Data Engineering

Examination

Project Report

Prepared by

Manmeet Kumar Chaudhuri

Matriculation No. 11011780

11.01.2019

SRH Hochschule Heidelberg
Fakultät für Information, Medien und Design

Building a Twitter Big Data Pipeline with Kafka, MongoDB and Python

Introduction

Social networks and microblogging sites have become the unparalleled source of both structured and unstructured data. This data is enormous in quantity and also in terms of the useful information they can provide if we process them effectively. This is due to the nature of microblogs on which people post real-time messages about their opinions on a variety of topics, discuss current issues, complain, and express their sentiment for products they use in daily life. In fact, many companies have started analysing such massive amount of important data to get a sense of general sentiment for their product and/or services. Many times, proactive companies study user reactions and reply to the user on social microblogs. This process provides on spot solution to make the user experience better but at large scale its very painful and time-consuming task.

Twitter is a popular real time microblogging service that allows users to share short information known as tweets which were originally limited to 140 characters but on November 7, 2017, this limit was doubled for all languages except Chinese, Japanese, and Korean. Users write tweets to express their opinion about various topics relating to their daily lives. Twitter is an ideal platform for the extraction of general public opinion on specific issues.

The task here is to build a data pipeline for collecting and storing tweets, and perform data analysis on these tweets.

Anatomy of a Tweet

Tweets are short messages, restricted to 280 characters in length. Due to the nature of this microblogging service (quick and short messages), people use acronyms, make spelling mistakes, use emoticons and other characters that express special meanings.

Following is a brief terminology associated with tweets:

- Emoticons: These are facial expressions pictorially represented using punctuation and letters; they express the user's mood.
- Target: Users of Twitter use the "@" symbol to refer to other users on the microblog. Referring to other users in this manner automatically alerts them.
- Hashtags: Users usually use hashtags to mark topics. This is primarily done to increase the visibility of their tweets.

A single tweet contains a lot of information related to users, the text of the tweet, created date of the tweet, the location of the tweet and many more fields. We will use some of the fields to complete the analysis. The key fields of a single tweet are following:

- text: text of the tweet,
- lang: acronym of the tweet language like 'en',
- created_date: date of creation of the tweet,
- favorite_count: number of favourites of the tweet,
- retweet_count: retweets of the tweet,
- place, geolocation: location information, if available,
- user: the full profile of user,

- entities: list of entities like url's, @mentions, #hashtags

We can imagine how these data already allow for some interesting analysis: we can check who is most favourited/ retweeted, who's discussing with who, what are the most popular hashtags and so on.

Building the data pipeline

The task is to create a data pipeline that covers the following aspects:

- Data ingestion

  From Twitter JSON stream:
  All Twitter APIs that return Tweets provide that data encoded using JavaScript Object Notation (JSON). JSON is based on key-value pairs, with named attributes and associated values. These attributes, and their state are used to describe objects.

- Data storage

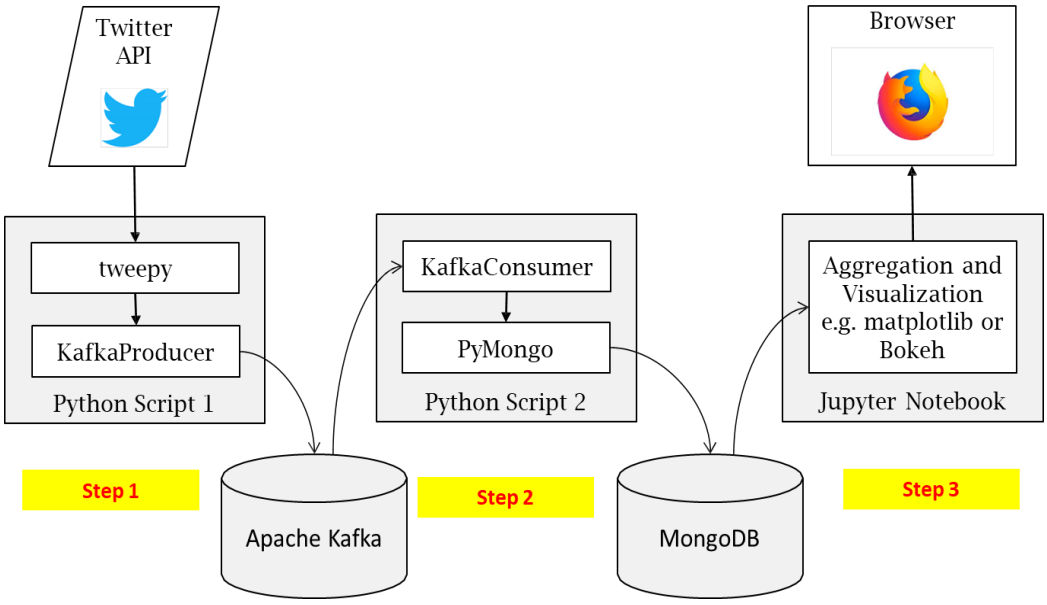
  Using Kafka as a buffer storage:
  Apache Kafka is a highly scalable, fast and fault-tolerant messaging application used for streaming applications and data processing.

