Phase2

1. Agenda for the Phase 2.

In phase 1, we use Twitter-Streaming API to collect the tweets data as json file. Then we collect the results in Hadoop. In Phase 2, we try to store the data in spark SQL and analyze the data using the platform. The data is collected by the same method as the phase 1.

We use python to connect to the Twitter-Streaming API to collect the data related to the most popular two candidates for this year's president election. So we request data contains "Trump" and "Biden"and "Kanye". The data size is 10,000. Also we collect a small size data set which contains 30000 tweets to test the environment before we run the large data set. There are ten analyze queries that we try to use to analyze the data. They are as follows :

1) How many people tweet about Trump ad Biden and Kanye in general?
2) What kind of hashtags are the most popular when tweeting about trump and Biden?
3) How popular the hashtag is when tweets about the candidates?
4) What kind of URLS are the most popular when tweeting about trump and Biden?
5) The popularity of the URLs related to Trump and Biden which base on the amount of the user followers to the URL.
6) What is the language distribution among tweets when tweets about candidates?
7) The popularity of the tweets among different languages.
8) How is the location distribution about the popularity like?
9) Changing of the popularity over timeline.
10) Change of the supportive language difference between candidates based on timeline.

2. Implementation

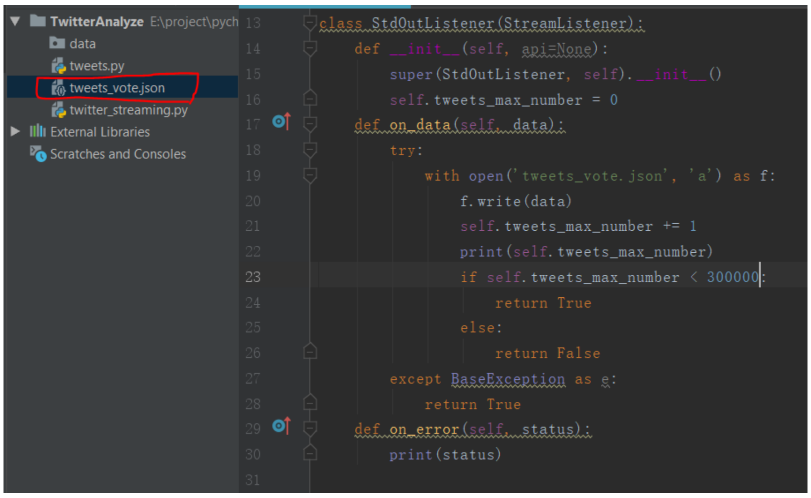
This project is to analyze the tweeter streaming data to get some interesting information about the popularity of the candidates for 2020 president election. We use python to collect the data and then use Spark SQL to do the query and analyze. The Spark SQL is based on Scala which shows high performance in both execute efficiency and stability. The data has been read from the local and then load into spark for the further queries.

So this project has been implies by several steps. First, request tweets streaming data from the website. Load the data and store the data in local as Json file. Second, load the data into spark and use spark SQL to do the query.

After the data process, we store data in local and we choose to store the data in csv format for doing the further exploration. Third, we extract information from the processed data using python. Python has large amount of libraries which can help us virtualize the data easier and more concise. So the whole process is like getting in touch with the data and store the data, then analyze and getting information from the data, finally display the data and extract the useful information.

Step 1: Get json data

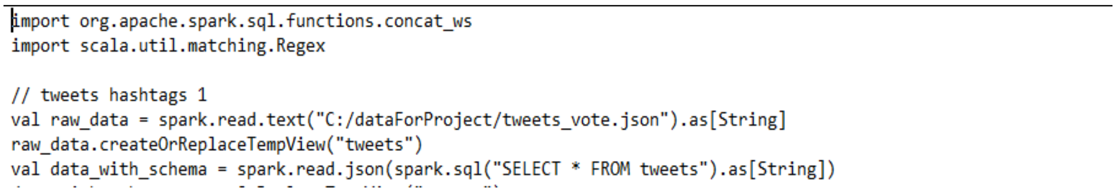
With the help of twitter API, we get 300000 tweets and store it as tweets_vote,json file in the local. And the data contains tweets related to hot candidates for the present election. The implementation has been talked about in the first phase.



```python
class StdOutListener(StreamListener):
    def __init__(self, api=None):
        super(StdOutListener, self).__init__()
        self.tweets_max_number = 0
    def on_data(self, data):
        try:
            with open('tweets_vote.json', 'a') as f:
                f.write(data)
                self.tweets_max_number += 1
                print(self.tweets_max_number)
                if self.tweets_max_number < 300000:
                    return True
                else:
                    return False
        except BaseException as e:
            return True
    def on_error(self, status):
        print(status)
```

Step 2: Analyze json data

This part of the project has been finished using spark SQL. First we use spark to read the raw data and then we operate on the temperature view on the raw data, Now the data has been loaded into spark platform. And then we select the text from the tweets.



