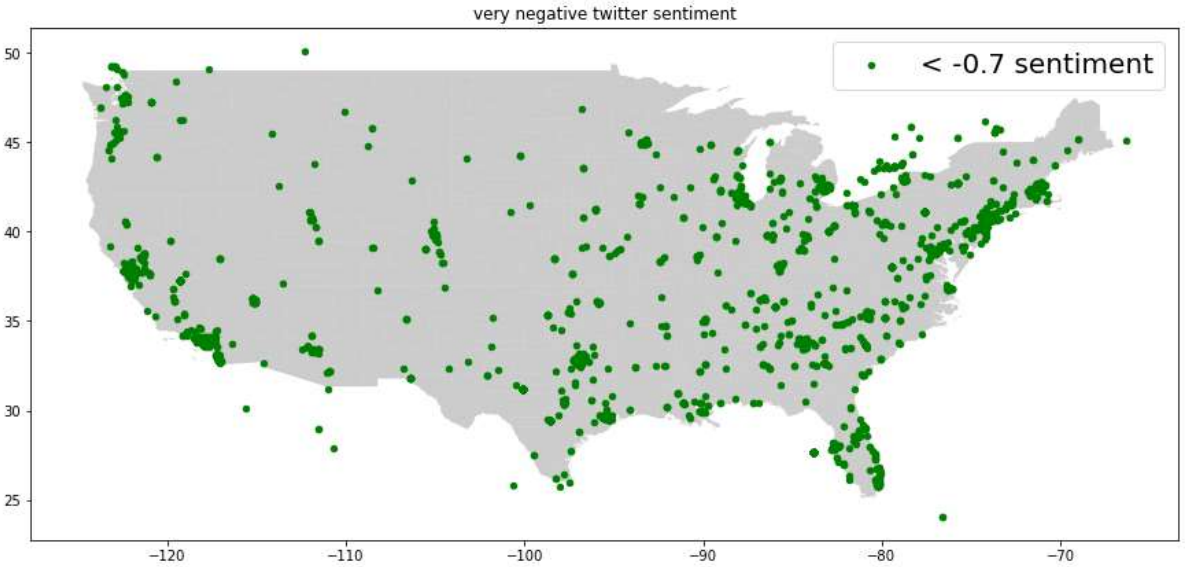
```
#have a deeper view into the very negative tweets
fig, ax = plt.subplots(figsize = (15,15))
gdf.plot(ax = ax, alpha = 0.4, color = 'grey')
sentiment_pd[sentiment_pd['vader'] <= -0.5].plot
(ax = ax, markersize = 20, color = 'green', marker = 'o', label = '< -0.5 sentiment')
sentiment_pd[(sentiment_pd['vader'] > -0.5) & (sentiment_pd['vader'] <= -0.1)].plot
(ax = ax, markersize = 20, color = 'blue', marker = 'o', label = '-0.5 - -0.1 sentiment')
plt.legend(prop={'size':20})
plt.title('negative twitter sentiment')
```



