**Part A - Analysis of Coronavirus Tweets and Political Tweet Traits**

|  |  |  |  |
| --- | --- | --- | --- |
| **Student** | **Email** | **Github ID** | **Class ID** |
| Acikgoz, Mehmet | ma96f@mail.umkc.edu | acikgozmehmet | 1 |
| Wolfe, Jonathan Andrew | jawhf4@mail.umkc.edu | JAWolfe04 | 17 |

**Team Details - Team 1A**

**Introduction**

The advances in science and technology have made the modern lifestyle more interactive with people through social media such as Twitter. With the advances in technology, people started to be more active in Twitter by sharing their ideas, criticism, support. It has become a new way for people to communicate among themselves in addition to actual use cases in business, marketing, and education etc.

Besides this personal use of Twitter, it also provides valuable information about the global threats such as COVID19 pandemic. We believe that Twitter messages provide instant pictures of threat that will shed light to unseen part of the pandemic.

**Background**

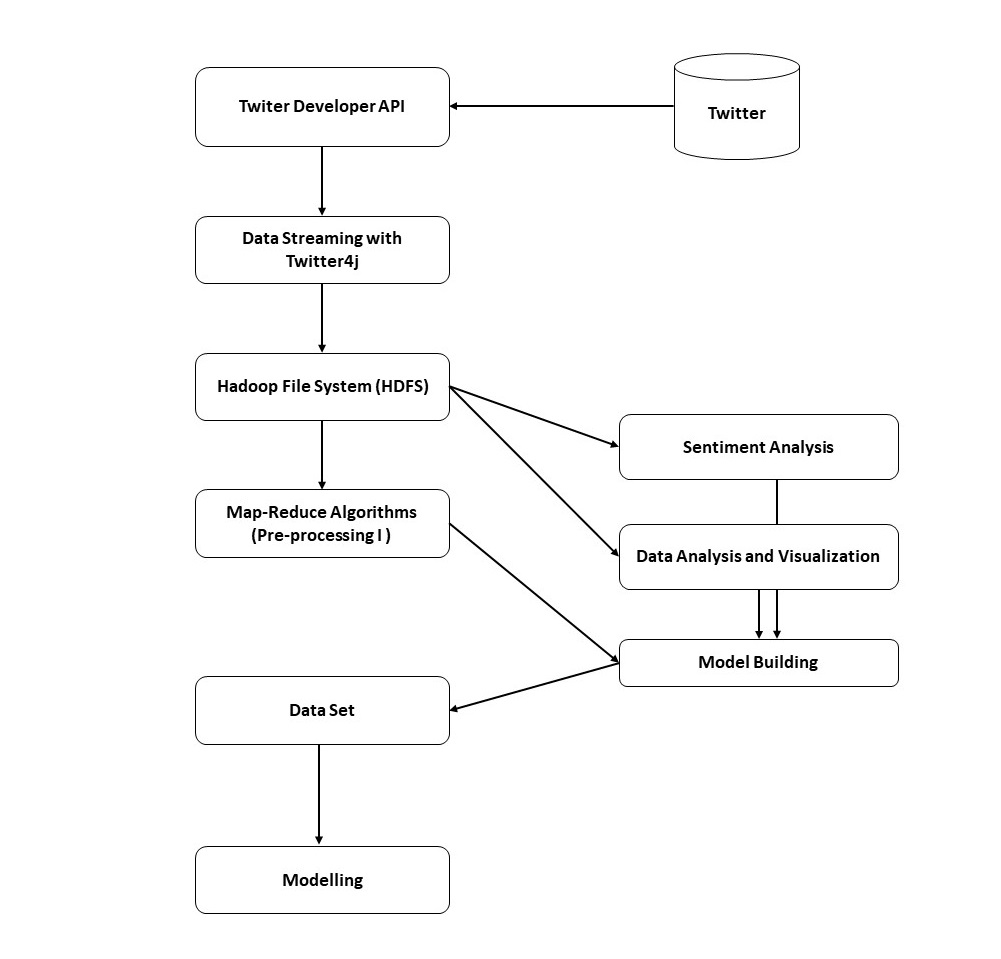
Twitter has been analyzed countless times involving numerous types of statistical interpretations. We have decided to look at the features of tweets with #Coronavirus in English and implement various queries on this dataset. In addition, we will extend past works analyzing political tweets and learn how features of tweets relate to its likelihood of being political([1](https://www.people-press.org/2019/10/23/national-politics-on-twitter-small-share-of-u-s-adults-produce-majority-of-tweets/),[2](https://knightfoundation.org/articles/polarization-in-the-twittersphere-what-86-million-tweets-reveal-about-the-political-makeup-of-american-twitter-users-and-how-they-engage-with-news/)).

The work presented in this document is an extension of the previous work (Increment-1). The previous work included analysis of data set, implementing Map-Reduce to get some insight of the data and sentiment analysis in Hive.

With the work in Phase 2, some extra analyses are performed in SparkSQL and visualized by cutting edge technologies available. With the in-depth analysis of data set, a classification model is going to be built by finding the correlation between some fields in the schema of data set.

**Model**

The flow of the model starts with creation of Twitter Developer account to query the Twitter data. The Twitter data is collected with the Twitter4j library by selecting tweets with the #Coronavirus and in the "English" language. This approach saves valuable time in cleaning the data. Then the data is stored in HDFS file system for further analysis. A set of Map-Reduce algorithm is implemented in order to get the preliminary insight of the data which will be used in the modelling part of the flow.



**Dataset**

The dataset used is not shared here as it would be a violation of the terms of service. The data set used in this project is collected on between 2.30-6.51 UTC on March 14. The data set contains 199,924 tweets. The data was collected with one tweet in json format per line. There are numerous keys in the tweet but we are only interest in the time the tweet was created(created\_at), the full text of the tweet(full\_text), the time the user created their Twitter account(user.created\_at), the amount of times the tweet has been retweeted(retweet\_count), the amount of followers the user has(user.follower\_count), the amount of friends the user has(user.friend\_count), the amount of groups the user is subscribed(user.listed\_count) and the tweet identifier(id).

**Analysis of data**

As we all know, Hadoop and the MapReduce frameworks have been around for a long time now in big data analytics. But these frameworks require a lot of read-write operations on a hard disk which makes it very expensive in terms of time and speed.