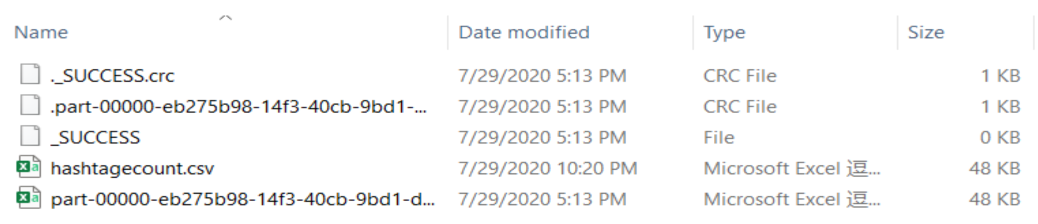
```scala
import org.apache.spark.sql.functions.concat_ws
import scala.util.matching.Regex

// tweets hashtags 1
val raw_data = spark.read.text("C:/dataForProject/tweets_vote.json").as[String]
raw_data.createOrReplaceTempView("tweets")
val data_with_schema = spark.read.json(spark.sql("SELECT * FROM tweets").as[String])
```

But the raw data will contain null value, which is not what we want. So we deal with this data with filters. After filter the null data, we get the filtered data. For the processing of the data, we repeatedly create temporary views and replace it with the views of same name, in order to save the space for the storing the views. In the first query, we try to extract the hashtag and use map reduce function to count the hash tags. From the reduce count result, we can get the top hashtags people tweets about the election.

In the map function, we use regex pattern to map the text content for each row.

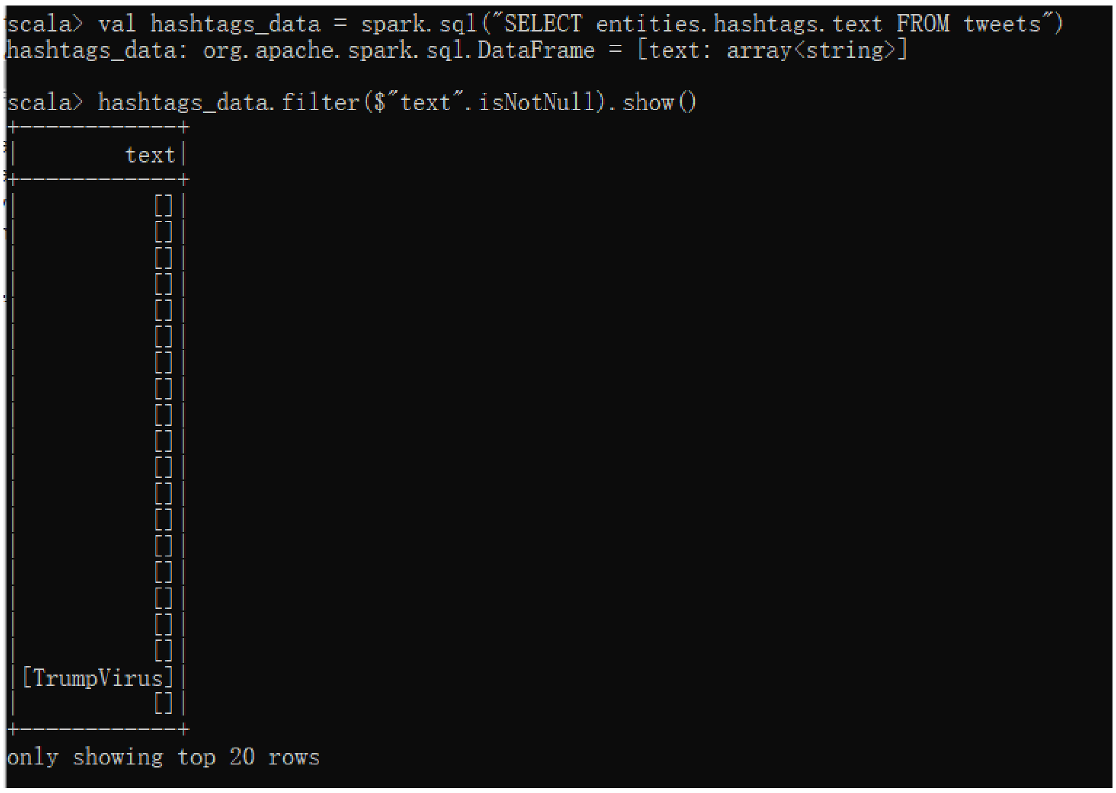
```
val patternForRegex = new Regex("^\\s*[A-Za-z0-9]+(?:\\s+[A-Za-z0-9]+)*\\s*$")

val mapped_data = data.flatMap(sqlRow => (patternForRegex findAllIn sqlRow(0).toString).toList)
val reduced_data = mapped_data.groupByKey(_.toLowerCase)
val counts = reduced_data.count().orderBy($"count(1)".desc)
```

The final output file is a csv file contains two columns, one is hashtag and the other one is the count for the hashtags. The count represent for the popularity in some degree. The local path contains the log file and the output file.



