CSEE5590/490: Big Data Programming

Final Project Report

Due Date: May 7(Fri) 2021, 11:59PM

PROJECT TITLE : COVID 19 Sentiment Analysis

GROUP 5

VYOMA DESAI - 16314631

AFFAN CHAROLIA - 16305677

ALI ALYAMI- 16293096

INTRODUCTION

COVID-19 has affected every single person around the globe and has had a huge impact on businesses everywhere. People are sounding their emotions on various social networking platforms and one of those platforms is twitter. The hashtag "Covid19" is trending and people are voicing their sentiments on it. Twitter makes public Tweets and replies available to developers. These endpoints can easily be used by people to identify, understand, and counter misinformation around public health initiatives.

BACKGROUND

Covid-19 has affected many people in different ways, and to ease up on its effect on them, great minds across the globe have tried to engineer solutions to this problem through vaccines. Although resolutions being available, there are doubts about its effectiveness. To gain insights on this issue, analysis is being made to better understand where the issues lie and to spread knowledge. Twitter has been a common platform where people have sounded their opinions and shared their

experience on taking vaccines. Also, dataset has been released by committees giving out vaccine and storing records on its effects on patients.

A. Cotfas et al in their paper entitled ‘The Longest Month: Analyzing Covid-19 Vaccination Opinions Dynamics From Tweets in the Month Following the First Vaccine Announcement’. [1] have used tweets of the first month after release of vaccine to the public and analyzed them through various classification algorithm. They have made comparison between classical machine learning and deep learning algorithm to find which of it would be better suited for this problem and have chosen the best classifier from them based on four performance metrics. 2,349,659 tweets have been used by them for their analysis. The tweets have been classified into favor, against and neutral with an accuracy of 78.94%. The tweets are analyzed with the local news around that location to monitor the evolution of perspective towards Covid-19 Vaccine. In their analysis they found the opinion of people being more towards the neutral side both on daily basis as well as on an overall basis.

The paper ‘Quantifying Covid-19 Content in the Online Health Opinion War Using Machine Learning’ [2] written by R. F. Sear et al talks about the opinions being shared about the vaccine online on various social media platform dividing people into two groups namely pro-vax and anti-vax. The authors have used dataset collected from various social media platforms and used it to divide people into two groups based on their opinions of this subject. They have used machine learning algorithm to cluster people into two groups. They have concluded that the anti-vax group is more vocal on these social media platforms and is made up of a much diverse group as compared to pro-vax whose number are less and are less engaging leading the authors to believe that this may lead the new people entering the discussion getting inclined towards anti-vax group leading to fake news forwarding becoming prevalent. Mechanistic Model has been used to further support the efficiency of the systems used by them.

N. Paul and S. S. Gokhale have talked in their paper entitled ‘Analysis and Classification of Vaccine Dialogue in the Coronavirus Era’ [3] about opinions of people on getting vaccinated. The authors like other have grouped people into ant-vax and pro-vax community. They have used the twitter dataset on Covid-19 vaccine for the analysis. They have used various data mining tool to get hold of this data and have cleaned and analyzed on it. They have done opinion mining to classify people into two communities of pro-vax and anti-vax. They have also used mathematical models to get some numbers to better understand their data. They made a model with an accuracy of 80% and concluded that the anti-vax opinions are more as compared to other group and are leading the authors to believe that this

will lead political parties and pharmaceutical companies to cut corners which will lead to adverse effects.

MOTIVATION

COVID-19 has affected every single person around the globe and has had a huge impact on businesses everywhere. People are sounding their emotions on various social networking platforms and one of those platforms is twitter. The hashtag “Covid19” is trending, and people are voicing their sentiments on it. Twitter makes public Tweets and replies available to developers. Hence, it allows developers to post Tweets via API. Developers can access Tweets by searching for specific keywords or requesting a sample of Tweets from specific accounts. These endpoints can easily be used by people to identify, understand, and counter misinformation around public health initiatives.

SIGNIFICANCE

The pandemic of Covid-19 has taken everyone by surprise. Many people have lost their jobs and their businesses. It's a topic which has grappled everyone and is a point of discussion and debate. There have been cases where people have had breakdowns because of this\environment that has been created because of it. So, it is significant to know how people are feeling about this and what are their sentiments towards it.

OBJECTIVES

1. To understand the sentiments of people.

2. To find out which people are affected the most.

3. Which locations are affected the most.

FEATURES

We collected tweets on the topic “Covid-19” and will be pre-processing the data to remove unnecessary data. Further we will be using Hadoop and Spark to analyze the data. We will be using various Hadoop tools like MapReduce, sqoop and hive and some of the spark tools to help assist us in this project.

DATASET

We are using COVID-19 tweets dataset from all over the world that we got from Kaggle website. Our aim is to perform sentiment analysis on the dataset so that we can conclude about the impact of COVID-19 around the world. Whatever they express on social media like twitter.