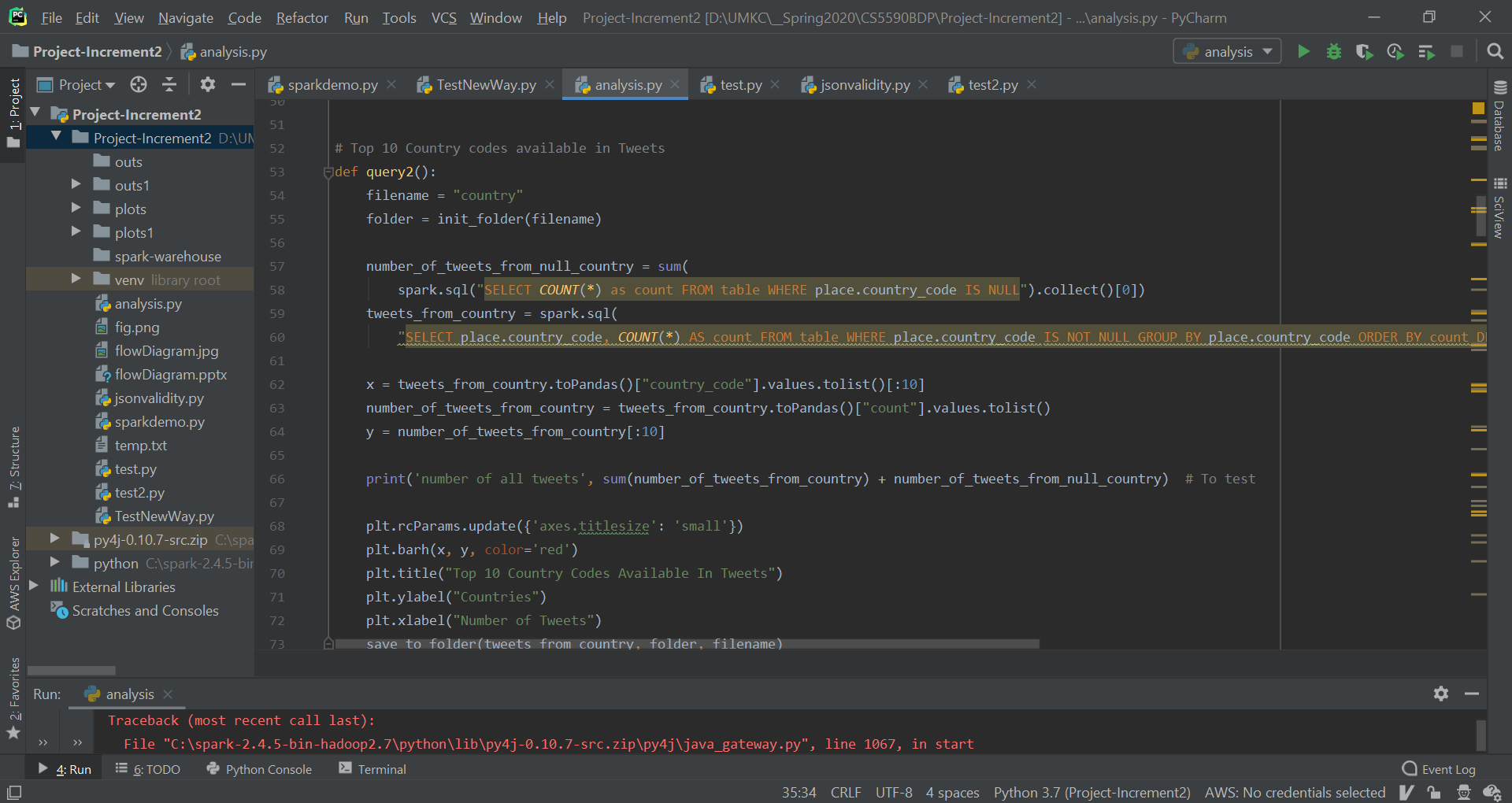
Apache Spark is the most effective data processing framework in enterprises today. It’s true that the cost of Spark is high as it requires a lot of RAM for in-memory computation but it’s still a hot favorite among Data Scientists and Big Data Engineers.

Spark SQL is an amazing blend of relational processing and Spark’s functional programming. It provides support for various data sources and makes it possible to make SQL queries, resulting in a very powerful tool for analyzing structured data at scale.

Here are some of the Spark SQL features:

* Query Structure Data within Spark Programs
* Compatible with Hive
* One Way to Access Data
* Performance and Scalability
* User-Defined Functions

We used the following queries in SparkSQL in order to better understand the data that will eventually help us in building the model. We used some other cutting edges tools such as pandas, matplotlib in addition to Apache Spark.



[Source code for analysis](https://github.com/JAWolfe04/CS5590-Group-Project/blob/master/Increment%202/Project%201a/SourceCode/analysis.py)

spark = SparkSession.builder.appName("Twitter PySpark Application").master("local[\*]").getOrCreate()

spark.sparkContext.setLogLevel("ERROR")

tweetsDF = spark.read.json("all.txt", multiLine=False)

tweetsDF.createOrReplaceTempView("table")

**1. Top 10 Countries where Twitter messages are tweeted.**

The countries which Tweets are coming from is important because, it shows the countries which are struggling with the COVID19 pandemic. It also provides information people's feelings about the pandemic. When we analyze the data USA being the first country; India, Australia and Great Britain are the next countries.

number\_of\_tweets\_from\_null\_country = sum(spark.sql("SELECT COUNT(\*) as count FROM table WHERE place.country\_code IS NULL").collect()[0])

tweets\_from\_country = spark.sql("SELECT place.country\_code, COUNT(\*) AS count FROM table WHERE place.country\_code IS NOT NULL GROUP BY place.country\_code ORDER BY count DESC")

x = tweets\_from\_country.toPandas()["country\_code"].values.tolist()[:10]

number\_of\_tweets\_from\_country = tweets\_from\_country.toPandas()["count"].values.tolist()

y = number\_of\_tweets\_from\_country[:10]

print('number of all tweets', sum(number\_of\_tweets\_from\_country) + number\_of\_tweets\_from\_null\_country) # To test

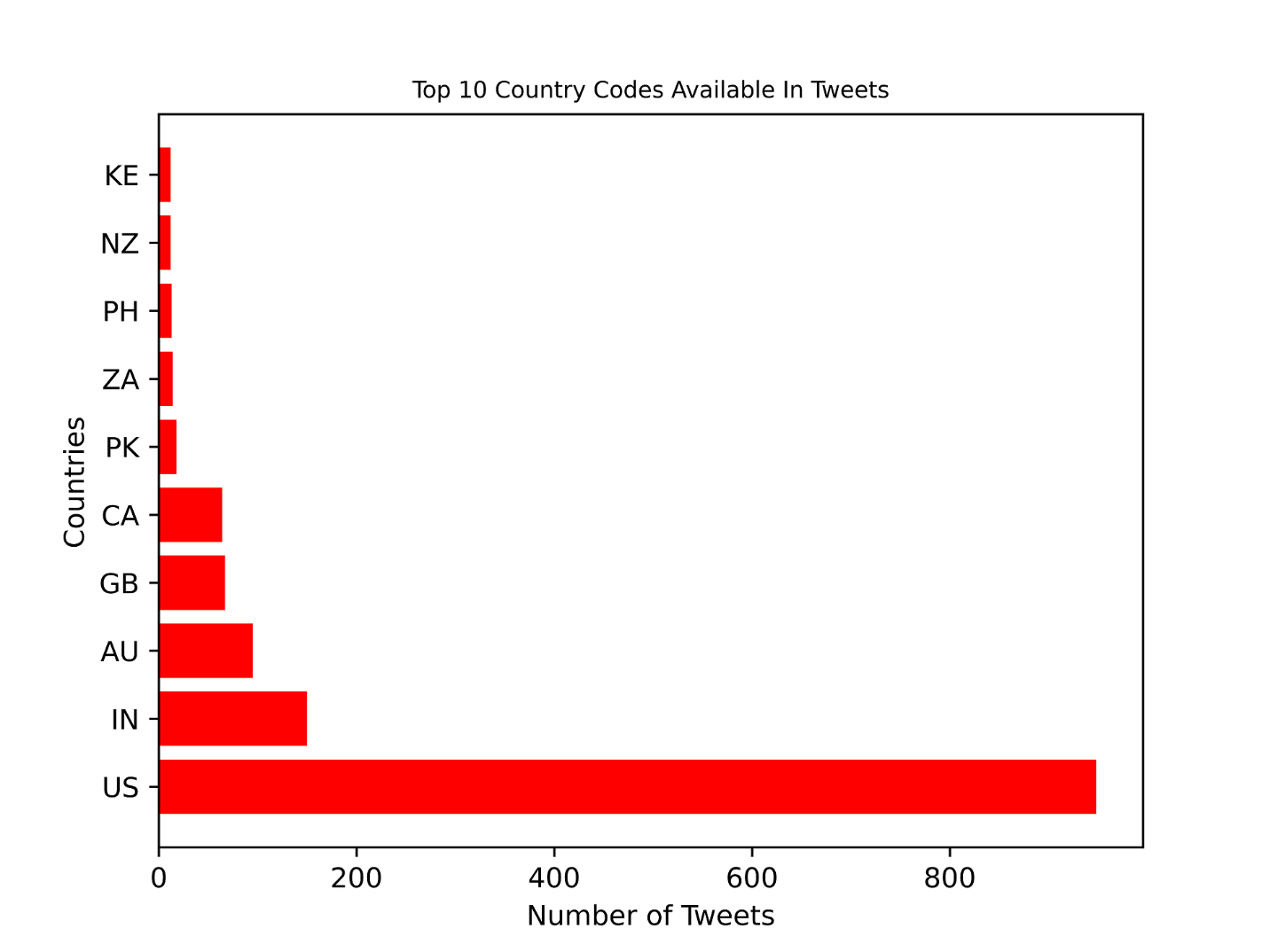
plt.rcParams.update({'axes.titlesize': 'small'})

plt.barh(x, y, color='red')

plt.title("Top 10 Country Codes Available In Tweets")

plt.ylabel("Countries")

plt.xlabel("Number of Tweets")