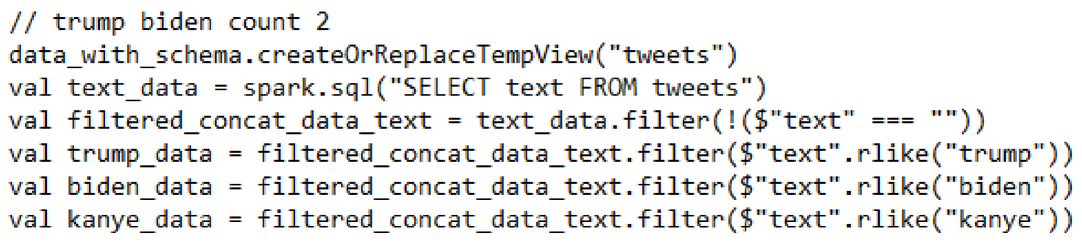
Name	Date modified	Type	Size
._SUCCESS.crc	7/29/2020 5:13 PM	CRC File	1 KB
.part-00000-eb275b98-14f3-40cb-9bd1-...	7/29/2020 5:13 PM	CRC File	1 KB
_SUCCESS	7/29/2020 5:13 PM	File	0 KB
hashtagecount.csv	7/29/2020 10:20 PM	Microsoft Excel 逗...	48 KB
part-00000-eb275b98-14f3-40cb-9bd1-d...	7/29/2020 5:13 PM	Microsoft Excel 逗...	48 KB

The Scala file is run in the windows terminal, which set up the spark and Hadoop environment. The version for spark is 2.4.4 and the Scala version is 2.11.12. The path for HDFS has been recognized by the system environment. And we use the environment to run small scale file first. It works well. The result for the hashtags query is like the following.



```
scala> val hashtags_data = spark.sql("SELECT entities.hashtags.text FROM tweets")
hashtags_data: org.apache.spark.sql.DataFrame = [text: array<string>]

scala> hashtags_data.filter($"text".isNotNull).show()
+------------+
|        text|
+------------+
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|          []|
|[TrumpVirus]|
|          []|
+------------+
only showing top 20 rows
```

We also search for the count of candidates separately. Like we select the tweets contains the name of the different candidates here.

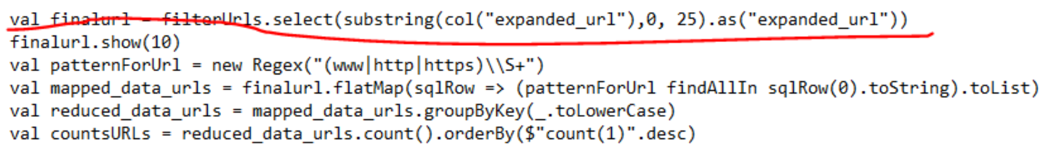


```
// trump biden count 2
data_with_schema.createOrReplaceTempView("tweets")
val text_data = spark.sql("SELECT text FROM tweets")
val filtered_concat_data_text = text_data.filter(!($"text" === ""))
val trump_data = filtered_concat_data_text.filter($"text".rlike("trump"))
val biden_data = filtered_concat_data_text.filter($"text".rlike("biden"))
val kanye_data = filtered_concat_data_text.filter($"text".rlike("kanye"))
```

And then we count the tweets about them. So that we can compare there popularity or their public attention based on the data we get. Then we output the data as a csv file with two column formats. One column is for name and the other one is for the counts of the tweets related the names.

For each query, we set up a separate query and stores as a different output file. We do not explain the details one by one here. However we will contain all the logfile in the attached files.

One thing we have to notice about during the query process is that when we extract the url, the text for the value would be long. So it will be a problem when we want to show the text in the further virtualization. So we use substring here to solve the problem.



```
val finalurl = filterUrls.select(substring(col("expanded_url"),0, 25).as("expanded_url"))
finalurl.show(10)
val patternForUrl = new Regex("(www|http|https)\\S+")
val mapped_data_urls = finalurl.flatMap(sqlRow => (patternForUrl findAllIn sqlRow(0).toString).toList)
val reduced_data_urls = mapped_data_urls.groupByKey(_.toLowerCase)
val countsURLs = reduced_data_urls.count().orderBy($"count(1)".desc)
```

Other than these queries, we also do a query that shows how popularity change based on a timeline. In order to query for the change of the distribution of the popularity which shows which language people use more often to tweet about the election, we select out the create time and the user followers counts and the language from the row data scheme.

After running the whole scala file , we get the output data and move to the next step.

