Sentimental analysis of Airbnb review on Spark

Big-data Management System

19512082 양승무
19512080 박종환
19512079 박민규

Contents

Background

- In this semester, we learned spark. We wanted to see it really works.

- What if dealing with big data?

  => Let's use RDD

- Airbnb reviews would fit on our work

Goal of the project

1. Performance Comparison between Local and Spark

2. Give useful review information to each host and customer

System Environment

Os : CentOS 7 (Red hat) / User : root

Java : 1.8.0_181

```
(base) [root@localhost ~]# java -version
java version "1.8.0_181"
Java(TM) SE Runtime Environment (build 1.8.0_181-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.181-b13, mixed mode)
```

Python : 3.7.4

```
(base) [root@localhost ~]# python --version
Python 3.7.4
```

Apache Spark : 2.3.2 / pyspark : 2.3.2

- We used spark in standalone mode

```
(base) [root@localhost ~]# pyspark --version
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.3.2
```

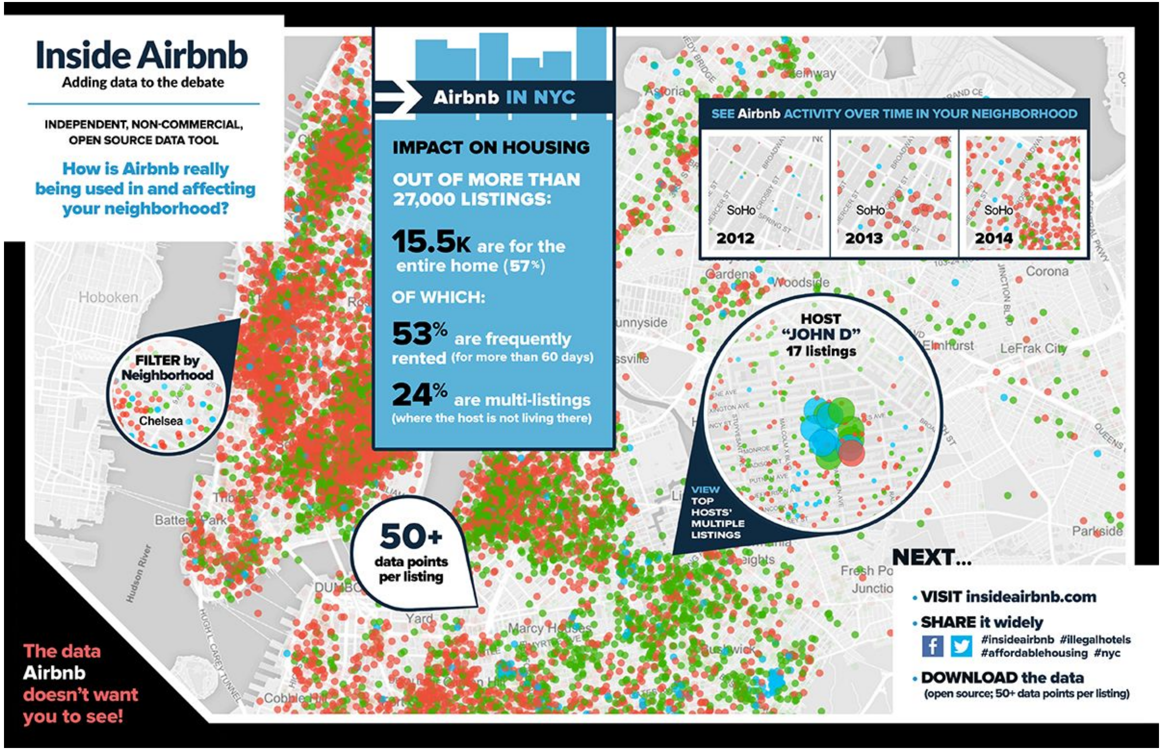
Data

1. Data Collection
    a. Airbnb dataset from : http://insideairbnb.com/get-the-data.html
    b. Berlin review data from the URL

2. Data Explanation
    a. reviews.csv.gz
    b. listings.csv.gz

Data

- Airbnb data



Data

- Airbnb data

Get the Data

The data behind the Inside Airbnb site is sourced from publicly available information from the Airbnb site.