**2. Tweets Distribution in USA**

This shows where the tweets most often originate in the collected data. The primary source of tweets is California and the second most common is unspecified states. It is worth noting that geographical data only represents 1% of all tweets.

tweets\_from\_USA = spark.sql("SELECT user.location, COUNT(\*) AS count FROM table WHERE user.location LIKE '%USA%' GROUP BY user.location ORDER BY count DESC")

labels = tweets\_from\_USA.toPandas()["location"].values.tolist()[:10]

sizes = tweets\_from\_USA.toPandas()["count"].values.tolist()[:10]

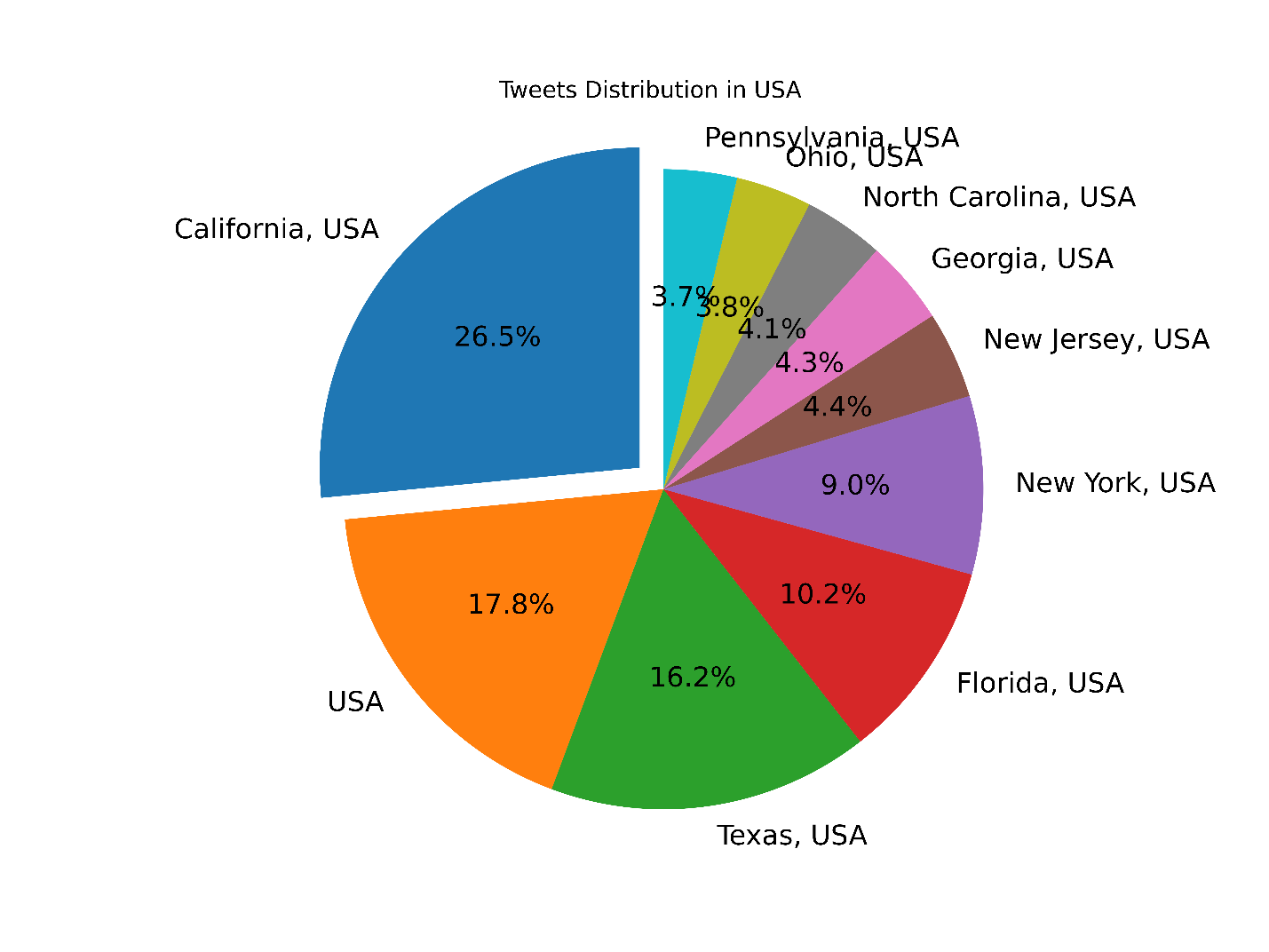
explode = (0.1, 0, 0, 0, 0, 0, 0, 0, 0, 0) # only "explode" the 1st slice

fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title("Tweets Distribution in USA")



**3. Top 10 Tweeters**

The top tweeters are accounts that have tweets that occur most often in the data set. This clearly shows a few account are tweeting much more often and tweet frequently enough for us to believe they are Twitter bots. Sometimes the user name can reveal more about these accounts and some are clearly marked as bots.

tweets\_dist\_person = spark.sql(Select user.id\_str, COUNT(user.id\_str) AS count from table WHERE user.id\_str is not null GROUP BY user.id\_str ORDER BY count DESC")

x = tweets\_dist\_person.toPandas()["id\_str"].values.tolist()[:10]

y = tweets\_dist\_person.toPandas()["count"].values.tolist()[:10]

figure = plt.figure()

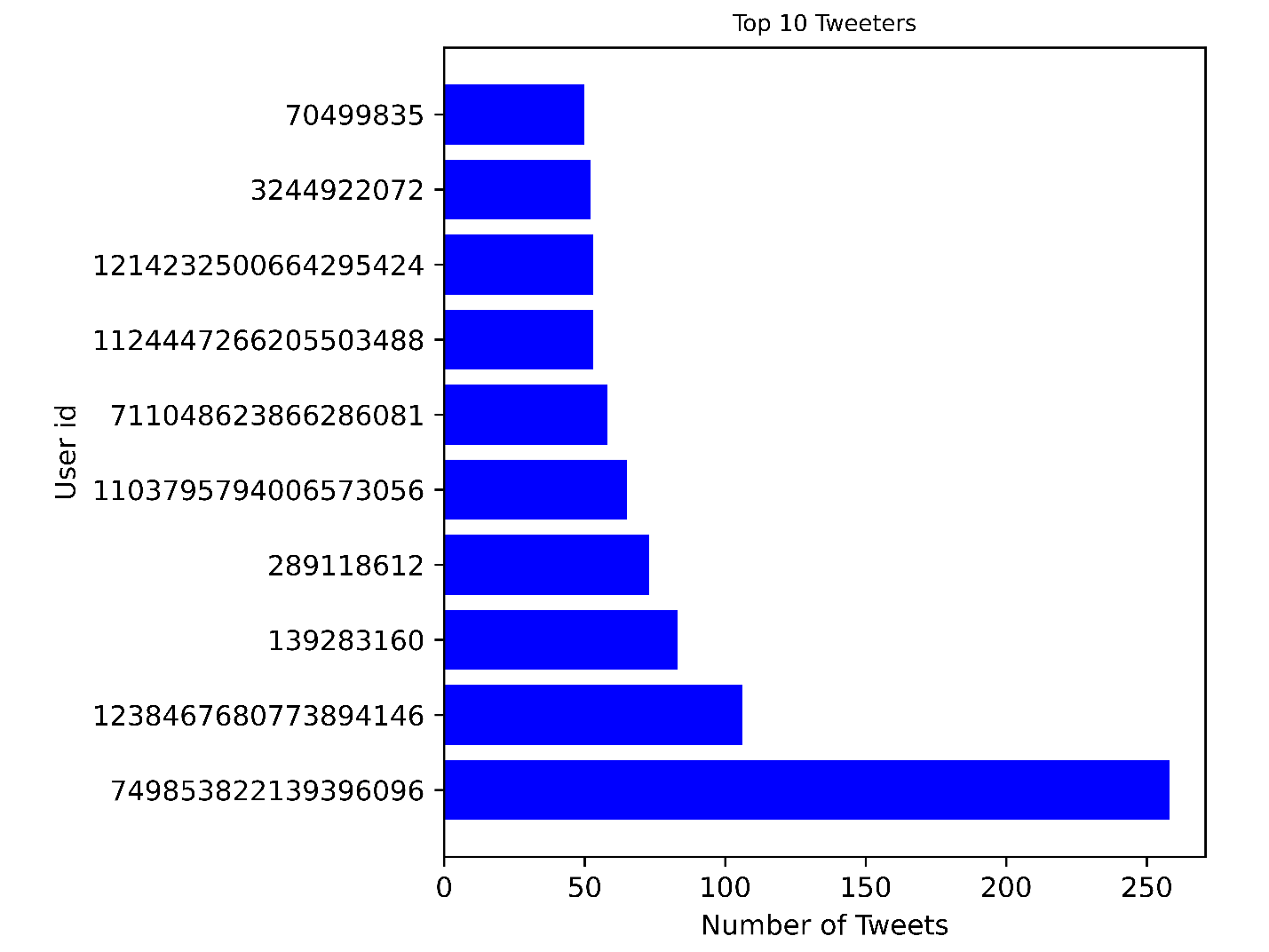
axes = figure.add\_axes([0.35, 0.1, 0.60, 0.85])