  Store data in MongoDB:
  MongoDB is a document database with the scalability and flexibility that one wants with the querying and indexing that one needs.

- Data aggregation and visualization

  Queries with aggregation
  Display of results in tabular format or other visualizations

Diagrammatic representation of the data pipeline



The whole data pipeline has been divided into three steps which are as follows:

- 1st Step: During this process, the data is streamed from the Twitter API and logged in the Kafka
- 2nd Step: Logs from Kafka are consumed and saved into MongoDB, a NOSQL database
- 3rd Step: In this step, some data aggregation and visualisation are performed on the data stored in the MongoDB

We shall go one by one through each of the above-mentioned steps and discuss in detail all the processes/steps undertaken for creating the data pipeline.

1st Step: Streaming data from Twitter API using Kafka

Pre-requisites for this step are:

- Python
- Zookeeper
- Kafka
- Twitter API credentials

Steps undertaken

The following steps were undertaken for creating the data pipeline from Twitter API to Kafka logs:

1. Firstly, we need to create an App on the Twitter API website. We will be able to generate the consumer ley and access token for the Twitter API. These keys are in the following formats.

   consumerKey = 'k9GUw0N0cAjVP7pGZx0s544rB'
   consumerSecret = 'Iz0SIsT3SFyxqtwmeCyZ1NwOlFK7ouC2TnHvDHkpyLvHAmc1Zi'
   accessToken = '1078376125414494208-QyqPbaYJNtkVOaLs8iJnwp5eFGvhmq'
   accessTokenSecret = 'VJE4XPJzeSQSCQxHp4tzzxwzrKZm0xexrd2MZcNCBSA4Q'

2. We would be installing Kafka from this website, https://kafka.apache.org.

3. Kafka uses Zookeeper which is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. Zookeeper can be in downloaded from the following website, https://zookeeper.apache.org/.

4. We would be installing “Tweepy” and “Kafka-python” package in Python environment by running the following command in the command prompt:

   ```
   conda install -c conda-forge tweepy
   conda install -c conda-forge kafka-python
   ```

5. Start Zookeeper form the command prompt window:

   ```
   zkServer
   ```

6. We shall now start Kafka by going into the Kafka directory and the following command:

   ```
   bin\windows\kafka-server-start.bat C:\kafka_2.12-2.1.0\kafka_2.12-2.1.0\config\server.properties
   ```

7. We shall create a topic now. For the purpose of this exercise, we will create the topic “trump” as it is one of the hot and trending topics in the media and Twitter nowadays.

   ```
   bin\kafka-topics.bat --create --zookeeper localhost:2181 --replication-factor 1 --partitions 1 --topic brexit
   ```

   As we are running this exercise from the single machine, we shall keep a replication factor of 1 and just one partition.

8. We will now write a python code for producing the tweets from Twitter. The code is as follows:

   ```
   from tweepy.streaming import StreamListener
   ```

```
from tweepy import OAuthHandler
from tweepy import Stream
from kafka import SimpleProducer, KafkaClient

consumer_key = "k9GUw0N0cAjVP7pGZx0s544rB"
consumer_secret = "Iz0SIsT3SFyxqtwmeCyZ1NwOlFK7ouC2TnHvDHkpyLvHAmc1Zi”
access_token = "1078376125414494208-QyqPbaYJNtkVOaLs8iJnwp5eFGvhmq”
access_token_secret = "VJE4XPJzeSQSCQxHp4tzzxwzrKZm0xexrd2MZcNCBSA4Q”

class StdOutListener(StreamListener):
   def on_data(self, data):
      producer.send_messages("trump", 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(track="trump")
```