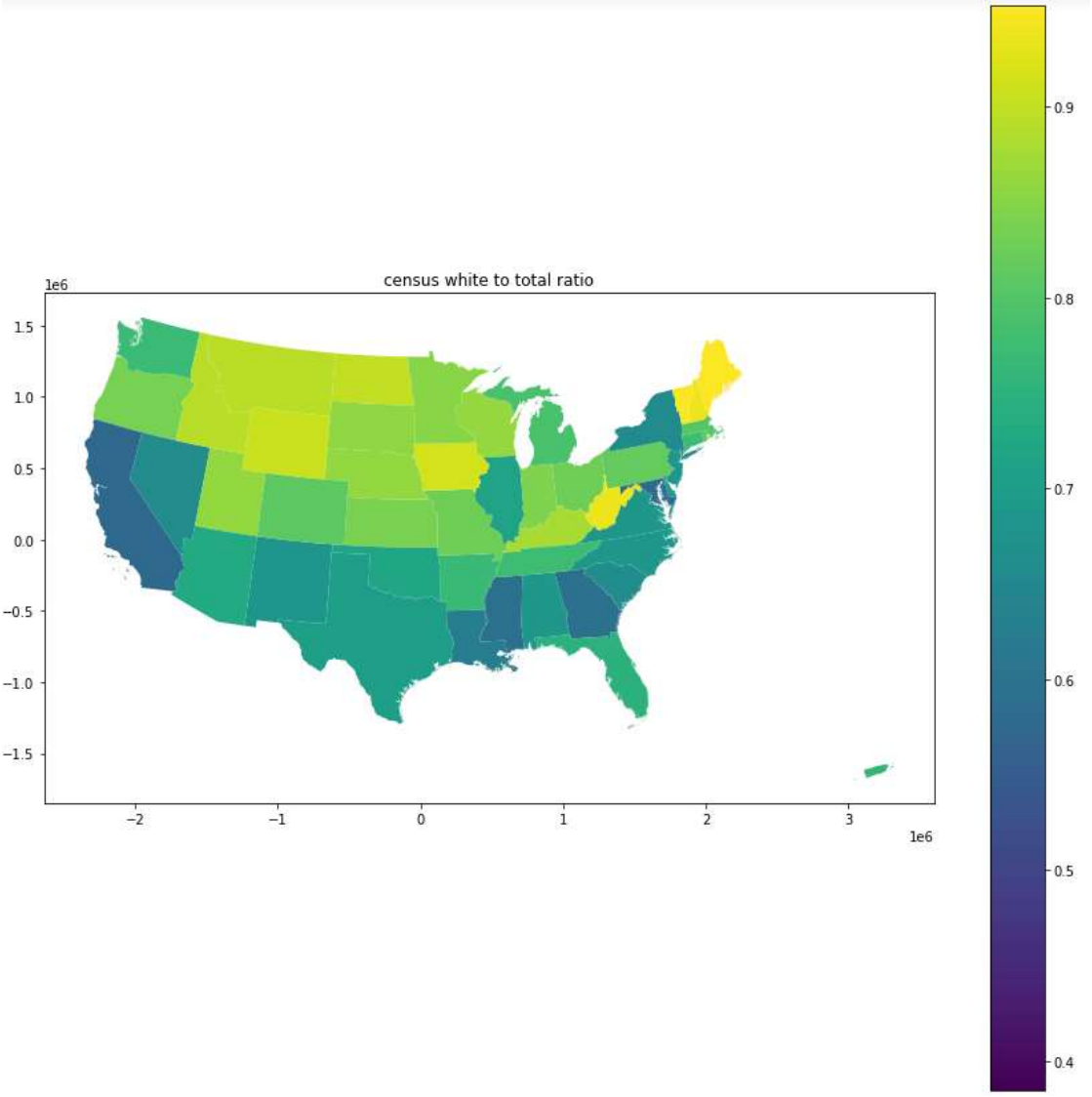
Indeed, the plot shows a lot of slightly negative to approximately neutral tweets all over the map. Furthermore, we can observe some very negative sentiments below -0.5 on the map. To learn more about these very negative sentiments we use an extra plot to gain some insides to these tweets.

We filtered only negative sentiments from -1 to -0.7.

```
#have a deeper view into the very negative tweets
fig, ax = plt.subplots(figsize = (15,15))
gdf.plot(ax = ax, alpha = 0.4, color = 'grey')
sentiment_pd[sentiment_pd['vader'] <= -0.7].plot
(ax = ax, markersize = 20, color = 'green', marker = 'o', label = '< -0.7 sentiment')
plt.legend(prop={'size':20})
plt.title('very negative twitter sentiment')
```



In fact, the very negative classified tweets are spread over america with some accumulations in metropolitan areas.



Steven Manson, Jonathan Schroeder, David Van Riper, and Steven Ruggles. IPUMS National Historical Geographic Information System: Version 14.0 [Database]. Minneapolis, MN: IPUMS. 2019. **http://doi.org/10.18128/D050.V14.0**

2.2.4.1. Conclusion

These maps represent a certain attitude or rather a certain sentiment over the USA regarding the 'BlackLivesMatter' movement. Using those, we cannot suggest any locational differences, as well as no obvious connections with the ratio of white people living in those areas.

2.3. Sentiment Analysis

Our sentiment analysis on the tweets in the USA worked as suggested. To have a better result, we used two tools to see possible calculational differences (see Biases). As a result, we even see some relatively great differences. Therefore, we made an average column of those two values.