The data has been analyzed, cleansed and aggregated where appropriate to faciliate public discussion. Read more disclaimers here.

If you would like to do further analysis or produce alternate visualisations of the data, it is available below under a Creative Commons CC0 1.0 Universal (CC0 1.0) "Public Domain Dedication" license.

Amsterdam, North Holland, The Netherlands

See Amsterdam data visually here.

Date Compiled	Country/City	File Name	Description
14 September, 2019	Amsterdam	listings.csv.gz	Detailed Listings data for Amsterdam
14 September, 2019	Amsterdam	calendar.csv.gz	Detailed Calendar Data for listings in Amsterdam
14 September, 2019	Amsterdam	reviews.csv.gz	Detailed Review Data for listings in Amsterdam
14 September, 2019	Amsterdam	listings.csv	Summary information and metrics for listings in Amsterdam (good for visualisations).
14 September, 2019	Amsterdam	reviews.csv	Summary Review data and Listing ID (to facilitate time based analytics and visualisations linked to a listing).
N/A	Amsterdam	neighbourhoods.csv	Neighbourhood list for geo filter. Sourced from city or open source GIS files.
N/A	Amsterdam	neighbourhoods.geojson	GeoJSON file of neighbourhoods of the city.

show archived data

Antwerp, Flemish Region, Belgium

See Antwerp data visually here.

Date Compiled	Country/City	File Name	Description
25 September, 2019	Antwerp	listings.csv.gz	Detailed Listings data for Antwerp
25 September, 2019	Antwerp	calendar.csv.gz	Detailed Calendar Data for listings in Antwerp
25 September, 2019	Antwerp	reviews.csv.gz	Detailed Review Data for listings in Antwerp
25 September, 2019	Antwerp	listings.csv	Summary information and metrics for listings in Antwerp (good for visualisations).
25 September, 2019	Antwerp	reviews.csv	Summary Review data and Listing ID (to facilitate time based analytics and visualisations linked to a listing).

Data

- Airbnb data
  - Has reviews on 101 City's accommodation. On this task, we used Berlin's data.
  - > Sentimental analysis takes enough time to compare.

  - reviews.csv.gz : Each accommodation has reviews of its each guest.

  - listings.csv.gz : Information of each accommodation.

Data

- reviews.csv.gz

	listing_id	id	date	reviewer_id	reviewer_name	comments
0	1944	7126992	2013-09-07	8207524	Mirko	I want to thank Laura&Emiliano for their hospi...
1	1944	7428447	2013-09-19	3021574	Rafiee	Very convenient and very quiet. You will stay...
2	1944	8455250	2013-10-31	5875429	Grzegorz	I've spent 2 nights at place of Laura and Emil...
3	1944	11105498	2014-03-20	5361252	Ngọc Thúy	The reservation was canceled 2 days before arr...
4	1944	15920963	2014-07-18	6659444	Nathalie	Laura est très sympathique et l'appartement fa...

- listing_id : ID code of accommodation
- id : User id
- reviewer_id , reviewer_name : Information of reviewers
- comments : review of accommodation

Data

- listings.csv.gz

	id	name	host_id	host_name	neighbourhood_group	neighbourhood	latitude	longitude	room_type	price	minimum_nights	number_of_reviews	last_review	reviews_per_month	calculated_host_listings_count	availab
0	1944	cafeheaven Pberg/Mitte/Wed for the summer 2019	2164	Lulah	Mitte	Brunnenstr. Nord	52.54425	13.39749	Private room	21	120	18	2018-11-11	0.25	1	
1	2015	Berlin-Mitte Value! Quiet courtyard/very central	2217	Ion	Mitte	Brunnenstr. Süd	52.53454	13.40256	Entire home/apt	60	4	127	2019-09-05	3.03	5	
2	3176	Fabulous Flat in great Location	3718	Britta	Pankow	Prenzlauer Berg Südwest	52.53500	13.41758	Entire home/apt	90	62	145	2019-06-27	1.16	1	
3	3309	BerlinSpot Schöneberg near KaDeWe	4108	Jana	Tempelhof - Schöneberg	Schöneberg-Nord	52.49885	13.34906	Private room	28	7	27	2019-05-31	0.36	1	
4	6883	Stylish East Side Loft in Center with AC & 2 b...	16149	Steffen	Friedrichshain-Kreuzberg	Frankfurter Allee Süd FK	52.51171	13.45477	Entire home/apt	125	3	126	2019-09-08	1.08	1	