9. This code will write relevant tweets to the screen as well as publish the tweets to Kafka. The tweets shall be in JSON format. The format of a tweet is as follows:


```
{
   "created_at": "Wed Jan 02 12:55:18 +0000  2019",
   "id": 1080447440346591232,
   "id_str": "1080447440346591232",
   "text": "RT @dcsportsbog: NFL owners are embarrassed and unhappy about
what's been happening in D.C., a league executive told Sally Jenkins, and
the…",
   "source": "<a href=\"http://twitter.com/download/iphone\"
rel=\"nofollow\">Twitter for iPhone<\/a>",
   "truncated": false,
   "in_reply_to_status_id": null,
   "in_reply_to_status_id_str": null,
   "in_reply_to_user_id": null,
   "in_reply_to_user_id_str": null,
   "in_reply_to_screen_name": null,
   "user": {
      "id": 39539679,
      "id_str": "39539679",
      "name": "steve whatley",
      "screen_name": "umduscuo",
      "location": null,
      "url": null,
      "description": null,
      "translator_type": "none",
      "protected": false,
      "verified": false,
      "followers_count": 138,
      "friends_count": 898,
      "listed_count": 5,
      "favourites_count": 19749,
      "statuses_count": 2971,
      "created_at": "Tue May 12 16:48:46 +0000 2009",
      "utc_offset": null,
      "time_zone": null,
```

```
      "geo_enabled": true,
      "lang": "en",
      "contributors_enabled": false,
      "is_translator": false,
      "profile_background_color": "352726",
      "profile_background_image_url":
"http://abs.twimg.com/images/themes/theme5/bg.gif",
      "profile_background_image_url_https":
"https://abs.twimg.com/images/themes/theme5/bg.gif",
      "profile_background_tile": false,
      "profile_link_color": "D02B55",
      "profile_sidebar_border_color": "829D5E",
      "profile_sidebar_fill_color": "99CC33",
      "profile_text_color": "3E4415",
      "profile_use_background_image": true,
      "profile_image_url":
"http://pbs.twimg.com/profile_images/829070963908161536/NpBnCZjz_normal.jpg"
,
      "profile_image_url_https":
"https://pbs.twimg.com/profile_images/829070963908161536/NpBnCZjz_normal.jpg
",
      "profile_banner_url":
"https://pbs.twimg.com/profile_banners/39539679/1430390438",
      "default_profile": false,
      "default_profile_image": false,
      "following": null,
      "follow_request_sent": null,
      "notifications": null
   },
   "geo": null,
   "coordinates": null,
   "place": null,
   "contributors": null,
   "retweeted_status": {
      "created_at": "Tue Jan 01 21:52:07 +0000 2019",
      "id": 1080220148047511554,
      "id_str": "1080220148047511554",
      "text": "NFL owners are embarrassed and unhappy about what's been
happening in D.C., a league executive told Sally Jenkins,…
https://t.co/ZPfEyOxO4k",
      "source": "<a href=\"http://twitter.com\" rel=\"nofollow\">Twitter Web
Client<\/a>",
      "truncated": true,
      "in_reply_to_status_id": null,
      "in_reply_to_status_id_str": null,
      "in_reply_to_user_id": null,
      "in_reply_to_user_id_str": null,
      "in_reply_to_screen_name": null,
      "user": {
         "id": 15234542,
         "id_str": "15234542",
         "name": "Dan Steinberg",
         "screen_name": "dcsportsbog",
         "location": "ÜT: 38.905475,-77.036856",
         "url": "http://www.washingtonpost.com/blogs/dc-sports-bog/",
         "description": "Sports editor (and sometimes writer) person for
http://washingtonpost.com. Not edgy. Send correspondence to
dan.steinberg@washpost.com.",
         "translator_type": "none",
         "protected": false,
         "verified": true,
         "followers_count": 73010,
         "friends_count": 974,
         "listed_count": 3045,
         "favourites_count": 3142,
         "statuses_count": 111381,
         "created_at": "Wed Jun 25 18:10:25 +0000 2008",
         "utc_offset": null,
```

```
         "time_zone": null,
         "geo_enabled": true,
         "lang": "en",
         "contributors_enabled": false,
         "is_translator": false,
         "profile_background_color": "C0DEED",
         "profile_background_image_url":
"http://abs.twimg.com/images/themes/theme1/bg.png",
         "profile_background_image_url_https":
"https://abs.twimg.com/images/themes/theme1/bg.png",
         "profile_background_tile": false,
         "profile_link_color": "1DA1F2",
         "profile_sidebar_border_color": "C0DEED",
         "profile_sidebar_fill_color": "DDEEF6",
         "profile_text_color": "333333",
         "profile_use_background_image": true,
         "profile_image_url":
"http://pbs.twimg.com/profile_images/644523539425443841/bvmnCm8o_normal.jpg"
,
         "profile_image_url_https":
"https://pbs.twimg.com/profile_images/644523539425443841/bvmnCm8o_normal.jpg
",
         "profile_banner_url":
"https://pbs.twimg.com/profile_banners/15234542/1528839905",
         "default_profile": true,
         "default_profile_image": false,
         "following": null,
         "follow_request_sent": null,
         "notifications": null
      },
      "geo": null,
      "coordinates": null,
      "place": null,
      "contributors": null,
      "is_quote_status": false,
      "extended_tweet": {
         "full_text": "NFL owners are embarrassed and unhappy about what's
been happening in D.C., a league executive told Sally Jenkins, and they
regard Bruce Allen as \"literally, a joke.\" https://t.co/m4QNP3BDmT",
         "display_text_range": [
            0,
            190
         ],
         "entities": {
            "hashtags": [],
            "urls": [
               {
                  "url": "https://t.co/m4QNP3BDmT",
                  "expanded_url": "https://wapo.st/2ToxdFA",
                  "display_url": "wapo.st/2ToxdFA",
                  "indices": [
                     167,
                     190
                  ]
               }
            ],
            "user_mentions": [],
            "symbols": []
         }
      },
      "quote_count": 314,
      "reply_count": 270,
      "retweet_count": 1133,
      "favorite_count": 2277,
      "entities": {
         "hashtags": [],
         "urls": [
            {
```

```
                "url": "https://t.co/ZPfEyOxO4k",
                "expanded_url":
"https://twitter.com/i/web/status/1080220148047511554",
                "display_url": "twitter.com/i/web/status/1…",
                "indices": [
                   116,
                   139
                ]
             }
          ],
          "user_mentions": [],
          "symbols": []
       },
       "favorited": false,
       "retweeted": false,
       "possibly_sensitive": false,
       "filter_level": "low",
       "lang": "en"
    },
    "is_quote_status": false,
    "quote_count": 0,
    "reply_count": 0,
    "retweet_count": 0,
    "favorite_count": 0,
    "entities": {
       "hashtags": [],
       "urls": [],
       "user_mentions": [
          {
             "screen_name": "dcsportsbog",
             "name": "Dan Steinberg",
             "id": 15234542,
             "id_str": "15234542",
             "indices": [
                3,
                15
             ]
          }
       ],
       "symbols": []
    },
    "favorited": false,
    "retweeted": false,
    "filter_level": "low",
    "lang": "en",
    "timestamp_ms": "1546433718178"
}
```