The first table describes the average sentiment where most tweets were tweeted. The average sentiment is positive. Even so, there are no values greater than 0.1 which suggests only a moderate positive attitude.

full_name	avg(vader)	avg(textblob)	combined	tweets
Los Angeles, CA	0.05560393162393152	0.061033070597211345	0.058318501110571436	2340
Houston, TX	0.03760219364599093	0.050618888346762786	0.04411054099637686	1322
Georgia, USA	0.06728624904507241	0.07156311590469341	0.06942468247488291	1309
Florida, USA	0.05338608852755185	0.06711636240486736	0.060251225466209604	1107
Chicago, IL	0.06140386064030122	0.07050854138189347	0.06595620101109734	1062
Brooklyn, NY	0.06683566810344817	0.059670803633626944	0.06325323586853755	928
Manhattan, NY	0.06075510428100981	0.06814019930855818	0.064447651794784	911
Texas, USA	0.079449716231555	0.05781444742355643	0.06863208182755572	881
Atlanta, GA	0.06460183098591546	0.06763263334214036	0.0661172321640279	710
Philadelphia, PA	0.04100477657935287	0.04578042791255425	0.043392602245953565	649

Top 10 places with the most tweets

Secondly, we took a look at the places with the lowest sentiment. Noticeably, there are only two cities with a relatively low, even negative sentiment – Hermosillo and Hesperia. The others have a remarkably greater value. Also, Indianapolis is listed here which could suggest that the protests brought a greater countermovement. However, a clear outcome with this table is very difficult to determine.

Because of using a square shaped form for getting the tweets, it was not possible to exclude cities near to the US border like Hermosillo or Montreal.

full_name	avg(vader)	avg(textblob)	combined	tweets_total
Hermosillo, Sonora	-0.08700695652173912	-0.00851449275362...	-0.04776072463768...	115
Hesperia, CA	-0.00438296296296...	0.014141093474426809	0.004879065255731926	135
Miramar, FL	0.03059902912621359	0.016048837893498084	0.023323933509855836	103
Portland, OR	0.020301454545454546	0.030173175477720932	0.02523731501158774	275
Indianapolis, IN	0.023017721518987344	0.035666449188996655	0.029342085353992	316
Bahamas	0.05487522935779816	0.012250536161086626	0.03356288275944239	109
Delaware, USA	0.026714545454545425	0.04622126022126023	0.03646790283790283	110
Long Beach, CA	0.03753468208092487	0.0375993523175604	0.03756701719924263	173
Cleveland, OH	0.05742133333333335	0.01972036482036482	0.03857084907684909	150
Montréal, Québec	0.03542176165803109	0.04210522134744933	0.03876349150274021	193

Top 10 places with the lowest sentiment on average

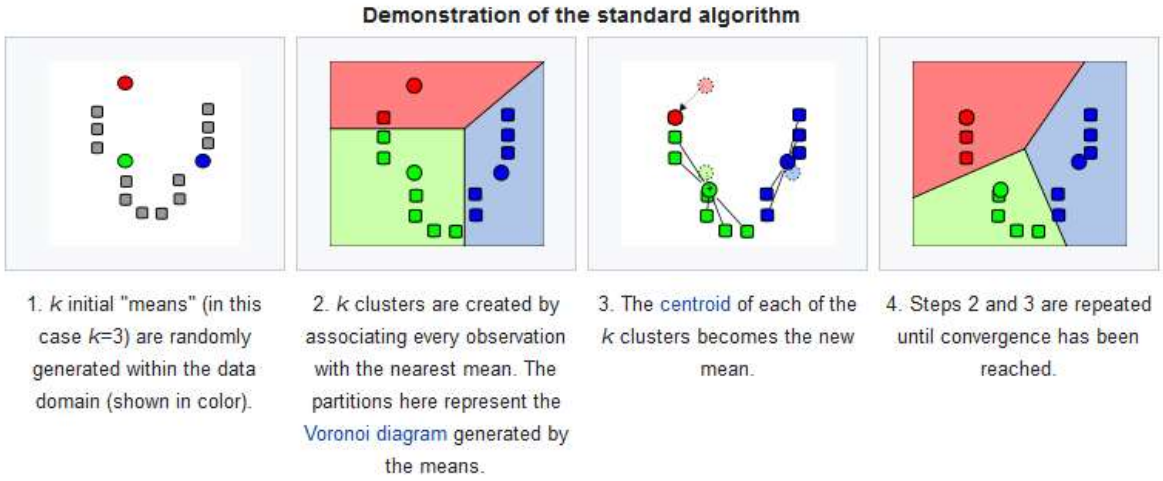
Last but not least, those are the places with the highest sentiment. Minnesota being highest with an average value of 0.15 is remarkably bigger than at places with most tweets. Also, those places don't have a high count of tweets. Therefore, it could be suggested that the number of tweets tweeted in a place doesn't determine the sentiment regarding the protests. However, also this suggestion would have biases.