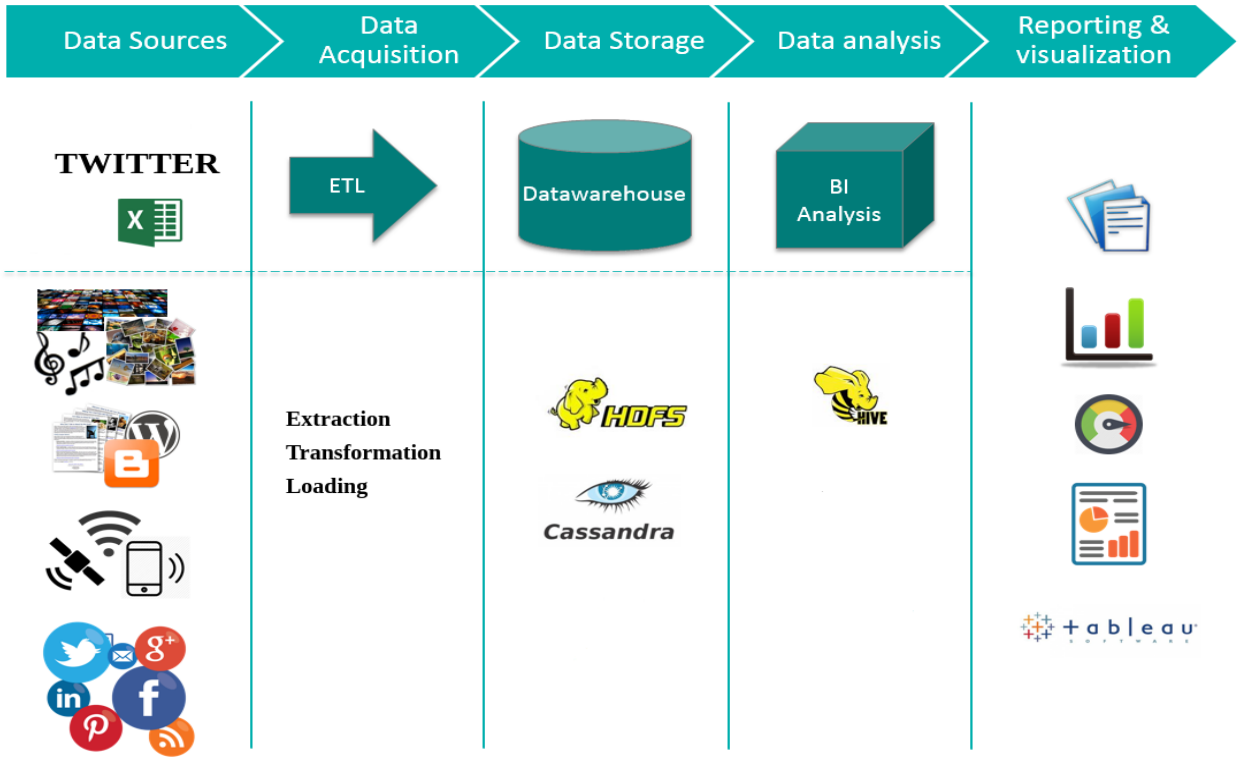
Dataset URL: https://www.kaggle.com/gpreda/covid19-tweets

Dataset Features:

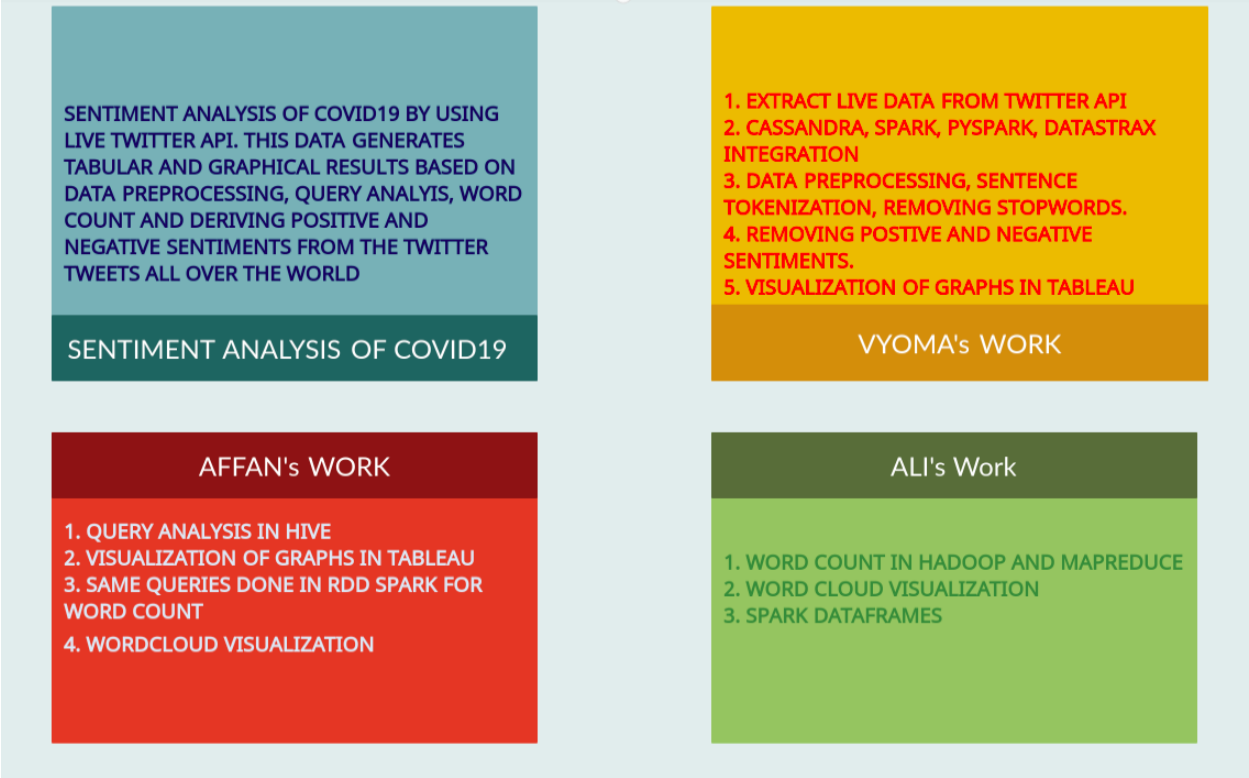
Provided dataset contains the following features or tables. User_name, user_location, user_description, user_created, user_followers, user_friends, user_favourites, user_verified, date, text, hashtag, source, is_retweet.

If we consider these fields then we can see that we need very few of them, such as we will be using the "text" field to get the text of tweet, whatever someone has written about the COVID-19. Other tables such as user location can be targeted to find out about the people of a certain area or geo position. We can also use hastags to find out how much the people are expressing the terms.