It can be clearly observed that the Tweet JSON has a long list of 'root-level' attributes, including fundamental attributes such as id, created_at, and text. Tweet objects are also the 'parent' object to several child objects. Tweet child objects include user, entities, and extended_entities. Tweets that are geo-tagged will have a place child object.

10. We can also confirm if the tweets are getting published in the relevant topic in Kafka by running the following command:
    `bin/kafka-console-consumer.bat --bootstrap-server localhost:9092 –topic trump --from-beginning`

11. We should get the same output as presented above.

2nd Step: Consuming data from Kafka and storing the data in MongoDB

Pre-requisites for this step are:

- Python
- Zookeeper
- Kafka
- MongoDB

Steps undertaken

1. Start the Zookeeper server and Kafka broker. Zookeeper runs by default on *localhost:2181* and Kafka runs on default on *localhost:9092*.

2. We would be installing "Pymongo" package in Python environment by running the following command in the command prompt:

```
conda install -c anaconda pymongo
```

3. We will call the following dependencies and libraries in the python script

```
from kafka import KafkaConsumer
from pymongo import MongoClient
from json import loads
```

4. Now, we shall create a Kafka consumer. The arguments to be used in the Kafka consumer are provided below:

- The first argument is the topic, *numtest* in our case.
- *bootstrap_servers=['localhost:9092']*: same as our producer
- *auto_offset_reset='earliest'*: one of the most important arguments. It handles where the consumer restarts reading after breaking down or being turned off and can be set either to *earliest* or *latest*. When set to *latest*, the consumer starts reading at the end of the log. When set to *earliest*, the consumer starts reading at the latest committed offset. And that's exactly what we want here.
- *enable_auto_commit=True*: makes sure the consumer commits its read offset every interval.
- *auto_commit_interval_ms=1000ms*: sets the interval between two commits. Since messages are coming in every five second, committing every second seems fair.
- *group_id='counters'*: this is the consumer group to which the consumer belongs. Remember from the introduction that a consumer needs to be part of a consumer group to make the auto commit work.
- The value deserializer deserializes the data into a common json format, the inverse of what our value serializer was doing.