full_name	avg(vader)	avg(textblob)	combined	tweets_total
Minnesota, USA	0.17827500000000002	0.11763537490559547	0.14795518745279773	136
St Louis, MO	0.14914545454545455	0.14402370041574591	0.14658457748060022	132
Tampa, FL	0.1590069930069931	0.10304417804417805	0.13102558552558558	143
Arlington, TX	0.1403728070175439	0.11817512531328321	0.12927396616541356	114
Massachusetts, USA	0.12595470085470087	0.10448824837713724	0.11522147461591906	117
Pennsylvania, USA	0.12305516014234867	0.09465626438936049	0.10885571226585458	562
Oklahoma City, OK	0.12686287878787877	0.08892364527307713	0.10789326203047794	132
Minneapolis, MN	0.1047654970760234	0.10726903538307048	0.10601726622954694	171
Raleigh, NC	0.11159674796747973	0.09646240277337838	0.10402957537042906	123
Fresno, CA	0.11132568807339445	0.09525679468340018	0.10329124137839732	109

Top 10 places with the highest sentiment on average

2.4. K-Means Clustering [10]

We are using K-Mean to see if clusters are correlated with the states of the USA. Frist, we need to build a Vector out of the features. Therefore, it is important to select only rows where the features are not null.



[10] https://en.wikipedia.org/wiki/K-means_clustering

```
from pyspark.ml.feature import VectorAssembler

assembler = VectorAssembler(inputCols=('CenterLo','CenterLat'),outputCol="features")

dataset=assembler.transform(df.filter('CenterLo is not null and CenterLat is not null '))
dataset.select("features").show()
```

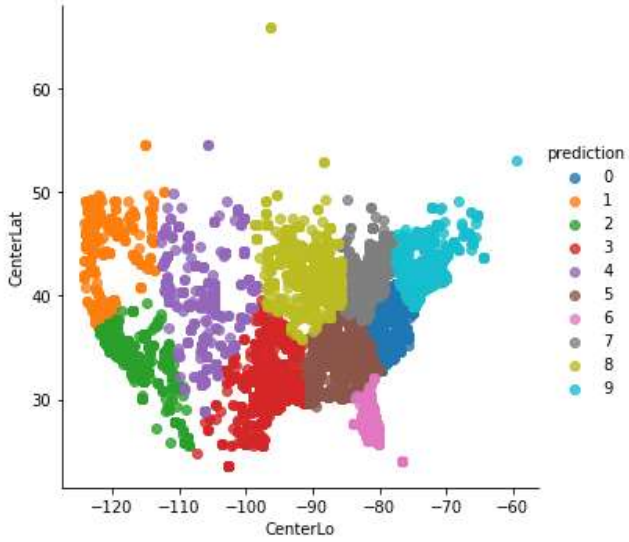
```
+--------------------+
|            features|
+--------------------+
|[-85.768241882324...|
|[-83.099288940429...|
|[-95.770195007324...|
|[-122.29648590087...|
|[-83.804473876953...|
|[-122.63090515136...|
|[-118.41190338134...|
|[-95.446487426757...|
|[-80.369155883789...|
|[-79.272567749023...|
|[-118.15670776367...|
|[-73.955986022949...|
|[-91.401008605957...|
|[-97.394462585449...|
|[-121.87373352050...|
|[-77.014396667480...|
|[-113.27558135986...|
|[-73.968544006347...|
|[-121.54537963867...|
|[-100.07688903808...|
+--------------------+
only showing top 20 rows
```