Data

- listings.csv.gz
  - id , name : hotel's information
  - host_id ,host_name : host's information
  - neighbourhood_group ,neighbourhood : Partition of City that located accommodation
  - latitude ,longitude : location
  - room_type : Type of room (Private room, apt, … etc)
  - minimum_nights : least of days
  - number_of_reviews : how many reviews for each accommodation
  - last_review : time of recent review
  - reviews_per_month ,calculated_host_listings_count : indicators
  - avaliability_365 : a day open for a year

Data

- dataset (merge listing.csv.gz , review.csv.gz)

	listing_id	id	date	reviewer_id	reviewer_name	comments	neighbourhood_group	host_id	latitude
0	1944	7126992	2013-09-07	8207524	Mirko	I want to thank Laura&Emiliano for their hospi...	Mitte	2164	52.54425
1	1944	7428447	2013-09-19	3021574	Rafiee	Very convenient and very quiet. You will stay...	Mitte	2164	52.54425
2	1944	8455250	2013-10-31	5875429	Grzegorz	I've spent 2 nights at place of Laura and Emil...	Mitte	2164	52.54425
3	1944	11105498	2014-03-20	5361252	Ngọc Thúy	The reservation was canceled 2 days before arr...	Mitte	2164	52.54425
4	1944	15920963	2014-07-18	6659444	Nathalie	Laura est très sympathique et l'appartement fa...	Mitte	2164	52.54425

Data

- EDA
    a. Top 1,2,3 host
    b. Language Detection

Data

- EDA - Top1 host

neighbourhood_group	listing_id
Friedrichshain-Kreuzberg	1
Mitte	47

room_type	listing_id
Private room	6
Shared room	42

Data

- EDA - Top2 host

neighbourhood_group	listing_id
Charlottenburg-Wilm.	3
Friedrichshain-Kreuzberg	2
Lichtenberg	1
Mitte	9
Neukölln	1
Pankow	20
Reinickendorf	3
Tempelhof - Schöneberg	7