plt.barh(x, y, color='blue')

plt.title("Top 10 Tweeters")

plt.ylabel("User id")

plt.xlabel("Number of Tweets")



**4. Top 10 People Who Have Most Friends**

Friends on Twitter follow each other and generally have more involvement to become friends than a follower.

friendsCountDF = spark.sql("select user.screen\_name, user.friends\_count AS friendsCount from table where (user.id\_str, created\_at) in (select user.id\_str, max(created\_at) as created\_at from table group by user.id\_str ) ORDER BY friendsCount DESC")

x = friendsCountDF.toPandas()["screen\_name"].values.tolist()[:10]

y = friendsCountDF.toPandas()["friendsCount"].values.tolist()[:10]

figure = plt.figure()

axes = figure.add\_axes([0.3, 0.1, 0.65, 0.85])

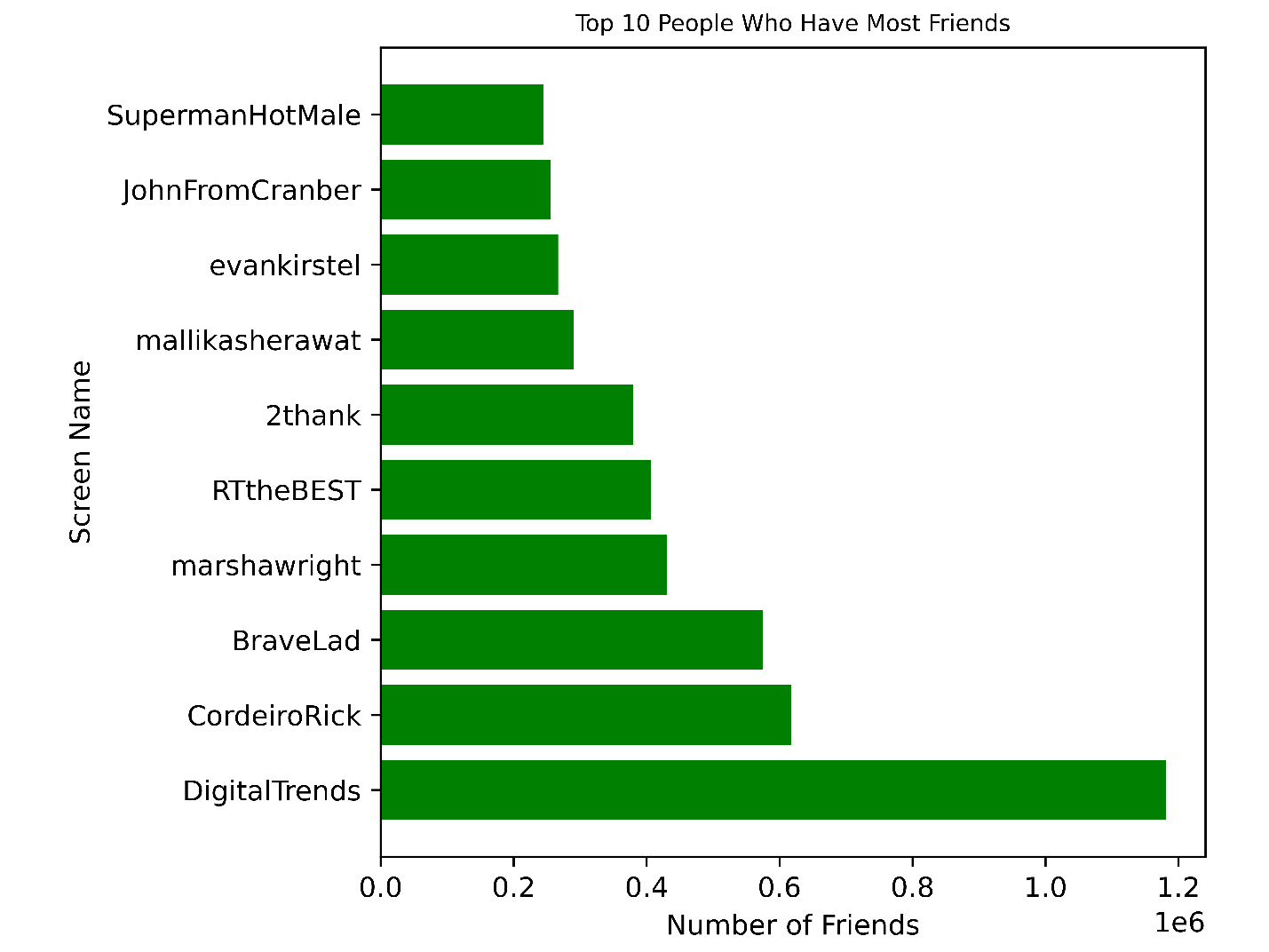
plt.rcParams.update({'axes.titlesize': 'small'})

plt.barh(x, y, color='green')

plt.title("Top 10 People Who Have Most Friends")

plt.ylabel("Screen Name")

plt.xlabel("Number of Friends")



**5. Hashtags Distribution**

Since we collected data around the #Coronavirus, we decided to separate hashtags by case as well to see if there is any interesting patterns.

hashtagsDF = spark.sql("SELECT hashtags, COUNT(\*) AS count FROM (SELECT explode(entities.hashtags.text) AS hashtags FROM table) WHERE hashtags IS NOT NULL GROUP BY hashtags ORDER BY count DESC")

labels = hashtagsDF.toPandas()["hashtags"].values.tolist()[:10]