The python script for creating the Kafka consumer is as follows:

```
consumer = KafkaConsumer(
    'trump',
     bootstrap_servers=['localhost:9092'],
     auto_offset_reset='earliest',
     enable_auto_commit=True,
```

```
    group_id='my-group',
    value_deserializer=lambda x: loads(x.decode('utf-8'))) import KafkaConsumer
```

5. The code for connecting to the trump collection in our MongoDB database is provided below:

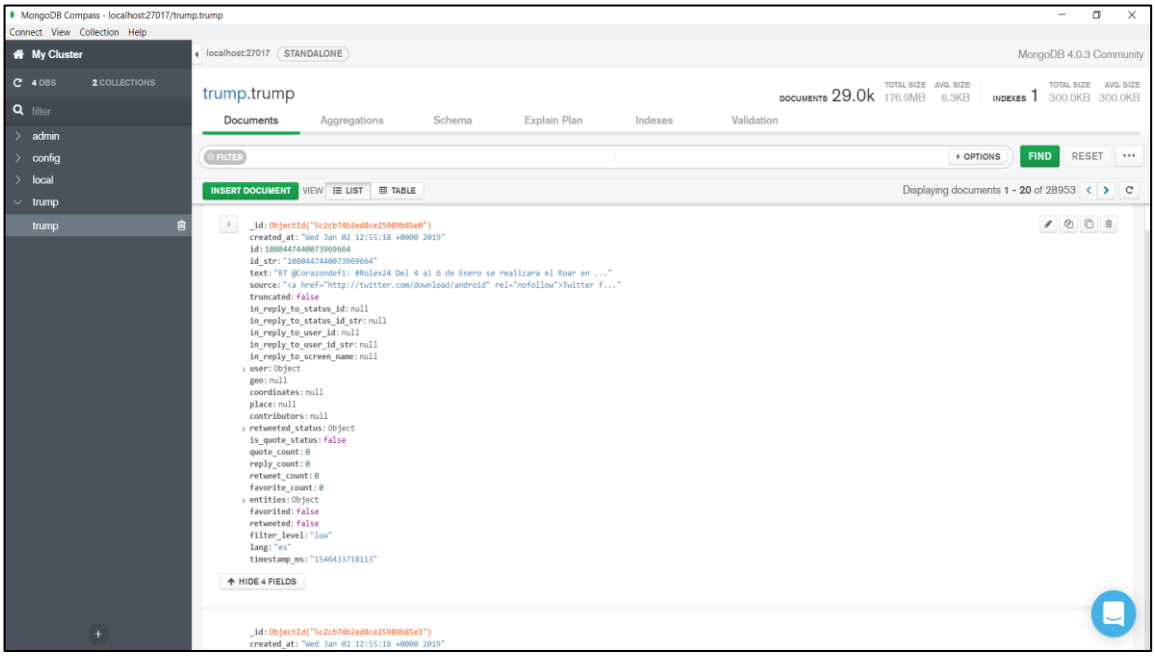
```
client = MongoClient('localhost:27017')
collection = client.numtest.numtest
```

6. We can extract the data from our consumer by looping through it (the consumer is an iterable). The consumer will keep listening until the broker doesn't respond anymore. A value of a message can be accessed with the value attribute. Here, we overwrite the message with the message value.

The next line inserts the data into our database collection. The last line prints a confirmation that the message was added to our collection. Note that it is possible to add call-backs to all the actions in this loop.

```
for message in consumer:
    message = message.value
    collection.insert_one(message)
    print('{} added to {}'.format(message, collection))
```

7. The kafka logs are now being stored into MongoDB. A screenshot of the data captured in the MongoDB Compass, a Graphical User Interface for MongoDB is provided below.



The collection has been named "trump" in MongoDB. This collection contains 28953 documents.

3rd Step: Data aggregation and visualisation

Now, we shall perform some data aggregation queries on the data stored in MongoDB using Jupyter Notebook.