PROJECT WORKFLOW



CONTRIBUTION OF WORK



Data Analysis: Fields Description:

Field Title	Description
User_name	Describes the user name of person on twitter
User_location	Describes the location of that person
User_description	Describes the description of that user from twitter account
User_created	Describes when the account of user was created

User_followers	Describes how many followers that user have
User_friends	Describes how many friends the user have
User_favourites	Describes the favorites of the user
User_verified	Describes if the user is verified or not
Date	Date when the tweet was made
Text	Describes the text that was written in the tweet
Hashtag	Describes the hashtags that were used in the tweet
Source	Describes the source from where tweet was made such as Phone/Desktop
Is_retweet	Describes if it's a retweet or not

the fields we are looking for are text, date, and user_location.

IMPLEMENTATION

INSTALLATION OF HADOOP, FLUME, HIVE, and SPARK Services:

Installing and configuring the Hadoop, Flume, Hive and Spark services.

1. Both Apache flume and Hadoop File System(HDFS) work seamlessly on Linux Operating System so it is recommended for the user to have Flume

Agent and HDFS installed on Linux Operating System. However with the users who are using Mac or Windows they need to install any virtual machine application such as VMware Workstation Pro or Oracle Virtual Box. For our project, we have installed VMware workstation pro. One can download the VMware workstation pro setup file from the following link https://www.vmware.com/products/workstation-pro.html . Once the file is downloaded one can just setup the software following the steps displayed on the screen .

2. After VMWorkstation Pro software is installed we need to set up a new virtual Machine. Again this is just a simple straight step where the user needs to click on the 'Create a New Virtual Machine' button displayed on the home page and the user is asked to provide all the specifications for the virtual machine they wanted to set up. Once all the steps are completed we have successfully set up a new virtual machine.
3. After setting up a new virtual machine, it is now time for us to install and configure Apache Flume on our new Virtual Machine. Apache Flume is an open source application so we can download the software free of cost visiting the following link https://flume.apache.org/download.html . There are multiple versions of Apache Flume available but for the project we have used flume version 1.6.0. There are 2 types of Apache Flume setup file available one is with the suffix of bin.tar.gz and the other is src.tar.ga. Among the two we need to choose the file with a suffix bin.tar.gz. After downloading the file it gets a file in a zip format. So we just need to unzip and extract the files out of that zipped folder.
4. Extracting the file will install Flume in the Virtual Machine. So in order to check that Flume has installed correctly in our Virtual Machine we need to open a new terminal and type the following commands

getit .bashrc

source . bashrc

flume-ng

After entering the above commands if the Flume has been installed correctly in our system then it will display all the help commands for Flume agent. After confirming that we have installed Flume correctly then it is now time to configure Flume to extract Twitter data. For that we need to create a config file under the conf folder of the directory where we have installed the Flume application. In our case our conf file is located at the following location.

/usr/local/hadoop-env/flume-1.6.0/conf

In order to create a conf file we can open any text editor and specify the following properties such as Flume agent name, source where the data is being extracted, channel which is a temporary data Staging storage and finally sink information which is our destination where the data is finally stored.

Here is the code snippet of our config file. If you need to explore more on the config file please check for the file twitter-flume-hdfs.conf on our project package.

twtagent.sources = Twitter

twtagent.channels = memchannel

twtagent.sinks = HDFS

Source Information

t.sources.Twitter.type = org.apache.flume.source.twitter.TwitterSource

twtagent.sources.Twitter.type = com.cloudera.flume.source.TwitterSource

Sink Information

twtagent.sinks.HDFS.channel = memchannel

twtagent.sinks.HDFS.type = hdfs

twtagent.sinks.HDFS.hdfs.path = hdfs://localhost:9000/twitter_Extract/

Channel Information

twtagent.channels.memchannel.capacity = 10000

twtagent.channels.memchannel.type = memory

twtagent.channels.memchannel.transactionCapacity = 100

5. After configuring Flume now we need to install the HDFS on our Virtual Machine. Again HDFS is an open source application so one can download it free of cost visiting the following URL https://archive.apache.org/dist/hadoop/core/ . There are multiple versions of HDFS available but for our project we have installed hadoop 2.7.0. In the download page we get two types of files one is with the suffix .src.tar.gz and

the other is .tar.gz. Among the two we need to choose .tar.gz. Once the file is downloaded we need to untar the file from the terminal. We need to execute the following command in the terminal to untar our application.

Tar -xvf hadoop-2.7.0.tar.gz -C /usr/local/hadoop-env

After untarring the file we need to set up environment variables and we can do that editing .bashrc file. In order to edit .bashrc file we can open a terminal and type the following command

gedit .bashrc

It will open the .bashrc file in edit mode and we need to set up JAVA path and Hadoop path as follows

#HADOOP path

export HADOOP_HOME=/usr/local/hadoop-env/hadoop-2.7.0

export HADOOP_PREFIX=/usr/local/hadoop-env/hadoop-2.7.0

export HADOOP_MAPRED_HOME=${HADOOP_HOME}

export HADOOP_COMMON_HOME=${HADOOP_HOME}

export HADOOP_HDFS_HOME=${HADOOP_HOME}

export YARN_HOME=${HADOOP_HOME}

export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop

#Java path

Add Hadoop bin/ directory to PATH

export PATH=$PATH:$HADOOP_HOME/bin:$HADOOP_HOME/sbin

After finishing this we need additional configuration on core-site.xml , hdfs-site.xml, yarn-site.xml and mapred-site.xml. Since we have included all the files in our project package so one can just copy the content of those files and create individual files on their instances. Just as FYI all these files needs to be created at the following location

/usr/local/<YOUR FOLDER NAME WHERE HADOOP IS INSTALLED>/etc/hadoop

6. Once we have configured everything it is now time to check if HDFS is installed correctly on my machine or not. To check that we need to open a new terminal and enter the following command

hadoop version

If the hadoop is installed correctly then it will display us the hadoop version and other additional information of the hadoop application. After the installation of Hadoop we need to create a directory in the hadoop system where our Twitter data will be stored. In order to create a new directory we need to enter the following command in a terminal window

hadoop fs -mkdir /<YOUR CHOICE OF FOLDER NAME

7. After setting up both the FLUME and Hadoop system it is now time to run the services to extract the data from Twitter. In order to start hadoop services we need to enter the following command in the terminal window