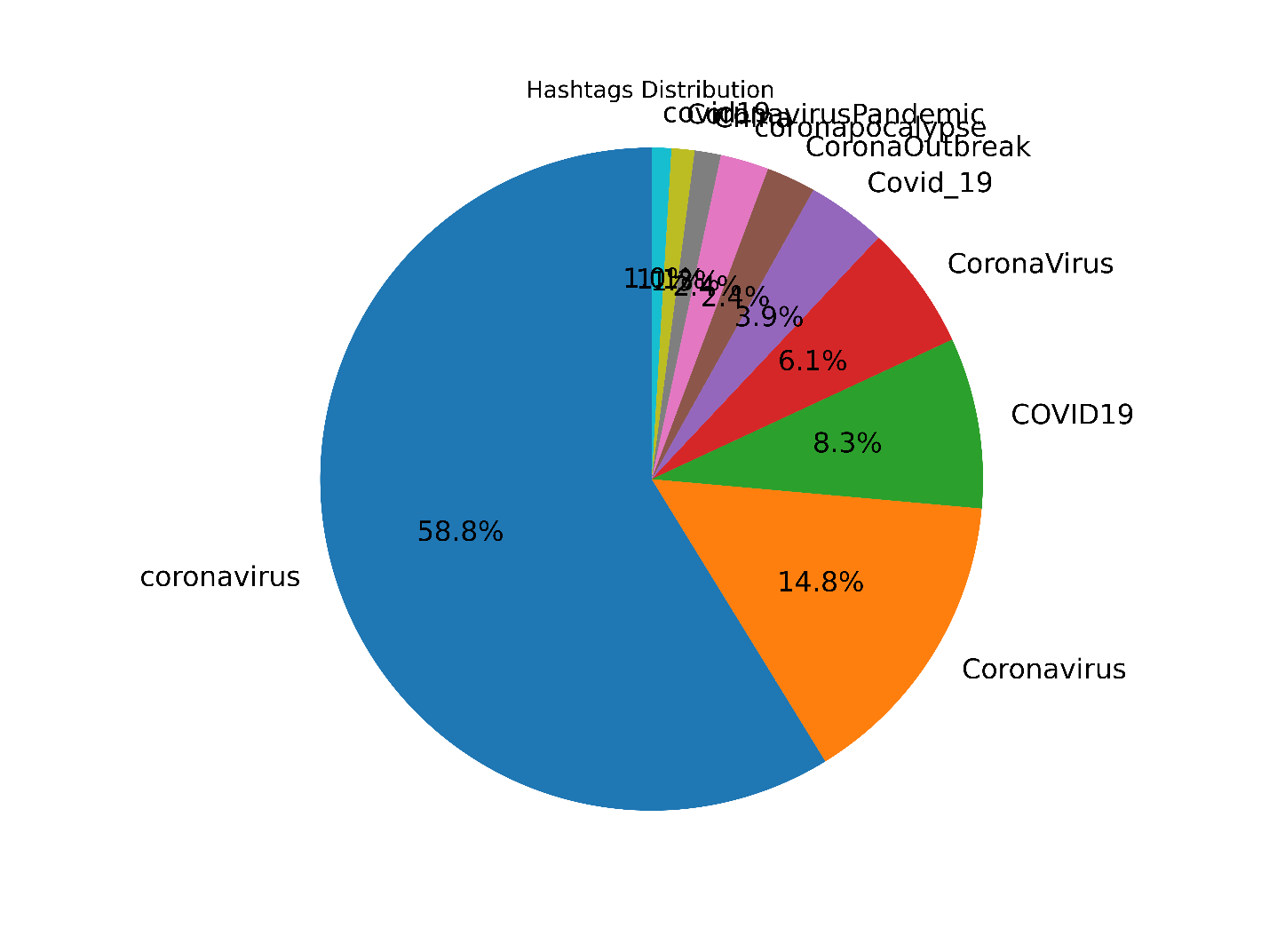
sizes = hashtagsDF.toPandas()["count"].values.tolist()[:10]

fig1, ax1 = plt.subplots()

ax1.pie(sizes, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title("Hashtags Distribution")



**6. Tweet Distribution according to time-Time series**

This revealed that people tweeted frequently, nearly 1000 tweets per minute about the coronavirus. It is also worth noting that at this point the Dow-Jones stock had dropped nearly 10,000 points, which might have caused more tweets.

tweet\_distributionDF1 = spark.sql("SELECT SUBSTRING(created\_at,12,5) as time\_in\_hour, COUNT(\*) AS count FROM table GROUP BY time\_in\_hour ORDER BY time\_in\_hour ")

from pyspark.sql import functions as F

tweet\_distributionDF = tweet\_distributionDF1.filter(F.col("count") > 2)

x = pandas.to\_numeric(tweet\_distributionDF.toPandas()["time\_in\_hour"].str[:2].tolist()) + pandas.to\_numeric(

tweet\_distributionDF.toPandas()["time\_in\_hour"].str[3:5].tolist()) / 60

y = tweet\_distributionDF.toPandas()["count"].values.tolist()

tick\_spacing = 1

fig, ax = plt.subplots(1, 1)

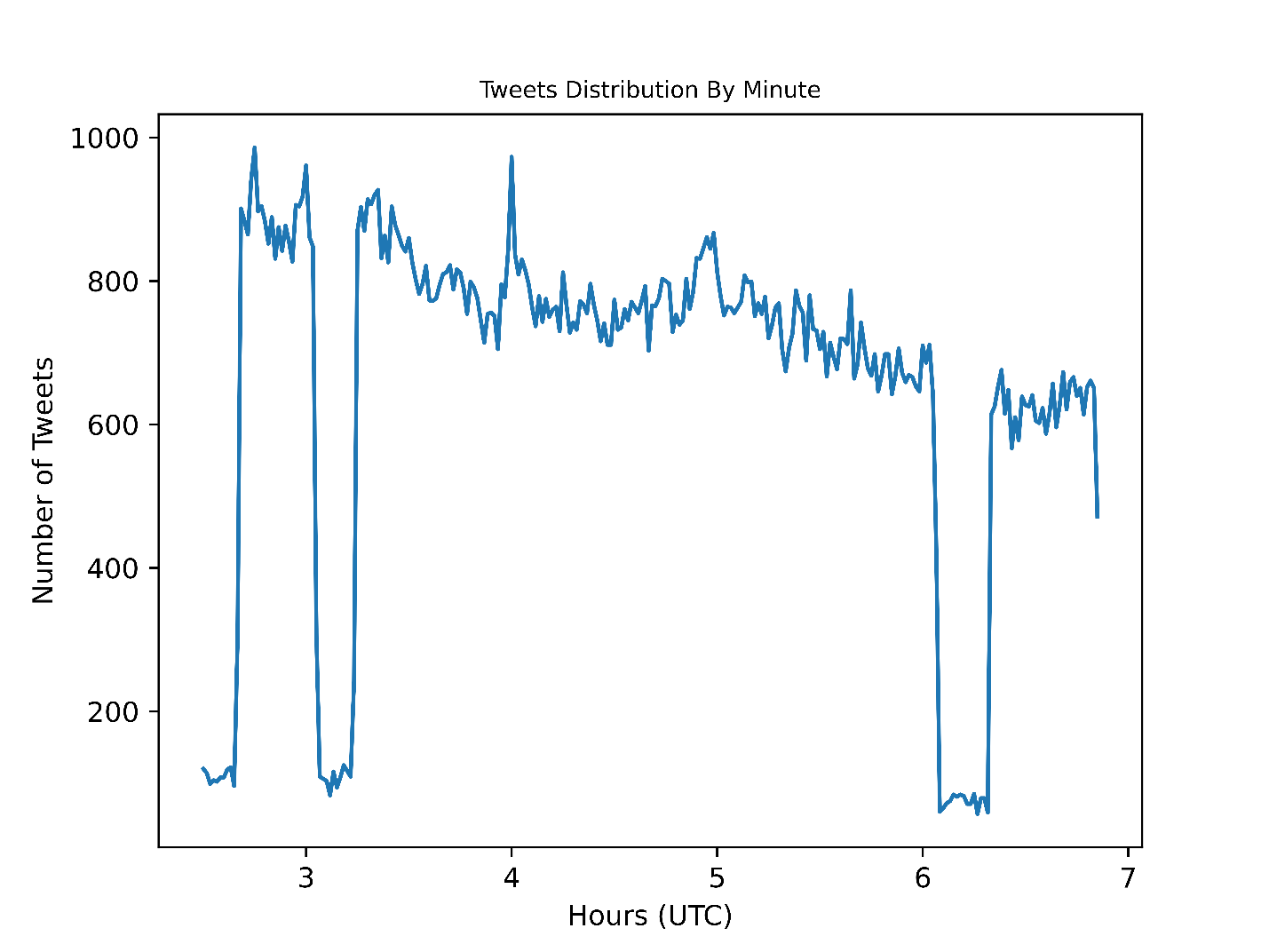
ax.plot(x, y)

ax.xaxis.set\_major\_locator(ticker.MultipleLocator(tick\_spacing))

plt.title("Tweets Distribution By Minute")

plt.xlabel("Hours (UTC)")

plt.ylabel("Number of Tweets")