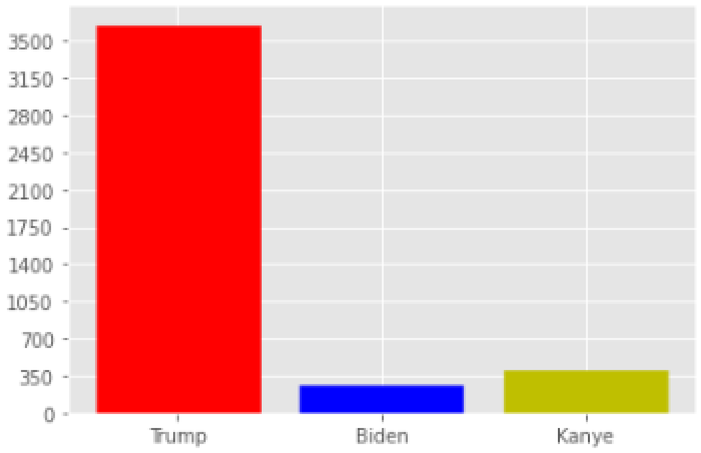
Step 3: virtualization of the data

For virtualization, we use python to display the data. And we use the online editor Colab. Python is a good language to do the virtualization for it has a powerful library behind it. And it is very convenient for us to use the online editor to do the analyze.

We have design 10 charts here for varies type of purpose.
First one, we design a bar chart to show the counts of tweets about the

candidates.

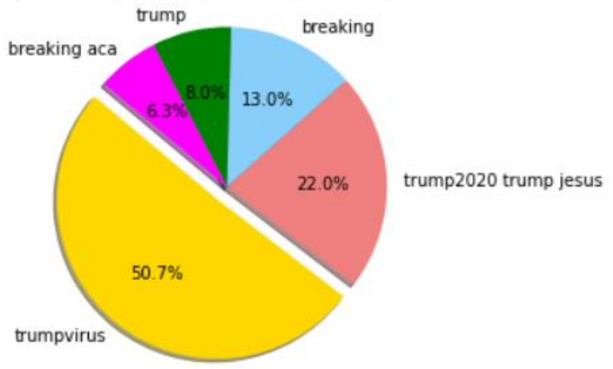
```
x = np.arange(len(value))
yinterval = 350

y_axis = [int(numeric_string) for numeric_string in count]
colors = ['gold', 'lightcoral', 'lightskyblue','green','magenta']
plt.yticks(range(0,max(y_axis),yinterval))


# x_pos = [i for i, _ in enumerate(value)]

barlist = plt.bar(x, y_axis)
barlist[0].set_color('r')
barlist[1].set_color('b')
barlist[2].set_color('y')

plt.xticks(x, value)
# plt.yticks(color='black')

plt.show()
```



From the chart we can see that Trump is the one attract most of the attention from public. Next we analyze the top5 hashtags to figure out what people care most. We use a pie chart to show the results.

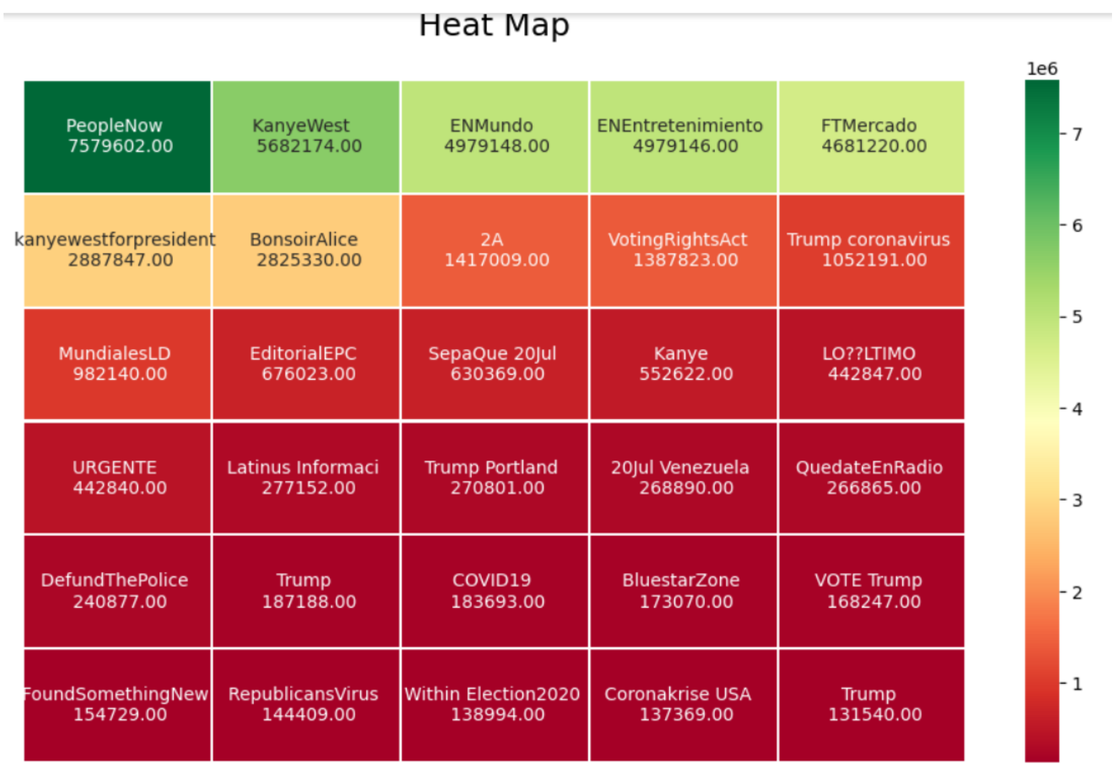
```
with open('/hashtagecount.csv') as csvDataFile:
        csvReader = csv.reader(csvDataFile)
        i=0
        for row in csvReader:
            if( i < 5 ):
                value.append(row[0])
                count.append(row[1])
            i += 1

colors = ['gold', 'lightcoral', 'lightskyblue','green','magenta']
explode = (0.1, 0, 0, 0, 0)

plt.pie(count, explode=explode, labels=value, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=140)

plt.axis('equal')
plt.show()
```

```
['trumpvirus', 'trump2020 trump jesus', 'breaking', 'trump', 'breaking aca']
['1064', '463', '273', '168', '132']
```