Start-all.sh

To make sure if all the Hadoop services are started correctly or not we can visit the following URL in the browser

http://localhost:50070

If the browser redirects to the welcome page of Hadoop system then it is confirmed that all the Hadoop services have started successfully. After starting the Hadoop services we need to start the Flume agent and extract the data from Twitter. To do so we need to enter the following command in the terminal window. When the user is executing the command he/she needs to be on the home directory where the Flume agent is installed

./bin/flume-ng agent -n <YOUR TWITTER AGENT NAME> -c conf -f conf/twitter-flume-<YOUR FLUME CONFIG FILE NAME> -Dflume.root.logger=INFO,console

8. Once the process gets started we can visit the URL http://localhost:50070 and under Utilities menu bar ,we choose Browse the file system sub menu. There we can see the folder that we created earlier and when we open that

folder we can see some data files that are being extracted. We can just download that file and open it in any text editor to view its content.

Hive Installation:

1. Verifying JAVA Installation ($ java -version)
2. Verifying Hadoop Installation ($ hadoop version)
3. Download Hive from http://apache.petsads.us/hive/hive-0.14.0/.
4. Make sure Hive is installed and HDFS, Hadoop is configured.
5. Extracting and verifying Hive Archive
6. $ tar zxvf apache-hive-0.14.0-bin.tar.gz
7. Copy the extracted files to local directory
8. Set up the Hive environment by copying the following path in ~/.bashrc file.
9. export HIVE_HOME=/usr/local/hive
10. export PATH=$PATH:$HIVE_HOME/bin
11. export CLASSPATH=$CLASSPATH:/usr/local/Hadoop/lib/*:.
12. export CLASSPATH=$CLASSPATH:/usr/local/hive/lib/*:.
13. Execute $ source ~/.bashrc to run the .bashrc file.
14. Configuring Hive
15. To configure Hive with Hadoop, we edit the “hive-env.sh file”, located in the $HIVE_HOME/conf directory.
16. The following commands redirect to Hive config folder and copy the template file:

$ cd $HIVE_HOME/conf

$ cp hive-env.sh.template hive-env.sh

17. Edit the hive-env.sh file by appending the following line:

export HADOOP_HOME=/usr/local/hadoop

18. Hive installation is completed

But we also need an external DB server to configure the Metastore, hence we also install Apache Derby.

Downloading Apache Derby:

19. Download the file using following command:

$wget http://archive.apache.org/dist/db/derby/db-derby-10.4.2.0/db-derby-10.4.2.0-bin.tar.gz

20. Extracting and verifying Derby archive

$ tar zxvf db-derby-10.4.2.0-bin.tar.gz

21. Copy the files to /usr/local/derby directory using the cp command.
22. Setting up the environment for Derby : edit the ~/.bashrc file and enter the following lines.

export DERBY_HOME=/usr/local/derby

export PATH=$PATH:$DERBY_HOME/bin

export CLASSPATH=$CLASSPATH:$DERBY_HOME/lib/derby.jar:$DERBY_HOME/lib/derbytools.jar.

23. Create a directory to store Metastore

$ mkdir $DERBY_HOME/data

24. Configuring Metastore of Hive:

Configuring Metastore means specifying to Hive where the database is stored. You can do this by editing the hive-site.xml file, which is in the $HIVE_HOME/conf directory. First of all, copy the template file using the following command:

$ cd $HIVE_HOME/conf

$ cp hive-default.xml.template hive-site.xml

25. Edit hive-site.xml and append the following lines :

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:derby://localhost:1527/metastore_db;create=true </value>
  <description>JDBC connect string for a JDBC metastore </description>
```

</property>

26. Create a file named jpox.properties and add the following lines into it:

javax.jdo.PersistenceManagerFactoryClass = org.jpox.PersistenceManagerFactoryImpl

org.jpox.autoCreateSchema = false

org.jpox.validateTables = false

org.jpox.validateColumns = false

org.jpox.validateConstraints = false

org.jpox.storeManagerType = rdbms

org.jpox.autoCreateSchema = true

org.jpox.autoStartMechanismMode = checked

org.jpox.transactionIsolation = read_committed

javax.jdo.option.DetachAllOnCommit = true

javax.jdo.option.NontransactionalRead = true

javax.jdo.option.ConnectionDriverName = org.apache.derby.jdbc.ClientDriver

javax.jdo.option.ConnectionURL = jdbc:derby://hadoop1:1527/metastore_db;create = true

javax.jdo.option.ConnectionUserName = APP

javax.jdo.option.ConnectionPassword = mine

27. Verifying Hive Installation: Before running Hive, you need to create the /tmp folder and a separate Hive folder in HDFS. Here, we use the /user/hive/warehouse folder. You need to set write permission for these newly created folders as shown below: chmod g+w
28. Now set them in HDFS before verifying Hive. Use the following commands:

$ $HADOOP_HOME/bin/hadoop fs -mkdir /tmp

$ $HADOOP_HOME/bin/hadoop fs -mkdir /user/hive/warehouse

$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /tmp

$ $HADOOP_HOME/bin/hadoop fs -chmod g+w /user/hive/warehouse

29. The following commands are used to verify Hive installation:

$ cd $HIVE_HOME

$ bin/hive

30. On successful Installation and running the above command, you should see the Hive Terminal.
31. On the Other Hand, we can also download HDP Sandbox 3.0, which is a virtual machine for using HDFS, Hive and other services. More at: https://www.cloudera.com/downloads/hortonworks-sandbox/hdp.html.

Spark Installation:

1. Verifying Java Installation

$java -version

2. Downloading Scala

Download the latest version of Scala from https://www.scala-lang.org/download/ .

3. Installing Scala

Extract the Scala tar file : *$ tar xvf scala-2.11.6.tgz*

Set PATH for Scala : *$ export PATH = $PATH:/usr/local/scala/bi*

4. Verifying Scala installation

$scala -version

5. Downloading Apache Spark

Download the latest version of Spark by visiting https://spark.apache.org/downloads.html.

6. Installing Spark: Extracting Spark tar file using the command below:

$ tar xvf spark-1.3.1-bin-hadoop2.6.tgz

7. Moving Spark software files: Use following commands for moving the Spark software files to respective directory (/usr/local/spark).

$ su –

Password:

 # cd /home/Hadoop/Downloads/

 # mv spark-1.3.1-bin-hadoop2.6 /usr/local/spark

 # exit

8. Setting up the environment for Spark by adding the following line to ~/.bashrc file.

export PATH=$PATH:/usr/local/spark/bin

9. Use the following command for sourcing the ~/.bashrc file.

$ source ~/.bashrc

10. Verify the Spark Installation:

Use the following command for opening Spark shell.

$spark-shell

11. On successful installation, you should see a pyspark terminal.

Grab Tweets Data from Twitter API (FLUME Queries):

Commands to Start Apache Flume and Hadoop Services

1. Starting Hadoop services

   start-all.sh

2. URL to access Hadoop via web interface

   http://localhost:50070

3. Create a new directory in Hadoop file system

hadoop fs -mkdir/ <Your Folder Name>

e.g.

hadoop fs -mkdir /twitter_data

4. Starting Flume Agent

./bin/flume-ng agent -n <HDFS AGENT NAME> -c conf -f <FLUME CONFIG FILE>

-Dflume.root.logger=INFO,console

e.g.

./bin/flume-ng agent -n hdfs-agent -c conf -f conf/twitter-flume-hdfs.conf --name TwitterAgent -Dflume.root.logger

Sentiments Analysis (HIVE Queries):

//Table to Store Twitter JSON Data.

CREATE EXTERNAL TABLE covid_raw_tweets(id BIGINT,created_at STRING,source STRING,favorited BOOLEAN, retweeted_status STRUCT<text:STRING, `user` :STRUCT<screen_name:STRING,name:STRING>,retweet_count:INT>,text STRING,entities STRUCT<hashtags:ARRAY<STRUCT<text:STRING>>>,`user` STRUCT<screen_name:STRING,friends_count:INT,followers_count:INT,statuses_count:INT,verified:BOOLEAN,utc_offset:INT,time_zone:STRING>,in_reply_to_screen_name STRING)