**7. Top 10 Devices Used in the Tweets**

This shows that people clearly tweet from their phones and not much from other means.

df = spark.sql("SELECT source, COUNT(\*) AS total\_count FROM table WHERE source IS NOT NULL GROUP BY source ORDER BY total\_count DESC")

first = df.toPandas()["source"].str.index(">") + 1

last = df.toPandas()["source"].str.index("</a>")

text = df.toPandas()["source"].values.tolist()[:10]

x = []

for i in range(len(text)):

x.append(text[i][first[i]:last[i]])

y = df.toPandas()["total\_count"].values.tolist()[:10]

figure = plt.figure()

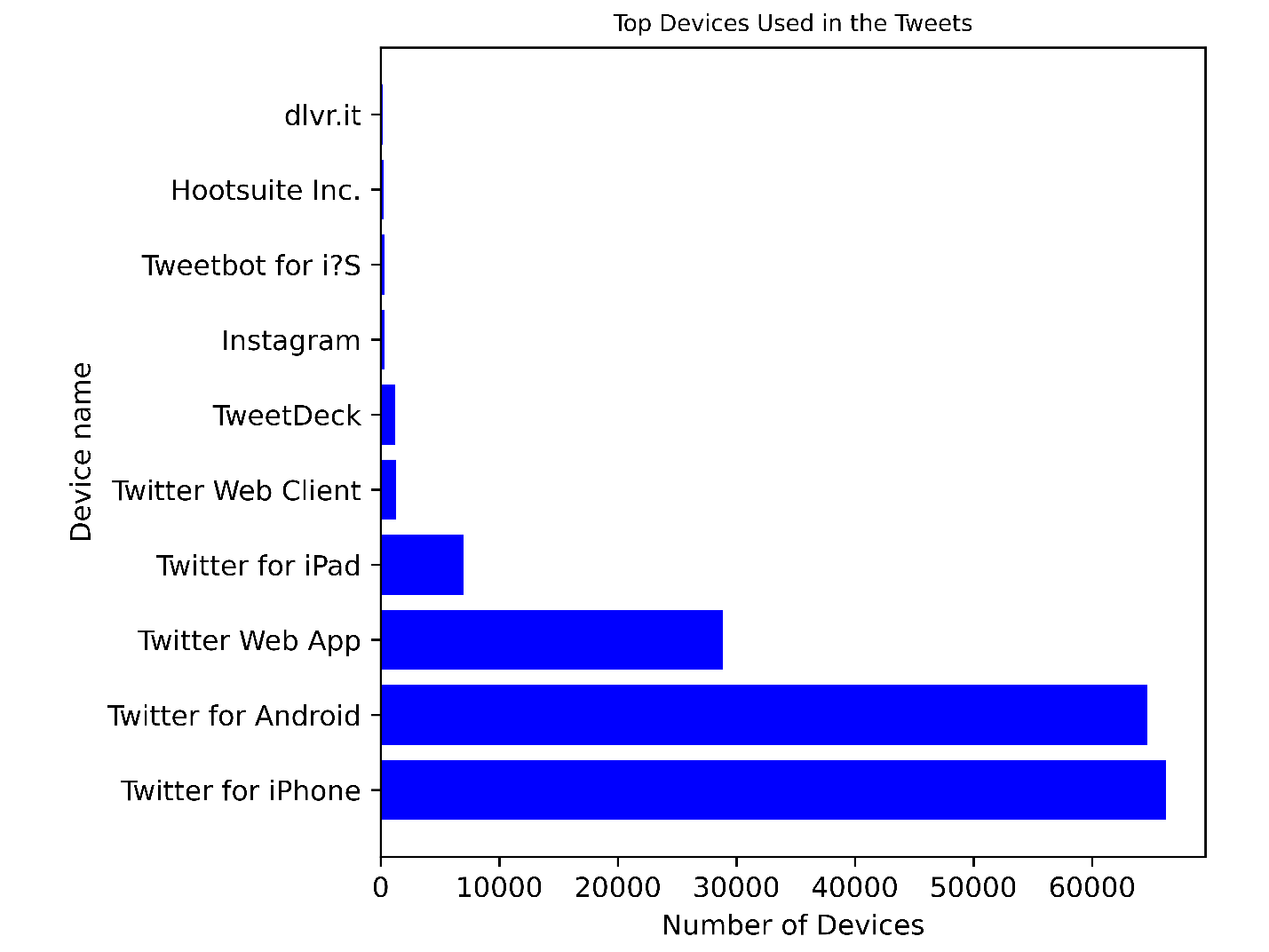
axes = figure.add\_axes([0.3, 0.1, 0.65, 0.85])

plt.barh(x, y, color='blue')

plt.ylabel("Device name")

plt.xlabel("Number of Devices")

plt.title("Top Devices Used in the Tweets")



**8. Tweets by Verified & Unverified Users**

Verified users are accounts that have applied for verification and Twitter has deemed worthy to verify. Generally this status is reserved for celebrities, politicians and organizations.

verified\_usersDF = spark.sql("SELECT user.verified, COUNT(\*) AS count FROM table GROUP BY user.verified ORDER BY user.verified ASC")

labels = verified\_usersDF.toPandas()["verified"].values.tolist()[:2]

sizes = verified\_usersDF.toPandas()["count"].values.tolist()[:2]

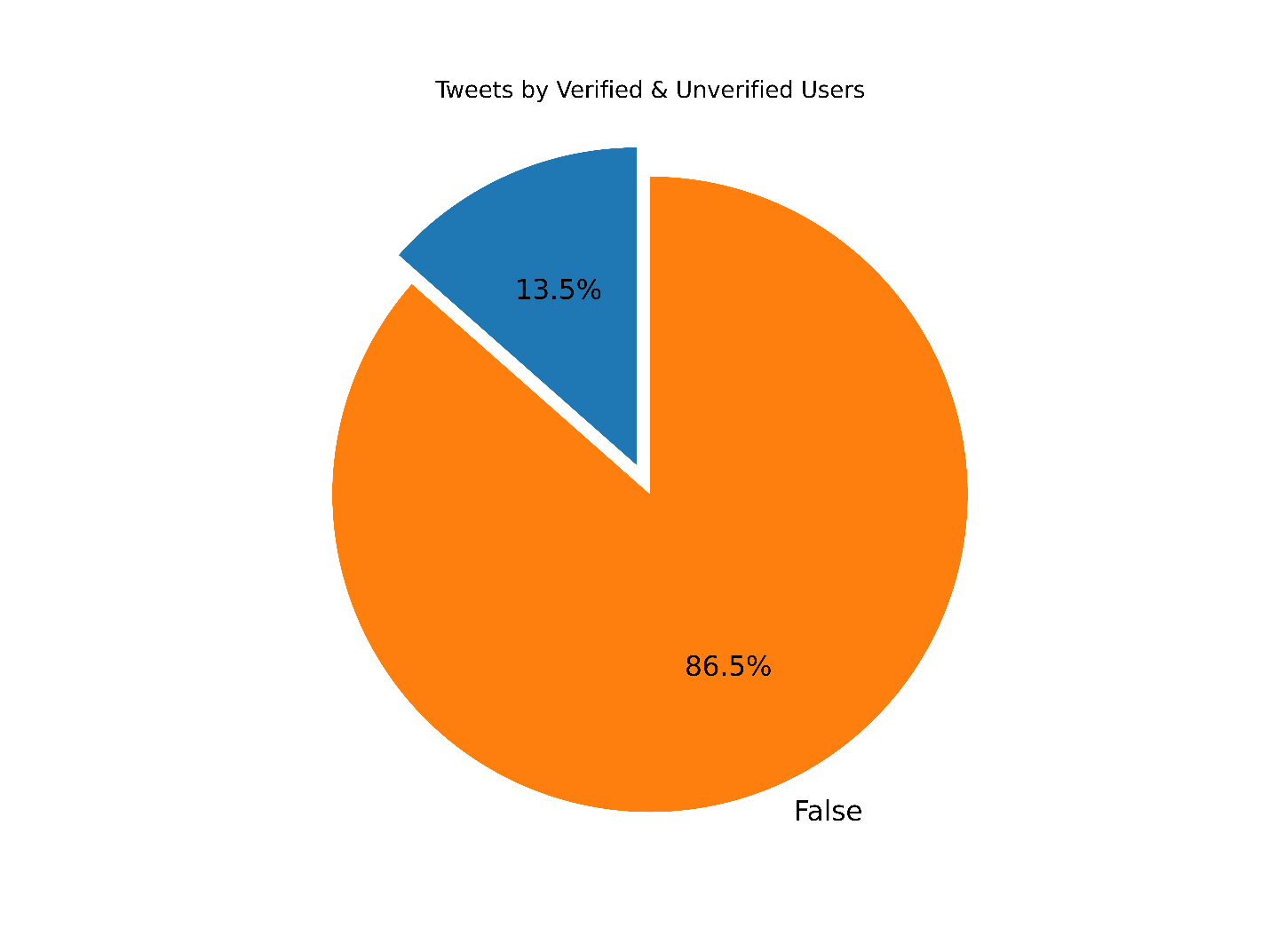
explode = (0, 0.1) # only "explode" the 2nd slice (i.e. 'Hogs')

fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title("Tweets by Verified & Unverified Users")



**Sentiment Analysis**

The AFINN lexicon is perhaps one of the simplest and most popular lexicons that can be used extensively for sentiment analysis. AFINN is basically a list of words rated with an integer value between minus five (negative) and plus five (positive) and zero (neutral). Before we use the python Afinn library for determining sentiment in the tweets, we cleared the special characters, emojis, RT and web sites links and used the cleared text to get the score in sentiment analysis.

import re

from afinn import Afinn

df = tweetsDF.select("full\_text").toPandas()

afinn = Afinn()

positive = 0;

neutral = 0

negative = 0;

for i in range(len(df)):

txt = df.loc[i]["full\_text"]

txt = re.sub(r'@[A-Z0-9a-z\_:]+', '', str(txt)) # replace username-tags

txt = re.sub(r'^[RT]+', '', str(txt)) # replace RT-tags

txt = re.sub('https?://[A-Za-z0-9./]+', '', str(txt)) # replace URLs

txt = re.sub("[^a-zA-Z]", " ", str(txt)) # replace hashtags

df.at[i, "full\_text"] = txt

sentiment\_score = afinn.score(txt)

if sentiment\_score > 0:

positive = positive + 1

elif sentiment\_score < 0 :

negative = negative + 1

else:

neutral = negative + 1

labels = ["Positive" , "Negative", "Neutral"]

sizes = [positive, negative, neutral]

explode = (0, 0.1, 0) # only "explode" the 2nd slice (i.e. 'Hogs')

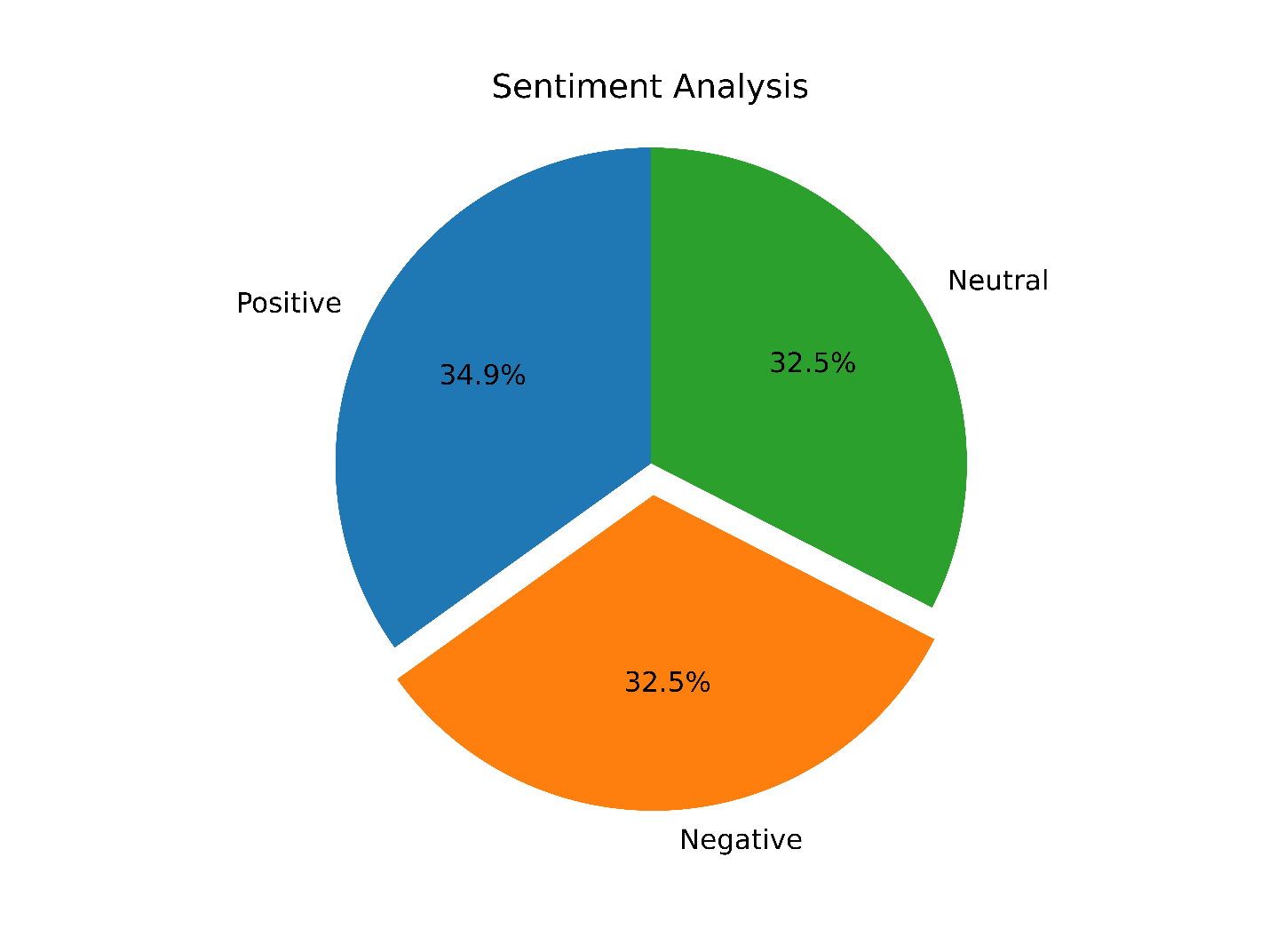
fig1, ax1 = plt.subplots()

ax1.pie(sizes, explode=explode, labels=labels, autopct='%1.1f%%', shadow=False, startangle=90)

ax1.axis('equal') # Equal aspect ratio ensures that pie is drawn as a circle.

plt.title("Sentiment Analysis")

plt.savefig(plots\_folder + filename + ".png", dpi=1200)



**Data Pre-processing**

[Map Reduce Code](https://github.com/JAWolfe04/CS5590-Group-Project/blob/master/TwitterWordCount/src/twitter/wordcount/App.java)  
[Spark Political Data Code](https://github.com/JAWolfe04/CS5590-Group-Project/blob/master/TwitterPolitical/political/src/main/java/twitter/political/App.java)