1. Top 10 languages used by Twitter users for topic related to "trump"

In our collection we have 28953 Tweets or documents. First of all, we shall include all the dependencies and libraries in the python script.

```
import numpy as np
import pandas as pd
import re
import warnings
import pymongo
from pymongo import MongoClient
```

We shall then connect to the Mongo client and access the trump collection.

```
myclient = MongoClient("mongodb://localhost:27017/")
mydb = myclient["trump"]
tweets = mydb["trump"]
```

Now, we shall run the aggregation query to find the top 10 languages used by Twitter users for topic related to "trump".

```
cursor=tweets.aggregate([
  {"$group":{"_id": "$lang", "count": {"$sum": 1}}},
  {"$sort":{"count": -1}},
  { "$limit": 10 }])
print(list(cursor)
```

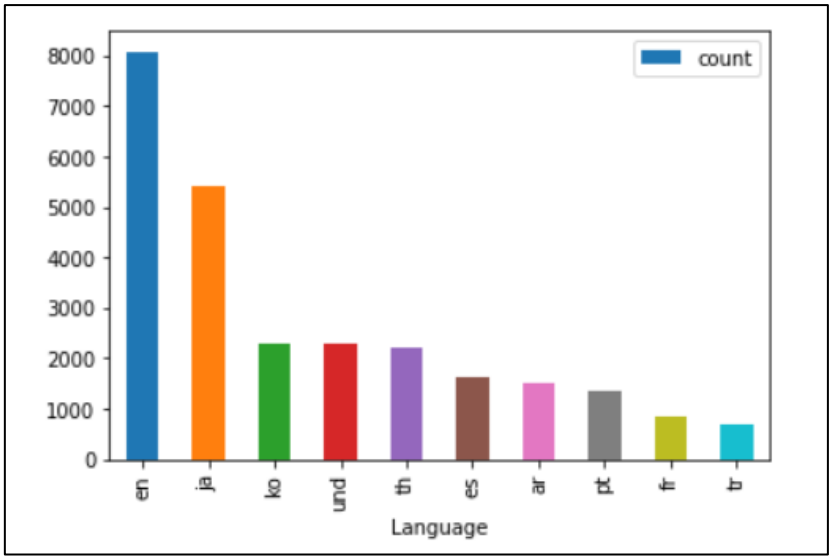
The output in the JSON format is as follows:

```
[{'_id': 'en', 'count': 8089}, {'_id': 'ja', 'count': 5405}, {'_id':
'ko', 'count': 2282}, {'_id': 'und', 'count': 2275}, {'_id': 'th', '
count': 2196}, {'_id': 'es', 'count': 1647}, {'_id': 'ar', 'count':
1504}, {'_id': 'pt', 'count': 1349}, {'_id': 'fr', 'count': 854}, {'
_id': 'tr', 'count': 701}]
```

It can be seen that English is the most common language used by the users, followed by Japanese language. Then, Korean language is the third most used language.

We can also plot these results in graphical representation using the "matplotlib" in Python. The script and the plot are as follows:

```
import matplotlib.pyplot as plt
dfr=pd.DataFrame.from_records(cursor)
plot1=dfr.plot(x="_id", y="count", kind="bar")
plot1.xaxis.set_label_text("Language")
```



2. Top 10 hashtags used by Twitter users for topic related to “trump”

We would run the following aggregation query to find the top 10 hashtags used by Twitter users for topic related to “trump”.

```
cursor1=tweets.aggregate([
{"$unwind": "$entities.hashtags"},
{ "$group": {
"_id": "$entities.hashtags.text", "tagCount": {"$sum": 1}}},
{ "$sort": {"tagCount": -1}},
{ "$limit": 10 }])
dfr2=pd.DataFrame.from_records(cursor1)
dfr2
```

The output in the tabular format is as follows:

	_id	tagCount
0	방탄소년단	160
1	BTS	152
2	ยื่อน่ารัก	125
3	GOT7	65
4	威神V	62
5	WeiShenV	61
6	태형	61
7	TEN	60
8	EXO	59
9	뷔	55

3. Top 10 user countries with the greatest number of followers

The aggregation query used to find the top 10 user countries with the greatest number of followers is as follows:

```
cursor2=tweets.aggregate([
   {"$group":{"_id": "$place.country", "total_followers":{"$sum":"$user.followers_count"}}},
   {"$sort":{"total_followers": -1}},
   {"$limit": 10}])
print(list(cursor2))
```