room_type	listing_id
Entire home/apt	46

Data

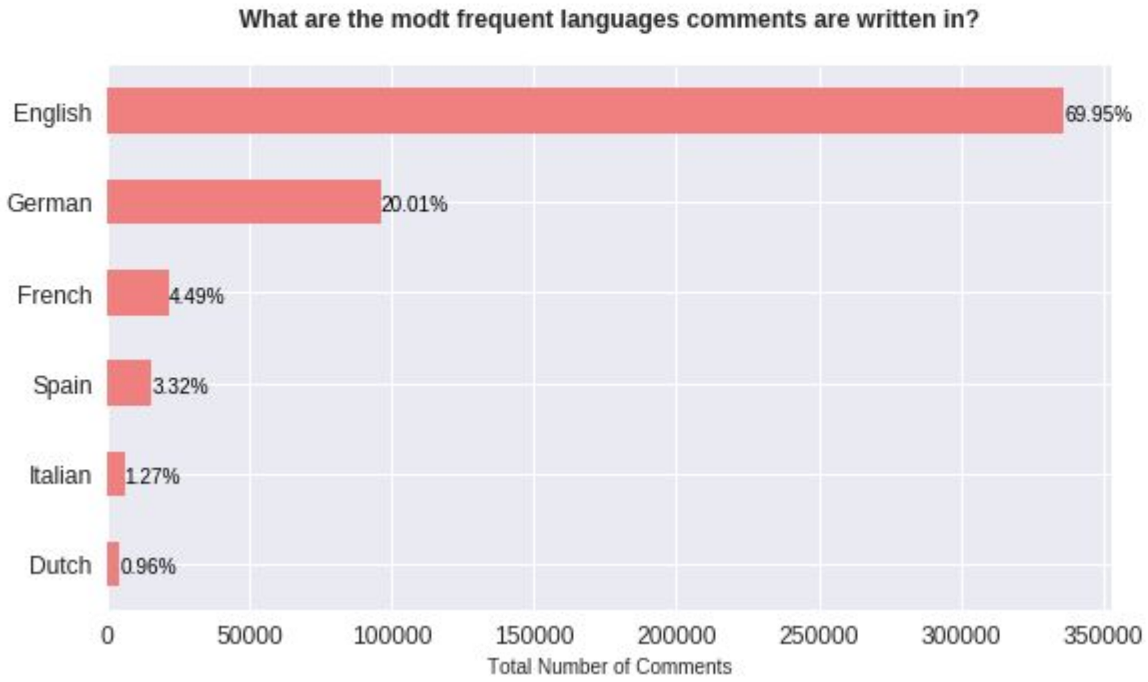
- EDA - Top3 host

neighbourhood_group	listing_id
Mitte	28

room_type	listing_id
Private room	4
Shared room	24

Data

- EDA - Language Detection



Analysis

1. Vader, Sentimental analysis
    a. Reviews were grouped by listing_id
    b. Local
    c. Spark
        i. First, we loaded csv file from a local filesystem(not hadoop)
        ii. Second, we changed dataframe into RDD to make it faster
        iii. Third, we used Vader with map
        iv. Fourth, we changed analyzed RDD into dataframe again
        v. Finally, we changed it into pandas to merge with listing_id by index

2. WordCloud for 3 listings

Analysis - Sentimental analysis on 'local' and 'Spark'

Sentiment Analysis on "local"

In [32]:
```
start = time.time()

analyzer = SentimentIntensityAnalyzer()

def negative_score(text):
    negative_value = analyzer.polarity_scores(text)['neg']
    return negative_value

def neutral_score(text):
    neutral_value = analyzer.polarity_scores(text)['neu']
    return neutral_value

def positive_score(text):
    positive_value = analyzer.polarity_scores(text)['pos']
    return positive_value

def compound_score(text):
    compound_value = analyzer.polarity_scores(text)['compound']
    return compound_value

df_eng['neg'] = df_eng['comments'].apply(negative_score)
df_eng['neu'] = df_eng['comments'].apply(neutral_score)
df_eng['pos'] = df_eng['comments'].apply(positive_score)
df_eng['compound'] = df_eng['comments'].apply(compound_score)

end = time.time()
print("time : {}".format(end - start))

df_eng
```

```
time : 839.0309975147247
```

839 sec : 14 minutes

254 sec : 4 minutes

```
def sentimentWordsFunct(x):
    senti_list = []
    neg = negative_score(x)
    neu = neutral_score(x)
    pos = positive_score(x)
    compound = compound_score(x)
    senti_list.append(float(neg))
    senti_list.append(float(neu))
    senti_list.append(float(pos))
    senti_list.append(float(compound))

    return senti_list

result = df2.map(sentimentWordsFunct)

# Create schema2 for dataframe
schema2 = StructType([StructField("neg", FloatType(), True)\
                     ,StructField("neu", FloatType(), True)\
                     ,StructField("pos", FloatType(), True)\
                     ,StructField("compound", FloatType(), True)])

# Turn to dataframe
sentiment_result = sqlContext.createDataFrame(result, schema2)

# Turn to pandas dataframe
pandas_sentiment = sentiment_result.toPandas()
df_local = pd.read_csv('df_eng.csv', sep='\t')

# Merge dataframe
df_final = pd.concat([df_local, pandas_sentiment], axis=1)

end = time.time()
print("time : {}".format(end - start))

df_final
```

```
time : 254.0568642616272
```