ROW FORMAT SERDE 'org.apache.hive.hcatalog.data.JsonSerDe'

// Load the data from HDFS Path to the above-created Table.

load data inpath '/user/maria_dev/covid_data_day (1)' into TABLE covid_raw_tweets;

//Divide the text into words.

create view temp_1 as select id,covid_raw_tweets.text, words from covid_raw_tweets lateral view explode(sentences(lower(text))) dummy as words;

//Divide words into a single word row.

create view temp_2 as select id,temp_1.text, word from temp_1 lateral view explode(words) dummy as word ;

//Now we import our Dicyinary file as well.

CREATE EXTERNAL TABLE dictionary (

type string,

length int,

word string,

pos string,

stemmed string,

polarity string

)

ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t'

STORED AS TEXTFILE

LOCATION '/user/maria_dev/dictionary';

//load dictionary file into the table:

load data inpath '/user/maria_dev/dictinary.tsv' into TABLE covid_raw_tweets;

//Calculate Polarity by joining with dictionary.

create view temp_3 as select

id,temp_2.text,

temp_2.word,

case s_d.polarity

when 'negative' then -1

when 'positive' then 1

else 0 end as polarity

from temp_2 left outer join dictionary s_d on temp_2.word = s_d.word;

//Sum single word polarity value for every single user based on UsedId and assign sentiment.

create table tweets_sent_final stored as orc as select

id,

case

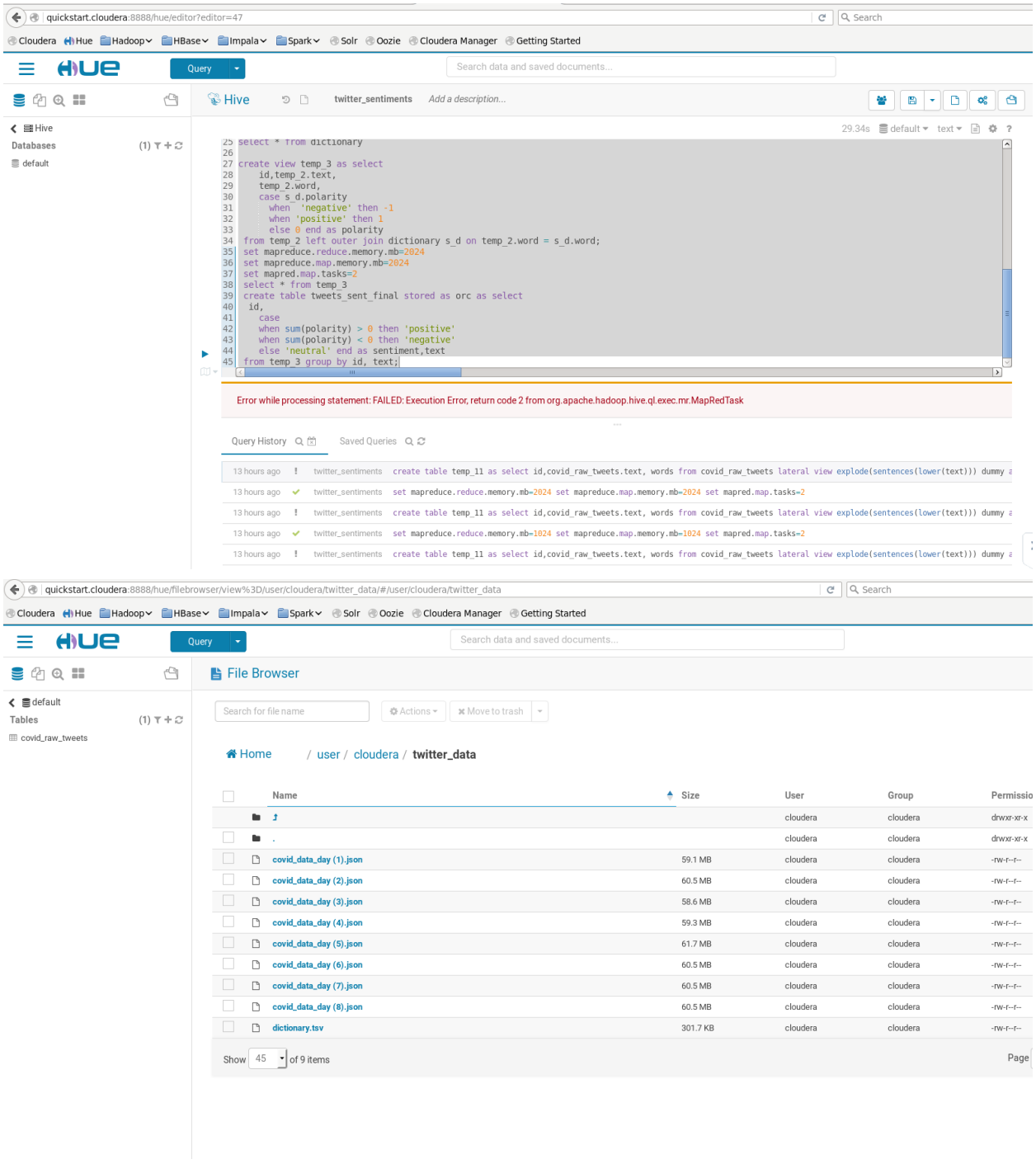
when sum(polarity) > 0 then 'positive'

when sum(polarity) < 0 then 'negative'

else 'neutral' end as sentiment,text

from temp_3 group by id, text;

HUE Screen:



PROBLEMS:

We were trying to work on some live twitter data that's why we have planned to use Twitter Streaming API, we have created an application in the Developers Account and we grab the data into JSON format using FLUME, when we got the data we have now to process it tables using HIVE, that causes us a big problem of

resources limitation we have got in trouble with data as it was a huge amount and the computer resource were unable to process it. We were getting this error.

FAILED Execution Error return code 2 from org apache hadoop hive ql exec mr MapRedTask

We tried it a lot and at the end we reached to the following article.

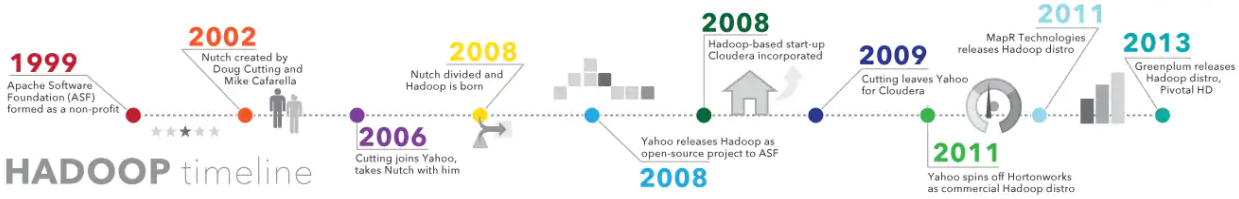
https://www.edureka.co/community/66949/failed-execution-error-return-apache-hadoop-hive-mapredtask#:~:text=This%20issue%20generally%20occurs%20when,Map%20Task'%20from%20existing%20value.&text=Hope%20this%20will%20help%20you.

It clearly states that we need a really good resource to process a huge data, we got a plan to do it on some cloud based HADOOP cluster, but it was costing us a huge amount. That's we need to work on some dataset that is not much bigger but and we can do sentiment analysis on it.

Hadoop

Apache Hadoop is an open-source suite of software utilities that make it easy to use a multicomputer network to solve data and computing problems. It provides a software framework for the process of distributing and storing large data using the Mapreduce programming model.

As the World Wide Web grew in the late 1900s and late 2000s, search engines and indexes were developed to help find relevant information between text-pages. In the early years, people returned search results. But as the Internet grew from tens to millions of pages, automation was needed. Web crawlers were developed, most of which were research projects, and new search engines (Yahoo, AltaVista, etc.) began to emerge.



One such project was an open source search engine called Dutch Cutter and Mike Caffrela's Brain Builder. They wanted to quickly return web search results by spreading data and calculations across different computers so that they could perform more than one task at a time. Now, another search engine project called

Google is being developed. It was based on the same concept - splitting, automatically storing and processing data so that relevant search results could be returned quickly.

In 2006, Cutting joined Yahoo! and, along with the Nich project, embraced the idea of Google's initial work to automate distributed data storage and processing. The Natch Project split: part of the web crawler remained nuts, and part of the distributed computing and processing became (named after the cutting son's toy elephant). In 2008, Yahoo released Hadoop as an open-source project. Today, the Hadoop Framework and Technology Ecosystem is managed and maintained by the nonprofit Apache Software Foundation (ASF), the global community of software developers and partners.