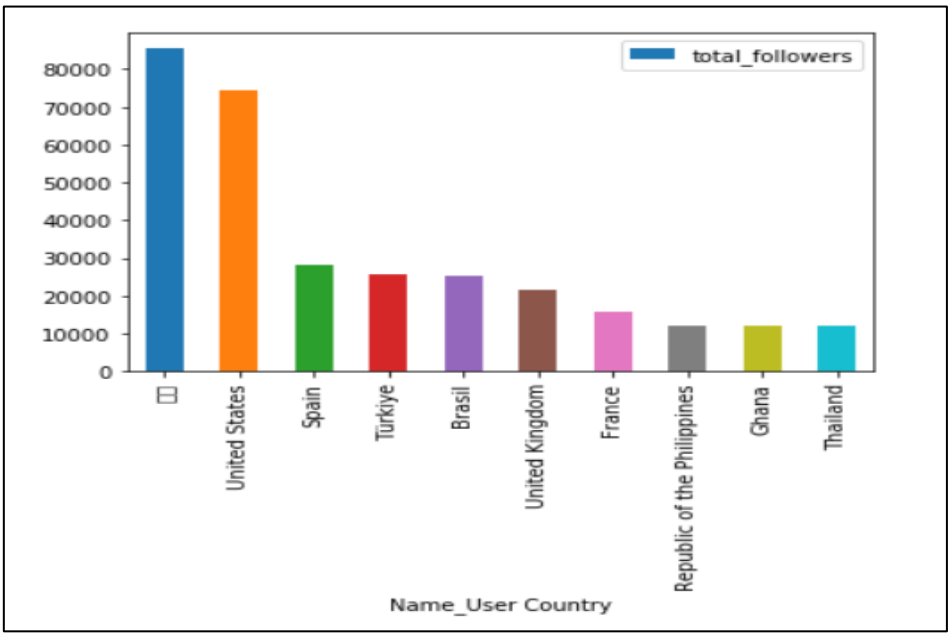
The output is as follows:

```
[{'_id': '日本', 'total_followers': 85479}, {'_id': 'United States',
'total_followers': 74558}, {'_id': 'Spain', 'total_followers': 28292
}, {'_id': 'Türkiye', 'total_followers': 25650}, {'_id': 'Brasil', '
total_followers': 25462}, {'_id': 'United Kingdom', 'total_followers
': 21574}, {'_id': 'France', 'total_followers': 15635}, {'_id': 'Rep
ublic of the Philippines', 'total_followers': 12229}, {'_id': 'Ghana
', 'total_followers': 12081}, {'_id': 'Thailand', 'total_followers':
11885}]
```

It can be seen that Japanese users have the most followers i.e. 85479. Users from USA have a follower base of 74558 and then, the users of Spain have a follower base of 28292. In 10th place, we have Thailand with a total of 11885 followers.

We can also plot these results in graphical representation using the matplotlib in Python. The script and the plot are as follows:

```
dfr1=pd.DataFrame.from_records(cursor2)
plot2=dfr1n.plot(x="_id", y="total_followers", kind="bar")
plot2.xaxis.set_label_text("Total followers")
```



4. Top 10 users with the greatest number of retweets

The aggregation query used to find the top 10 users with the greatest number of retweets is as follows:

```
cursor3=tweets.aggregate([
   {"$group":{"_id": "$retweeted_status.user.name",
"total_retweets":{"$sum":"$retweeted_status.retweet_count"}}},
   {"$sort":{"total_retweets": -1}},
   { "$limit": 10 }])
print(list(cursor3))
```

The output is as follows:

```
[{'_id': '방탄소년단', 'total_retweets': 11470247}, {'_id': 'スターライトステージ', 'total_retweets': 5144690}, {'_id': 'c', 'total_retweets': 2976406}, {'_id': 'Jordan Ireland', 'total_retweets': 1851077}, {'_id': 'David Tra', 'total_retweets': 1530645}, {'_id': 'juany', 'total_retweets': 1400217}, {'_id': 'Maria Fernanda', 'total_retweets': 997301}, {'_id': 'BTS_official', 'total_retweets': 959946}, {'_id': 'Aquiles', 'total_retweets': 953754}, {'_id': 'BamBam', 'total_retweets': 895634}]
```

It can be observed that the user with most retweets is a Korean user named "방탄소년단" [total retweets = 11470247]. The second most retweeted user "スターライトステージ" is from Japan [total retweets = 5144690].