Spark is faster than Local

Sentiment Analysis on "Spark"

```python
# Set Spark
conf = SparkConf().setMaster("spark://10.10.20.53:7077")\
.setAppName("sentiment_analy")
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
```

```python
# Create schema
schema = StructType([
    StructField("linsting_id", IntegerType(), True),
    StructField("comments", StringType(), True)])

df = sqlContext.read\
.format('com.databricks.spark.csv')\
.options(header = 'true', inferschema = 'true')\
.options(delimiter = '\t')\
.schema(schema)\
.load('df_eng.csv')

# Turn into RDD
df_rdd = df.select("comments").rdd.flatMap(lambda x: x)
header = df_rdd.first()
df2 = df_rdd.filter(lambda row: row != header)

# Sentiment Analysis and check time
start = time.time()

analyzer = SentimentIntensityAnalyzer()

def negative_score(text):
    negative_value = analyzer.polarity_scores(text)['neg']
    return negative_value

def neutral_score(text):
    neutral_value = analyzer.polarity_scores(text)['neu']
    return neutral_value

def positive_score(text):
    positive_value = analyzer.polarity_scores(text)['pos']
    return positive_value
```

```python
    compound_value = analyzer.polarity_scores(text)['compound']
    return compound_value

def sentimentWordsFunct(x):
    senti_list = []
    neg = negative_score(x)
    neu = neutral_score(x)
    pos = positive_score(x)
    compound = compound_score(x)
    senti_list.append(float(neg))
    senti_list.append(float(neu))
    senti_list.append(float(pos))
    senti_list.append(float(compound))

    return senti_list

result = df2.map(sentimentWordsFunct)

# Create schema2 for dataframe
schema2 = StructType([StructField("neg", FloatType(), True)\
                     ,StructField("neu", FloatType(), True)\
                     ,StructField("pos", FloatType(), True)\
                     ,StructField("compound", FloatType(), True)])

# Turn to dataframe
sentiment_result = sqlContext.createDataFrame(result, schema2)

# Turn to pandas dataframe
pandas_sentiment = sentiment_result.toPandas()
df_local = pd.read_csv('df_eng.csv', sep='\t')

# Merge dataframe
df_final = pd.concat([df_local, pandas_sentiment], axis=1)

end = time.time()
print("time : {}".format(end - start))

df_final
```

time : 254.0568642616272

Analysis - Sentimental analysis on ‘local’ and ‘Spark’

10.10.20.53:8080

Spark Master at spark://10.10.20.53:7077

Spark 2.3.2

URL: spark://10.10.20.53:7077
REST URL: spark://10.10.20.53:6066 *(cluster mode)*
Alive Workers: 1
Cores in use: 12 Total, 6 Used
Memory in use: 6.5 GB Total, 6.0 GB Used
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory
worker-20191206205901-10.10.20.53-38287	10.10.20.53:38287	ALIVE	12 (6 Used)	6.5 GB (6.0 GB Used)

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration
app-20191206214341-0000 (kill)	sentiment_analy	6	2.0 GB	2019/12/06 21:43:41	root	RUNNING	3.4 h

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Submitted Time	User	State	Duration

Analysis - Sentimental analysis on ‘local’ and ‘Spark’

10.10.20.53:4040/jobs/

Spark 2.3.2 | Jobs | Stages | Storage | Environment | Executors | SQL | sentiment_analy application UI

Spark Jobs (?)

User: root
Total Uptime: 3.4 h
Scheduling Mode: FIFO
Completed Jobs: 2

▸ Event Timeline

Completed Jobs (2)

Job Id ▾	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
1	toPandas at <ipython-input-34-c0580aba0553>:64 toPandas at <ipython-input-34-c0580aba0553>:64	2019/12/06 21:43:48	4.2 min	1/1	6/6
0	runJob at PythonRDD.scala:152 runJob at PythonRDD.scala:152	2019/12/06 21:43:47	1 s	1/1	1/1