We did the clustering with 10 clusters. The Silhouette with squared Euclidian distance = 0,65 this means that the tweets are well classified.

```
from pyspark.ml.evaluation import ClusteringEvaluator
from pyspark.ml.clustering import KMeans

# Trains a k-means model.
kmeans = KMeans().setK(10).setSeed(1)
model = kmeans.fit(dataset)

# Make predictions
predictions = model.transform(dataset)

# Evaluate clustering by computing Silhouette score
evaluator = ClusteringEvaluator()

silhouette = evaluator.evaluate(predictions)
print("Silhouette with squared euclidean distance = " + str(silhouette))


# Evaluate clustering.
cost = model.computeCost(dataset)
print("Within Set Sum of Squared Errors = " + str(cost))

# Shows the result.
print("Cluster Centers: ")
ctr=[]
centers = model.clusterCenters()
for center in centers:
    ctr.append(center)
    print(center)
```

```
Silhouette with squared euclidean distance = 0.6517400600151246
Within Set Sum of Squared Errors = 691602.1945047432
Cluster Centers:
[-77.96114298  37.94426871]
[-121.85574793   42.1820093 ]
[-116.93507834   34.18773203]
[-96.68570449  31.49828754]
[-106.34073794   38.61772224]
[-85.37538357  33.582666  ]
[-81.66078941  27.32569536]
[-81.72411779  41.85329821]
[-89.7852252   41.76280131]
[-73.71610136  41.48924352]
```

```
import seaborn as sns

#plot clusters with seaborn
facet = sns.lmplot(data=predictions.toPandas(), x='CenterLo', y='CenterLat', hue='prediction',
                   fit_reg=False, legend=True, legend_out=True)
```



The plot shows the 10 clusters in which the location of the tweets is divided. We can see that cluster 6 clearly covers Florida, cluster 2 covers California and cluster 1 covers Oregon and Washington. Also, we see that the tweets are from all over the US. Through clustering we can also get an idea how the states of the US look like.

2.5. Discussion on the challenges encountered

The first challenge we encountered was to process the stream data gained from Twitter using Kafka. First, we didn't fully understand from where we could access the data but then we used Faust as a stream processor to finally push the data to a database in SQLite. After loading our data to spark, we could do further processing and, in the end, to plot the data.

A huge challenge was to deal with big sized shape files. Because of that we had difficulties with merging data frames and plotting both, the average sentiments and census data, on a map. After a while we decided to plot the data separately using two maps.

Plotting the results of the sentiment analysis was doable. However, plotting the census data turned out to be more challenging. We stopped counting how often the kernel has died abruptly. In the end we found a smaller sized shape file with only the states as shapes which finally worked.

The first challenge we encountered was to process the stream data gained from Twitter using Kafka. First, we didn't fully understand from where we could access the data but then we used Faust as a stream processor to finally push the data to a database in SQLite. After loading our data to spark, we could do further processing and, in the end, to plot the data.

A huge challenge was to deal with big sized shape files. Because of that we had difficulties with merging data frames and plotting both, the average sentiments and census data, on a map. After a while we decided to plot the data separately using two maps.

Plotting the results of the sentiment analysis was doable. However, plotting the census data turned out to be more challenging. We stopped counting how often the kernel has died abruptly. In the end we found a smaller sized shape file with only the states as shapes which finally worked.

2.6. Summary of experienced gained by each team member

Wolfgang: I learned how to use a stream processor to transfer data to a database and another topic. Also, I experienced to transfer data from a database to spark. So, I learned a lot about transforming data.

Kerstin: I learned how to handle a data stream and what are the difficulties with stream processing. Hence, I learned to consider ethical and legal issues while doing a project and how to deal with upcoming problems regarding these issues it.

Robin: I learned about data preparation with spark and geopandas and plotting various information on a map. Also, I learned about the legal aspects of using big data and what can be done with a sentiment analysis. However, I know that there is a lot more to learn for me.

2.7. Recommendations for future work

The Coordinate Reference System (CRS) of the US states shapefile should be transformed from Esri's USA Contiguous Albers Equal Area Conic projection to Latitude/Longitude. This is necessary to join the tweets with the US Census data on a map.

Furthermore, trying to reduce more biases would be possible, for instance by collecting more tweets. We could use the paid APIs, which allow more than one filter rule, from Twitter to also filter for keywords. So, we would get a sentiment analysis specific for a topic.