5. Sentiment analysis of the Tweets

For this exercise, we shall be using TextBlob library. TextBlob is a Python library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more. First of all, we shall include all the dependencies and libraries in the python script.

```
from pymongo import MongoClient
import pandas as pd
import json
from textblob import TextBlob
from bson.json_util import dumps
from bson import json_util, ObjectId
from pandas.io.json import json_normalize
import reimport pandas as pd
import re
```

We shall then connect to the Mongo client and access the trump collection. The tweets are then passed into TextBlob package for sentiment analysis. A polarity score of something in between +1 and -1 is generated for each of the tweets. A negative score signifies negative sentiment associated with the tweet. A score of 0 means that the users have neutral view of the tweet. A positive score means that that users share a positive opinion of the tweet. The remaining code is as follows:

```
#Connect to Mongo database
connection=MongoClient("mongodb://localhost:27017/")
db=connection.trump
names=db.trump
mongo_data = names.find()
sanitized = json.loads(json_util.dumps(mongo_data))
```

```
for i in sanitized:

# pass tweet into TextBlob
    tweet = TextBlob(i["text"])
    print(tweet)

# output sentiment polarity
    print(tweet.sentiment.polarity)

# determine if sentiment is positive, negative, or neutral
    if tweet.sentiment.polarity < 0:
      sentiment = "negative"
    elif tweet.sentiment.polarity == 0:
        sentiment = "neutral"
    else:
      sentiment = "positive"
```

The result of the sentiment analysis is as follows [a select few tweets have been displayed here]:

```
RT @imillhiser: This is your friendly reminder that the entire Repub
lican Party opposes democracy and would be happy to see an authorita
ria…
0.39166666666666666
RT @dcsportsbog: NFL owners are embarrassed and unhappy about what's
been happening in D.C., a league executive told Sally Jenkins, and t
he…
-0.6
RT @dmightyangel: Whatever happened to USA during Obama's regime wil
l never be credited to Governors of States in America, what is happe
nin…
0.0
RT @nitin_nitinsr: The future is now! We explore how fully #automate
d #robotic enterprises may soon become a reality: https://t.co/Sql94
Qyc…
-0.05
11/04/18 second row for Harry at the O2Arena. Had the time of my lif
e. It was a very emotional day because I have s… https://t.co/lcCJMK
p3Zf
0.0
RT @dnlbrns: Watching CNN's entire roster get trashed on NYE is quic
kly becoming one of my favorite traditions https://t.co/79wXnVcSwV
0.31666666666666665
Boho Style Earrings, Copper Bead Earrings, Gifts Under 50, Art Nouve
au Style, Romantic Earrings, Boho Chic Earrings… https://t.co/4prFtK
59Uv
0.0
```

Sentiment analysis of tweets of requisite topics could be very insightful for brands/companies to gauge the real time perception of customers with regard to the respective brands/companies/entities, etc.

Some difficulties/challenges faced in the project

1. Integrating python script with Twitter API and Kafka, Kafka and MongoDB

Identifying the requisite python libraries and syntaxes were challenging tasks. With the help of resources available over the internet and a lot of practice, required python scripts were developed for usage with Twitter API, Kafka and MongoDB.

2. Usage of “matplotlib” library in python

Requisite literature was reviewed to understand the usage of “matplotlib” library in python. Matplotlib was used for plotting graphs for data analysis. Matplotlib is a very versatile data visualisation tool. Further expertise needs to be developed on this tool.

3. Aggregation functions in MongoDB

Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result.

MongoDB Manual was frequently referred to for help with the aggregation queries in this project. It can be summarised that constant inquisitiveness and perseverance have been the key to overcoming most of the difficulties and challenges in the project.

References

1. Wisdom, Vivek and Gupta, Rajat. *An introduction to Twitter Data Analysis in Python*. Research Gate, 2016. <https://www.researchgate.net/publication/308371781_An_introduction_to_Twitter_Data_Analysis_in_Python>
2. Sarlan, Aliza and Shuib, Basri. *Twitter* Sentiment *Analysis*. Research Gate, 2014. < https://www.researchgate.net/publication/301408174_Twitter_sentiment_analysis>
3. Kafka Development core team. *Kafka 2.1 Documentation*. < https://kafka.apache.org/documentation/>.
4. Mongo Development team. "*MongoDB Docs*." <https://docs.mongodb.com/>
5. Dorpe, Steven. *Kafka-Python explained.* 2018. <https://towardsdatascience.com/kafka-python-explained-in-10-lines-of-code-800e3e07dad1>
6. Python Development team. *Python 3.7.2 documentation.* < https://docs.python.org/3/>