The first step in generating a data set to use for the development of a model was to develop a list of political terms to determine if a tweet is political. To develop this list, we use Map-Reduce to make a word count and order it by the count so that the most frequent words appear on top.  
The first part of this program was a traditional Map-Reduce applied to each tweet's full text converted to lower-case and with special characters removed:

private final static IntWritable one = new IntWritable(1);

private Text word = new Text();

@SuppressWarnings("deprecation")

@Override

public void map(Object key, Text value, Context context)

throws IOException, InterruptedException {

String tweet = value.toString();

try {

JsonObject jsonObject = new JsonParser().parse(tweet).getAsJsonObject();

String text = jsonObject.get("full\_text").getAsString();

text = text.toLowerCase().replaceAll("[\\,\\.\\|\\(\\)\\:\\'\\?\\-\\!\\;\\#\"\\$\\d]","");

if (text != null && text.length() > 0){

StringTokenizer tokenizer = new StringTokenizer(text);

while (tokenizer.hasMoreTokens()) {

word.set(tokenizer.nextToken());

context.write(word, one);

}

}

}

catch (JsonSyntaxException e) {

Logger.getRootLogger().log(Level.ERROR, tweet);

e.printStackTrace();

}

catch(IllegalStateException e) {

e.printStackTrace();

}

}

private IntWritable result = new IntWritable();

@Override

public void reduce(Text key, Iterable<IntWritable> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

for (IntWritable value : values)

sum += value.get();

result.set(sum);

context.write(key, result);

}

The second part of the Map Reduce sorted the word count by the count value:

@Override

public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

String[] wordCount = value.toString().split("\\s+");

context.write(new LongWritable(Long.parseLong(wordCount[1])), new Text(wordCount[0]));

}

@Override

protected void reduce(LongWritable key, Iterable<Text> trends, Context context) throws IOException, InterruptedException {

for (Text val : trends) { context.write(new Text(val.toString()), new Text(key.toString())); }

}

The data was processed to remove null values for select columns and dates were converted to timestamps to make the dates usable by Spark:

data = data.filter("retweet\_count is not null and user.followers\_count is not null and "

+ "user.friends\_count is not null and user.listed\_count is not null and id is not null"

+ " and created\_at is not null and user.created\_at is not null and user.statuses\_count is not null");

data = data.withColumn("created\_at", to\_timestamp(data.col("created\_at"), "EEE MMM dd HH:mm:ss '+0000' yyyy"));

data = data.withColumn("user\_created\_at", to\_timestamp(data.col("user.created\_at"), "EEE MMM dd HH:mm:ss '+0000' yyyy"));

Next a column identifying if the tweet is political with a 1 and not political with a 0 was generated from a list of key words after the text had been converted to lower-case and special characters were removed using a user-defined function:

sqlContext.udf().register("isPolitical", (UDF1<String, Integer>)(columnValue) -> {

String[] triggers = {"trump", "@realdonaldtrump", "president", "government", "administration",

"obama", "@teamtrump", "biden", "govt", "@realdonaldtrump’s", "voted", "donald", "media",

"@teampelosi", "journalists", "vote", "@speakerpelosi", "democrats", "senate", "federal",

"bipartisan", "republicans", "campaign", "maddow", "trumps", "legislation", "pres", "pelosi",

"democrat", "representatives", "governments", "maralago", "trump’s", "dems", "economy", "@vp",

"@reuters", "foreign", "parliamentary", "@whitehouse", "sen", "fauci", "fox", "@billdeblasio",

"@gavinnewsom", "conspiracy", "boomer", "department"};

if(columnValue != null && !columnValue.isEmpty()) {

String testTrigger = columnValue.toLowerCase().replaceAll("[\\,\\.\\|\\(\\)\\:\\'\\?\\-\\!\\;\\#\"\\$\\d]","");

for(String trigger : triggers) {

if(testTrigger.contains(trigger)) {

return 1;

}

}

}

return 0;

}, DataTypes.IntegerType);

data = data.withColumn("political", callUDF("isPolitical", col("full\_text")));

Then the timestamps for the time the tweet was created and the time the user account was created were compared to get the time the account active and divide the amount of status updates for the account by this time to get the average tweets per day, then the selected data was output to a json file:

data = data.withColumn("days\_since\_started", datediff(data.col("created\_at"), data.col("user\_created\_at")));

data = data.withColumn("tweets\_per\_day", data.col("user.statuses\_count") .divide(data.col("Days\_since\_started")));

data = data.select("id","political", "tweets\_per\_day", "retweet\_count", "user.followers\_count",

"user.friends\_count", "user.listed\_count");

data.coalesce(1).write().option("header", "true")

.json("C:\\Users\\Jonathan\\Desktop\\Shared Folder\\Political.json");

The resulting word count from the Map Reduce was used to select political words and generate a data set with select data:  
[Word Count Result](https://github.com/JAWolfe04/CS5590-Group-Project/blob/master/TwitterWordCount.txt)  
[Political Dataset](https://github.com/JAWolfe04/CS5590-Group-Project/blob/master/PoliticalData.json)  
The following list consists of the selected political terms to determine if the tweet is political:

"trump", "@realdonaldtrump", "president", "government", "administration",

"obama", "@teamtrump", "biden", "govt", "@realdonaldtrump’s", "voted", "donald", "media",

"@teampelosi", "journalists", "vote", "@speakerpelosi", "democrats", "senate", "federal",

"bipartisan", "republicans", "campaign", "maddow", "trumps", "legislation", "pres", "pelosi",

"democrat", "representatives", "governments", "maralago", "trump’s", "dems", "economy", "@vp",

"@reuters", "foreign", "parliamentary", "@whitehouse", "sen", "fauci", "fox", "@billdeblasio",

"@gavinnewsom", "conspiracy", "boomer", "department"

The resulting dataset creates rows like the following:  


**Graph model with explanation**

The graph and model will be created in the final paper.

**Project Management**

**Responsibility (Task, Person)**

**Data collection**

1. Analysing Streaming API and Twitter Developer Access - Jonathan (100 %)
2. Data Streaming Code - Jonathan (100 %)
3. Documentation - Mehmet (50 %) Jonathan (50 %)

**Data Cleaning**

1. Data Cleaning code - Jonathan ( 70 %) Mehmet (30 %)
2. Data Merging - Mehmet (100%)
3. Documentation - Mehmet (50%) Jonathan (50%)

**Sentiment Analysis**

1. Algorithm Analysis & Design - Mehmet (100 %)
2. Coding - Mehmet (100 %)
3. Visualization - Mehmet (100 %)
4. Documentation - Mehmet (100 %)

**Data Analysis and Visualization**

1. Algorithm Analysis & Design - Mehmet (95 %), Jonathan (5 %)
2. Coding - Mehmet (100 %)
3. Visualization - Mehmet (100 %)

**MapReduce Framework**

1. Design for Mapper, Reducer, Main - Jonathan (80 %), Mehmet (20%)
2. Coding for Mapper - Jonathan (100 %)
3. Coding for Reducer - Jonathan (100 %)
4. Documentation - Jonathan (100 %)

**Work to be completed**

* Description
  + Spark ML implementation on the political data to develop model(10-15% work left of the project)
* Responsibility (Task, Person)
  + Spark ML implementation on the political data to develop model (Mehmet, Jonathan)
* Issues/Concerns
  + System size is restricted due to our laptops
  + Features do not map well in model generation

**References**

1. <https://www.analyticsvidhya.com/blog/2020/02/hands-on-tutorial-spark-sql-analyze-data/>
2. <https://www.people-press.org/2019/10/23/national-politics-on-twitter-small-share-of-u-s-adults-produce-majority-of-tweets/>
3. <https://knightfoundation.org/articles/polarization-in-the-twittersphere-what-86-million-tweets-reveal-about-the-political-makeup-of-american-twitter-users-and-how-they-engage-with-news/>