Analysis - WordCloud

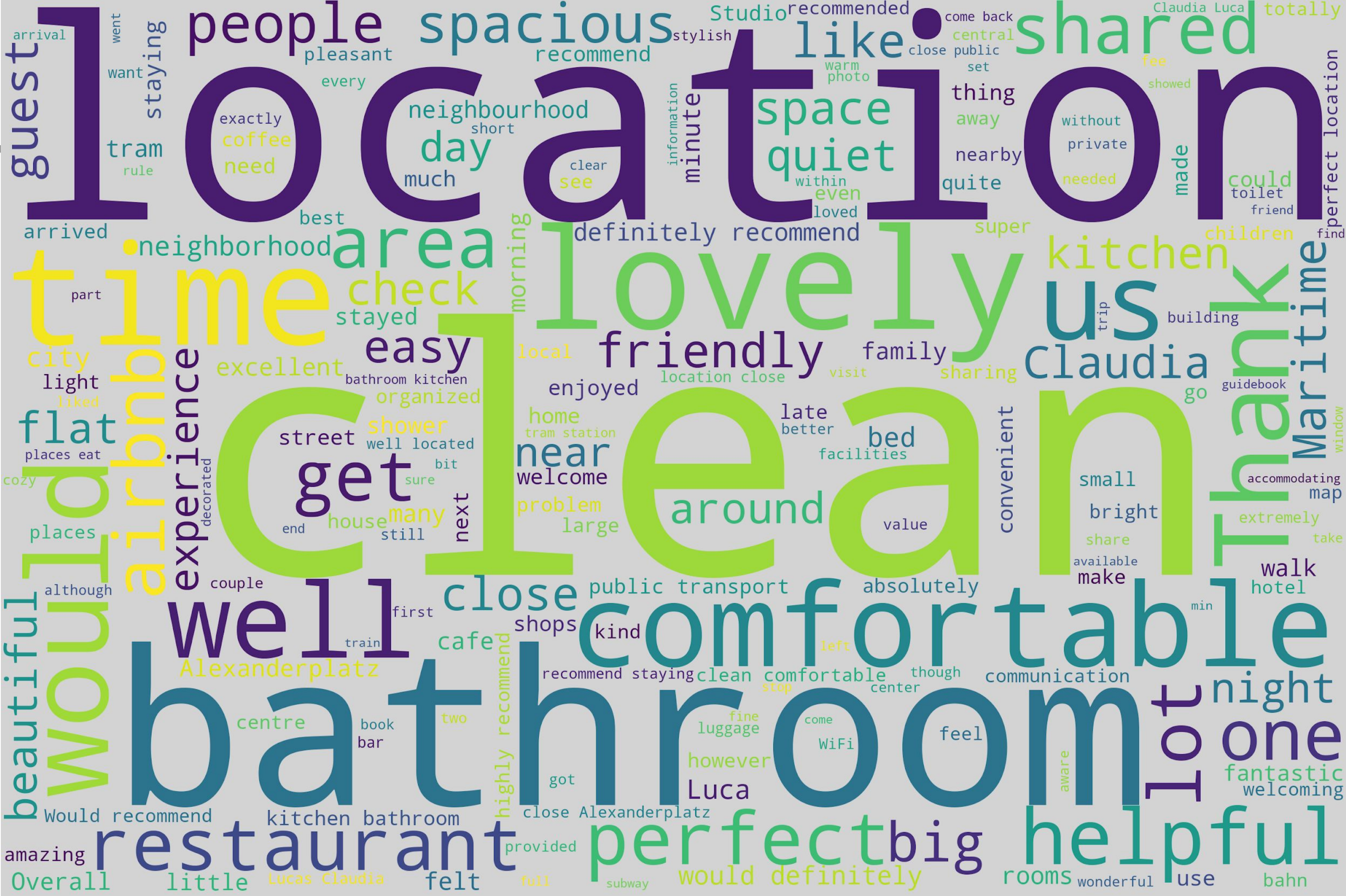
- First, we saw the sorted review_number by listing_id
- We chose top 1,2,4 listing_id from the rank(not same host_id)
- We made WordCloud except stopwords that we filtered by our own subjective criteria.
- The host and customers can use these information for their own purpose.
    - example : customers can choose an accommodation among WordCloud by their priorities.