Then we use heatmap to display the popularity of the hashtags which prople talked most. The heatmap shows the text of the hashtags and the counts of the user followers for these hashtags.

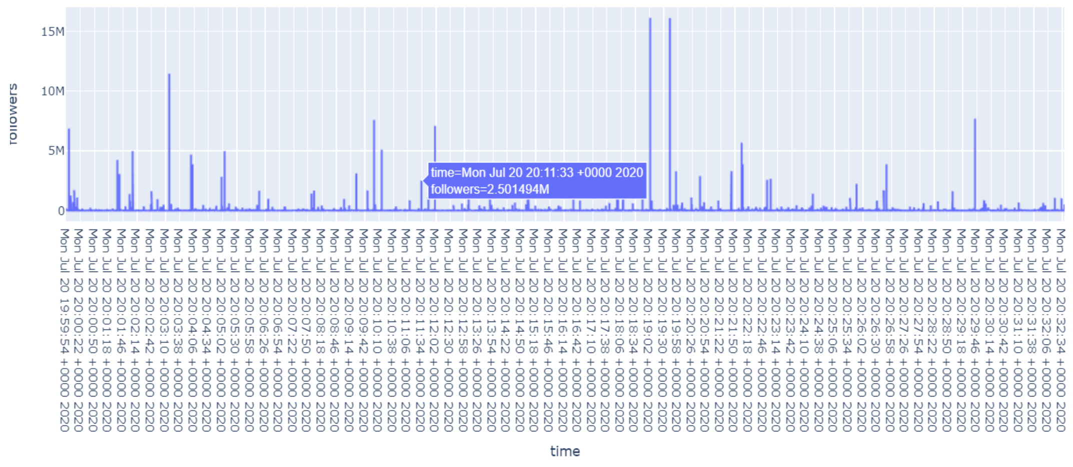
```
labels = (np.asarray(["{0} \n {1:.2f}".format(value, count)
for value, count in zip(value.flatten(), count.flatten()) ])).reshape(6,5)

fig, ax=plt.subplots(figsize=(12,7))

title="Heat Map"

plt.title(title,fontsize=18)
ttl=ax.title
ttl.set_position([0.5,1.05])

ax.set_xticks([])
ax.set_yticks([])

ax.axis('off')

sns.heatmap(result,annot=labels,fmt="",cmap='RdYlGn',linewidths=0.8,ax=ax)

plt.show()
```



The heat map shows the popularity of the hashtags based on different colors. And the label in the right shows the corresponding level for these colors. We can say that people and Kanye are the most hot topics now.

Then we use plot library to show how the language distribution changed based on the time line.

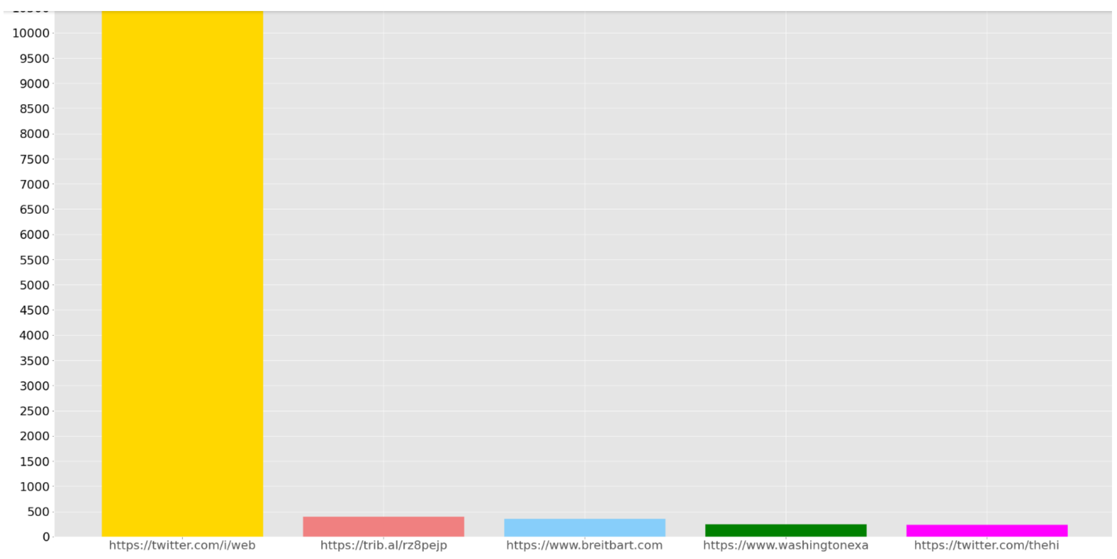
```
import plotly.express as px

import pandas as pd
df = pd.read_csv('/timeline_of_followers.csv')

fig = px.line(df, x='time', y='followers')

fig.show()
```