Why is Hadoop important?

Ability to quickly store and process large amounts of any data: This is an important consideration as the amount of data is increasing in various ways, especially from social media and the Internet of Things (IoT).

Computing power: The HUDP grid computing model is faster in processing large data. The more compute nodes you use, the more computational power you have.

Fault tolerance: is protected from data and application processing hardware failures. If the node fails, the jobs are automatically redirected to other nodes to ensure that grid computing does not fail. Multiple copies of all data are automatically saved.

Flexibility: Unlike traditional relative databases, you do not need to pre-process the data before storing it. You can store as much as you want and decide how to use it later. This includes unstructured data such as text, images and videos.

Low cost: open-source platform is free and uses common hardware to store large amounts of data.

Scalability: You can easily extend your system to handle more data by adding nodes. Requires small administration.

MapReduce:

MapReduce is a processing method and programming model for Java-based distributed computing. MapReduce is based on java distributed computing

programming model and processing model. The MapReduce algorithm has two main functions, namely mapping and reduction. The map takes one dataset and converts it into another dataset, where the individual elements are divided into topples (key / value pairs).

Steps to complete a MapReduce Task:

Step 1: A block is processed by a mapper at the same time. In mapper, the developer can define his proposed logic as needed. As such, the map runs across all cluster nodes and processes data blocks in parallel.

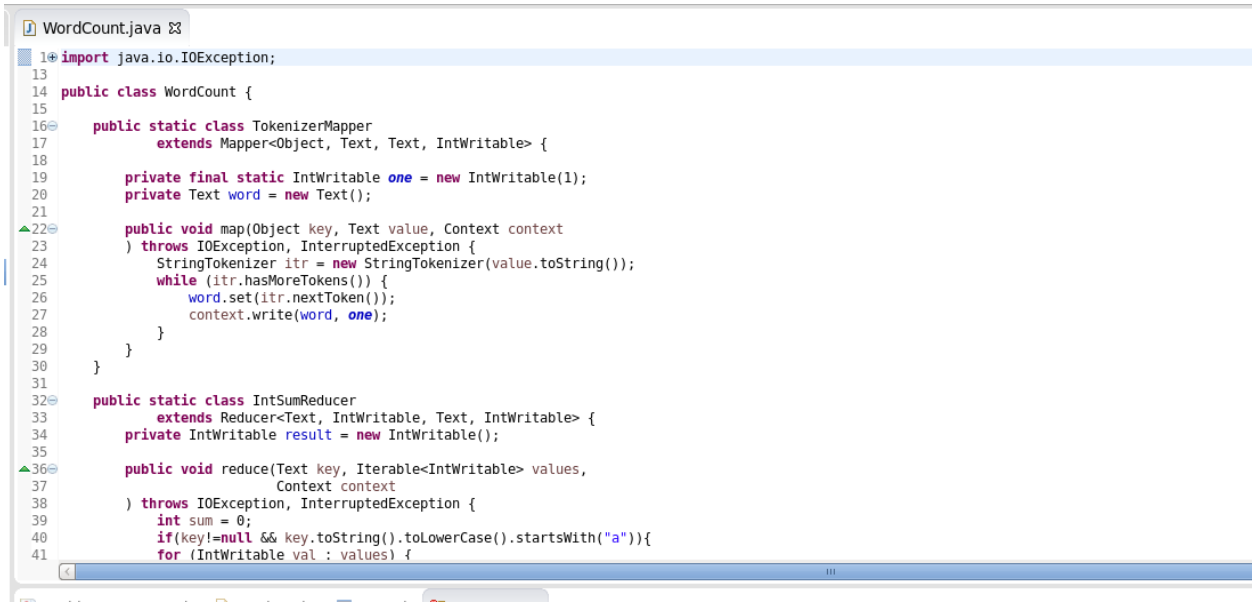
Step 2: Mapper output, also called staging output, is written to the local disk. Mapper output is not stored in HDFS because it is temporary data and writing to HDFS will create unnecessary copies.

Step 3: Output is converted to a reducer node (which is a common slave node, phase will continue here, hence it is called reducer node). Shuffling / copying is the physical movement of data across a network.

Step 4: After all the matches have been completed and their output is adjusted on the reducer nodes, this intermediate output is integrated and configured. Which is then provided as input to reduce the phase.

Step 5: Reduce is the second processing step where the user can articulate their business logic as needed. An input to a reducer is provided from all the mappers. The reducer output is the final output listed on the HDFS.

Java Code (MapReduce WordCount):



hadoop fs -mkdir /user/cloudera/tweets

It will create a directory named tweets under user/cloudera in hadoop file system

hadoop fs -mkdir /user/cloudera/tweets/input

We will create a sub directory named as input to save our input file here

hadoop fs -put text.txt /user/cloudera/tweets/input

It will save the text file in input directory

hadoop jar WordFreqCount.jar WordCount /user/cloudera/tweets/input /user/cloudera/tweets/output

It will run the jar file and count word in our input file and store them to ouput which we can either check through

HUE or we can run the command below

hadoop fs -cat /user/cloudera/tweets/ouput/part-r-00000

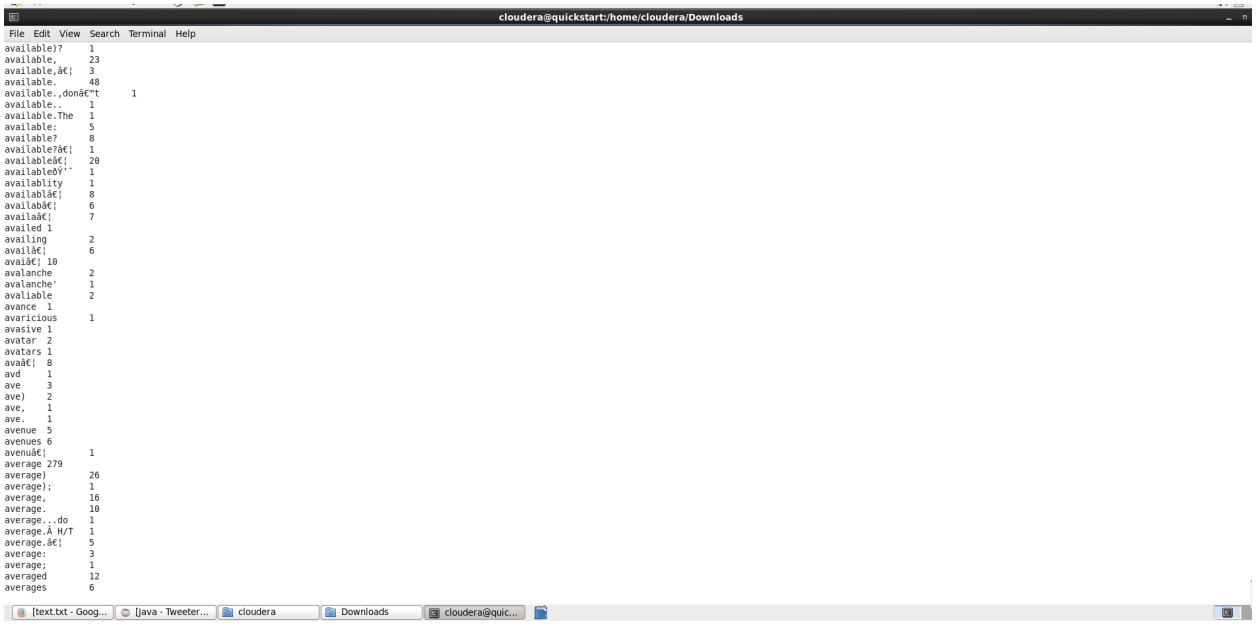
WordCount Output here is showing how many times a certain word was found in the text of tweets we provided from the COVID-19 dataset. From this WordCount we can also create a WordCloud.

hadoop jar WordFreqaCount.jar WordCount /user/cloudera/tweets/input /user/cloudera/tweets/outputa

It will run the jar file and count word starting with a letter 'a' in our input file and store them to ouput which

we can either check through HUE or we can run the command below

hadoop fs -cat /user/cloudera/tweets/ouputa/part-r-00000



WordCount Output here is showing how many times a certain word that starts with a letter a was found in the text of tweets we provided from the COVID-19 dataset. From this WordCount we can also create a WordCloud.

Data Exploration with PySpark Dataframe:

Connecting Drive to Colab

The first thing we want to do when you are working on Colab is mounting your Google Drive. This will enable us to access any directory on your Drive inside the Colab notebook.