Top 1 listing_id : 292864

number of reviews : 582

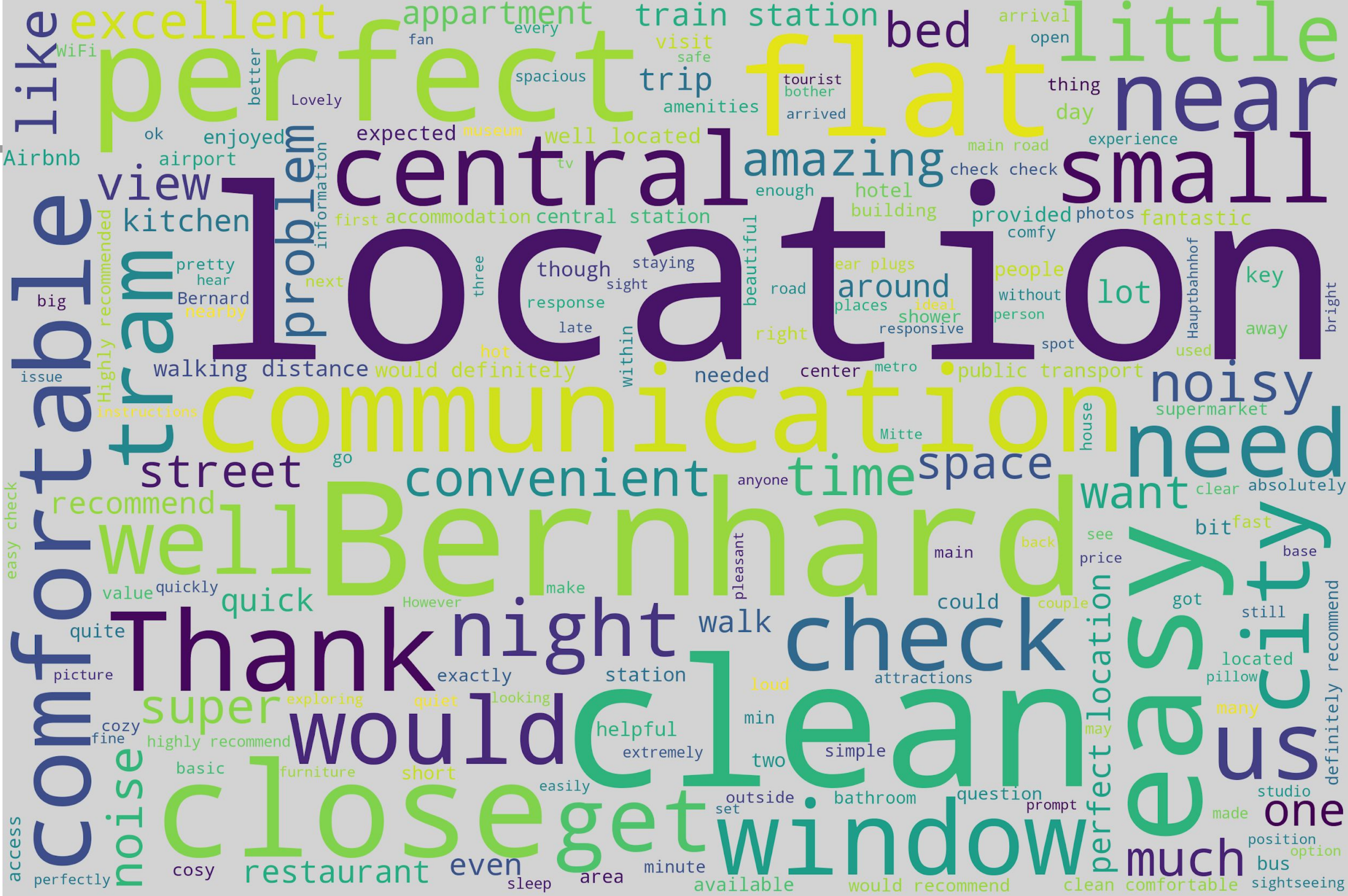
Top 1 listing_id : 517425

number of reviews : 561

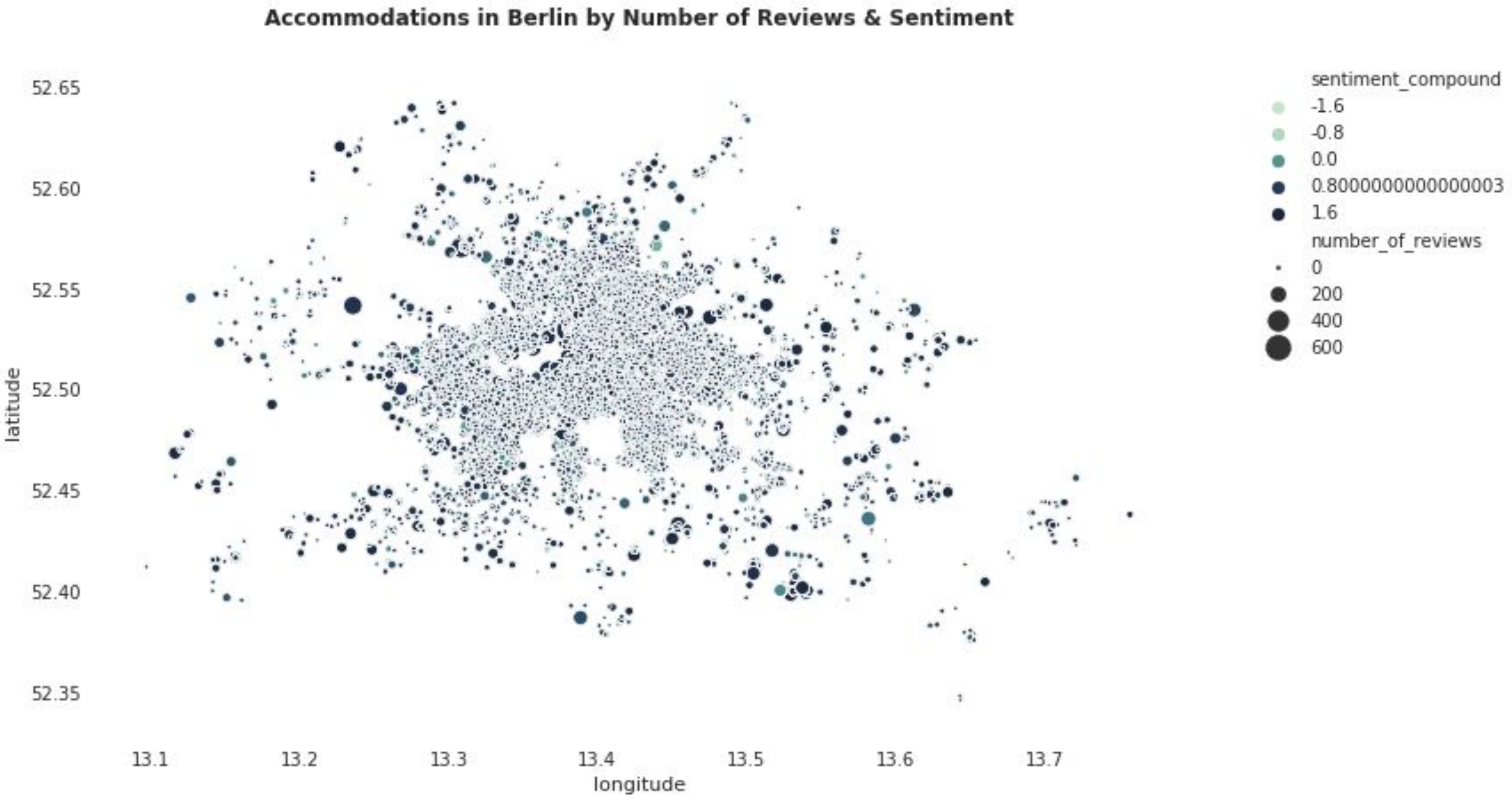


Top 1
listing_id :
26970536

number of
reviews :
551



Analysis



Accommodations in Berlin by Number of Reviews & Sentiment
52.65
52.60
52.55
52.50
52.45
52.40
52.35
latitude
13.1
13.2
13.3
13.4
13.5
13.6
13.7
longitude
sentiment_compound
-1.6
-0.8
0.0
0.8000000000000003
1.6
number_of_reviews
0
200
400
600

Result

- Spark allows you to solve time-consuming task in a short period of time.
    - Spark was 3 times much faster than local on our analysis

- Review data would give useful information to host and customer.
    - They can get sentimental results from each accommodation.
    - They can compare WordCloud among accommodations according to their priorities.

Discussion

- We can use big data much over 100mb combining other cities' data using HDFS.

- Some accommodations have a little number of reviews. So analysis would show biased result.

Q&A