In these timeline, it is more meaningful if we can get data among days. But due to the computer ability, we only use data within hours. We can see that the x axis show the exact time and the graph shows the change of the popularity. And also when u move your mouse on the line, you can see the language this point represents and the count it represents.

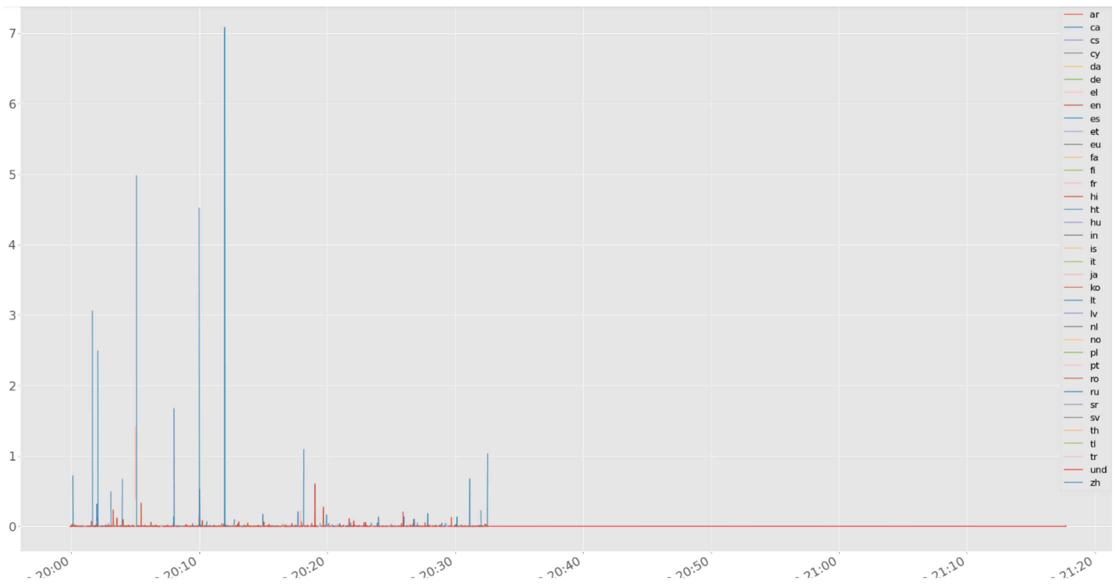
Also we etract the url data and analyze the top five url related.

```
x  =  np.arange(len(url))
yinterval  =  500
y_axis  =  [int(numeric_string)  for  numeric_string  in  count]

plt.yticks(range(0,max(y_axis),yinterval))

barlist  =  plt.bar(x,  y_axis)
barlist[0].set_color('gold')
barlist[1].set_color('lightcoral')
barlist[2].set_color('lightskyblue')
barlist[3].set_color('green')
barlist[4].set_color('magenta')

#  Plot  the  figure.

plt.xticks(x,  url)
plt.yticks(color  =  'black')
fig=  plt.gcf()
plt.legend(prop={"size":16.4})
plt.rc('axes',  titlesize=20)
fig.set_size_inches(35,  20)
plt.show()
```



Also we use the pyplot to show show the timeline of the change of the languages, which represent the attention among different language users.

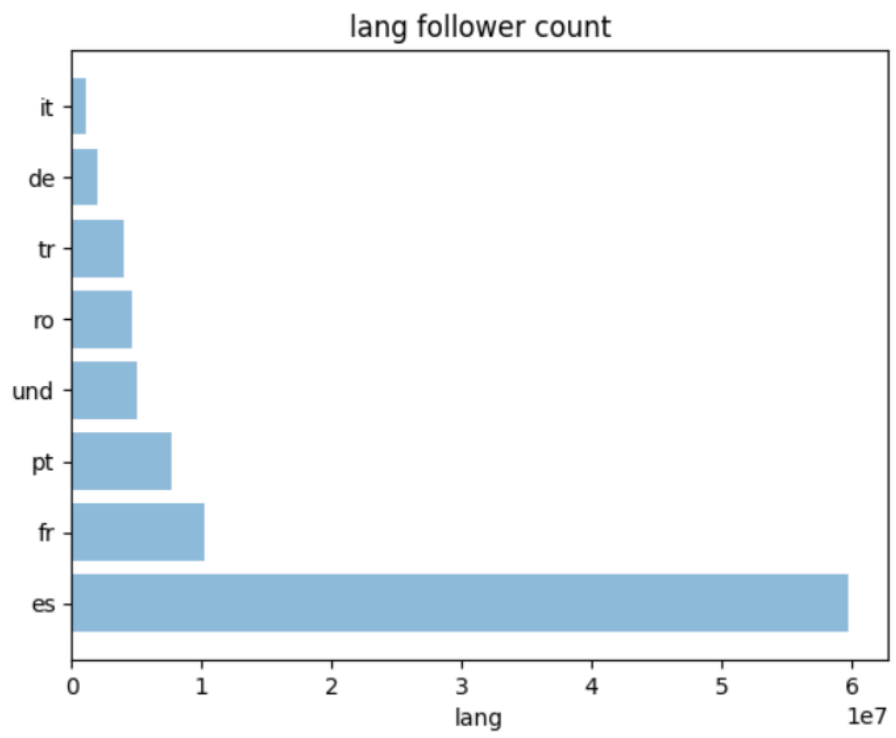
```
import  io
import  pandas  as  pd
import  matplotlib.pyplot  as  plt

df  =  pd.read_csv('/timelineOfLang.csv')
df["time"]  =  pd.to_datetime(df["time"])
df.pivot_table(index="time",  columns="lang",  values="followers").plot()
print(df.pivot_table(index="time",  columns="lang",  values="followers"))
fig=  plt.gcf()
plt.legend(prop={"size":16.4})
plt.rc('axes',  titlesize=20)
fig.set_size_inches(35,  20)
plt.show()
```



In this plot, the label in the right of the chart shows the name of different languages and different color represent different language. In the chart we can the line change and figure out the language distribution easily based on the varies colors.

Next we use the pyplot to show the most pupolar language distribution.

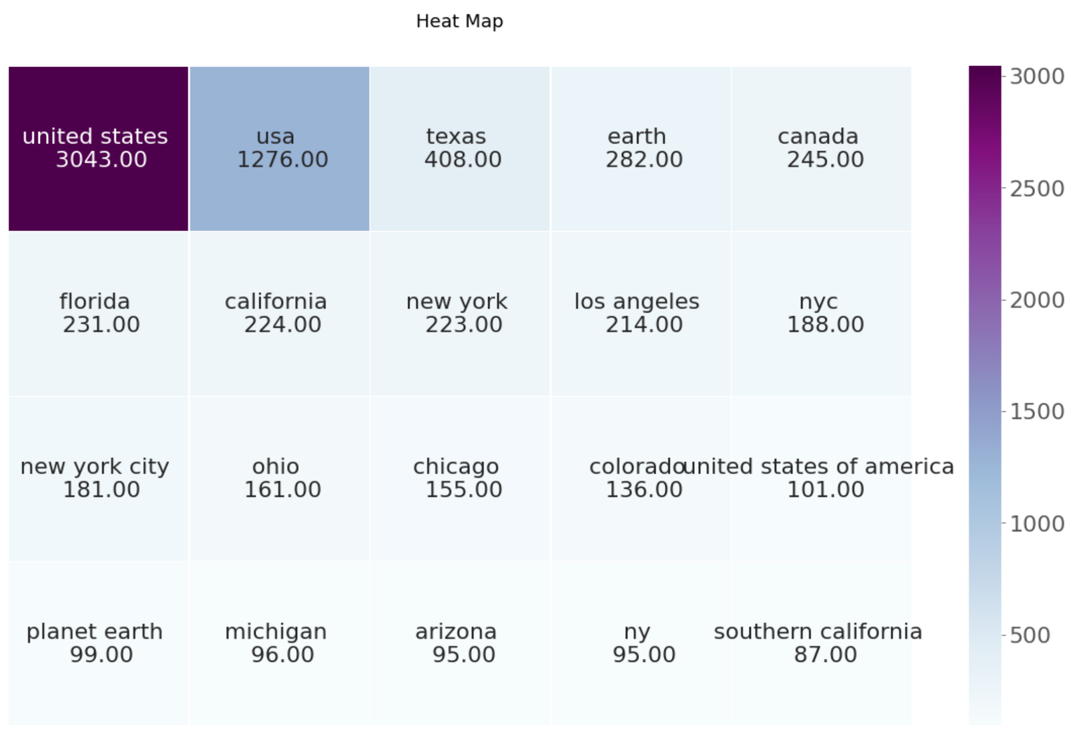
```
y_pos = np.arange(len(value))
x_axis = [int(numeric_string) for numeric_string in count]


plt.barh(y_pos, x_axis, align='center', alpha=0.5)
plt.yticks(y_pos, value)
plt.xlabel('lang')
plt.title('lang follower count')

plt.show()
```



We move out the top language in the data because English user is a super large group in tweets and it is no doubt. The only thing we need to analyze is to figure out how the rest language popularity distribute.

Then we analyze the location information about the tweets. We use heatmap here to compare the distribution of location.

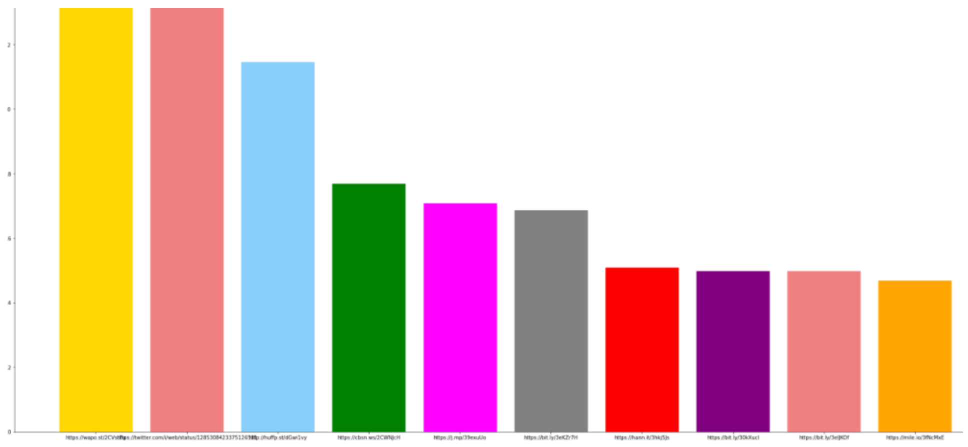
```
labels = (np.asarray(["{0} \n {1:.2f}".format(value, count)
for value, count in zip(value.flatten(), count.flatten()) ])).reshape(4,5)

fig, ax=plt.subplots(figsize=(20,12))

title="Heat Map"

plt.title(title,fontsize=18)
ttl=ax.title
ttl.set_position([0.5,1.05])

ax.set_xticks([])
ax.set_yticks([])

ax.axis('off')

sns.heatmap(result,annot=labels,fmt="",cmap='BuPu',linewidths=0.50,ax=ax)

plt.show()#
```



The name of the location and the number shows clear on the plot and also the color indicate the most frequent locations at which people tweets about the election most.

we use a bar chart to figure out the popularity of url related based on the count of the url has been tweeted.

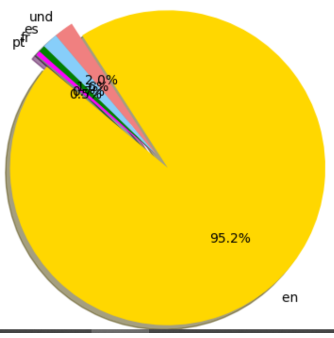
```
x = np.arange(len(value))
yinterval = 350

y_axis = [int(numeric_string) for numeric_string in count]
# plt.yticks(range(0,max(y_axis),yinterval))


# x_pos = [i for i, _ in enumerate(value)]

barlist = plt.bar(x, y_axis)
# barlist[0].set_color('r')
# barlist[1].set_color('b')

plt.xticks(x, value)
# plt.yticks(color='black')
fig= plt.gcf()
plt.legend(prop={"size":16.4})
plt.rc('axes', titlesize=20)
fig.set_size_inches(35, 20)
barlist[0].set_color('gold')
barlist[1].set_color('lightcoral')
barlist[2].set_color('lightskyblue')
barlist[3].set_color('green')
barlist[4].set_color('magenta')
barlist[5].set_color('grey')
barlist[6].set_color('red')
barlist[7].set_color('purple')
barlist[8].set_color('lightcoral')
barlist[9].set_color('orange')
plt.show()
```



The different color shows varies types of the url and the labels show the content of the urls.

We use a pie chart to figure out that English language user is still the largest part in users who tweets. It is a more than 95 percentage.

```
import  matplotlib.pyplot  as  plt
import  csv

value  =  []
count  =  []
with  open('/languagecount.csv')  as  csvDataFile:
        csvReader  =  csv.reader(csvDataFile)
        i=0
        for  row  in  csvReader:
            if(  i  <  5  ):
                value.append(row[0])
                count.append(row[1])
            i  +=  1

colors  =  ['gold',  'lightcoral',  'lightskyblue','green','magenta']
explode  =  (0.1,  0,  0,  0,  0)

plt.pie(count,  explode=explode,  labels=value,  colors=colors,
autopct='%1.1f%%',  shadow=True,  startangle=140)

plt.axis('equal')
plt.show()
```



3. System

For this project, our basic operating is windows 10. We use Scala for query analyze and the version for Scala is 2.11.12. The spark has been implied on windows and the version for this spark is 2.4.4. Then we use python book for virtualization in google collab.

We have tested the environment using a small test set and it turns out that the environment is good working. Then we imply the real dataset.

For font end analyze we plan to use Tableau.

.