```
[1] #connecting to Google Drive
    from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

Setting up PySpark in Colab

Spark is written in the Scala programming language and requires the Java Virtual Machine (JVM) to run. Therefore, our first task is to download Java then Spark. At the end we have installed spark.

```
#installing opendjdk-8 for JAVA
!apt-get install openjdk-8-jdk-headless -qq > /dev/null
```

```
[8] #downloading apache spark
    !wget -q https://downloads.apache.org/spark/spark-3.1.1/spark-3.1.1-bin-hadoop2.7.tgz
```

```
[10] #extracting apache spark
     !tar xf spark-3.1.1-bin-hadoop2.7.tgz
```

```
[11] #intalling apache spark
     !pip install -q findspark
```

Now that we have installed all the necessary dependencies in Colab, it is time to set the environment path. This will enable us to run Pyspark in the Colab environment.

```
[13] #setting JAVA_HOME and SPARK_HOME variables
     import os
     os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
     os.environ["SPARK_HOME"] = "/content/spark-3.1.1-bin-hadoop2.7"
```

Now, we can import SparkSession from pyspark.sql and create a SparkSession, which is the entry point to Spark.

```
[14] #importing findspark library
     import findspark
     findspark.init()
```

```
[15] #finding the spark library path
     findspark.find()
```

```
     '/content/spark-3.1.1-bin-hadoop2.7'
```

```
[16] #creating our first sparksession
     from pyspark.sql import SparkSession

     spark = SparkSession.builder\
             .master("local")\
             .appName("Colab")\
             .config('spark.ui.port', '4050')\
             .getOrCreate()
```

We can get the Spark UI and hence can create a public UI that can be accessed from a public URL.

```
#code to get the Spark UI
!wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
!unzip ngrok-stable-linux-amd64.zip
get_ipython().system_raw('./ngrok http 4050 &')
!curl -s http://localhost:4040/api/tunnels
```

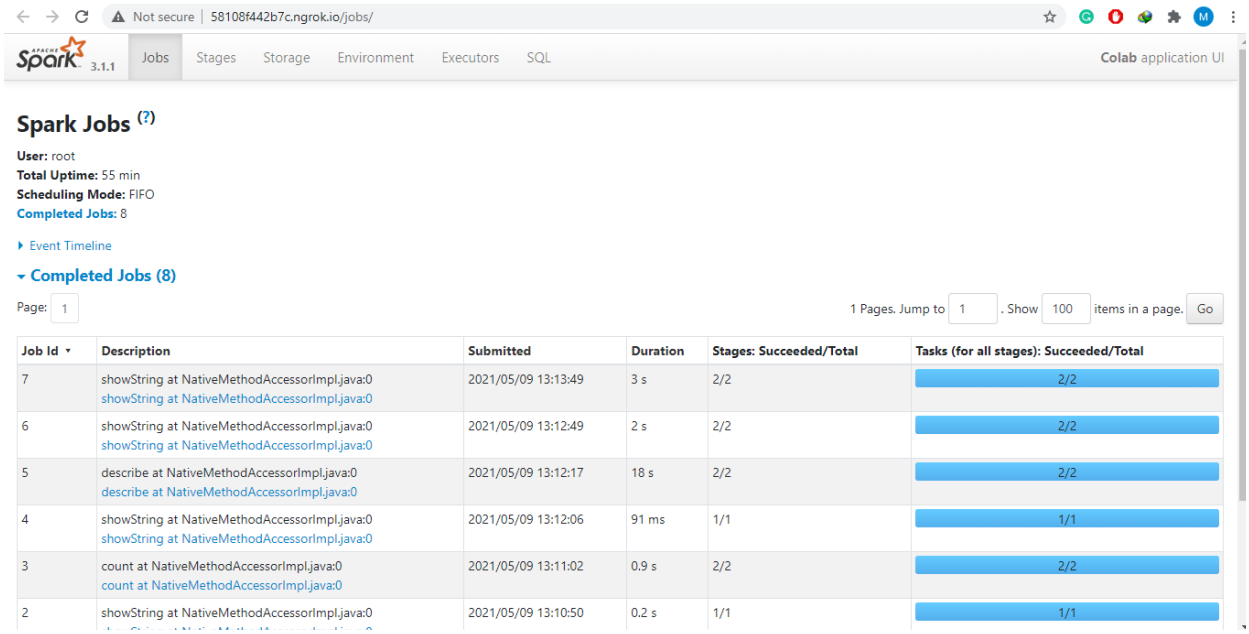
```
--2021-05-09 12:55:50--  https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip
Resolving bin.equinox.io (bin.equinox.io)... 52.45.2.52, 34.235.3.193, 3.222.240.112, ...
Connecting to bin.equinox.io (bin.equinox.io)|52.45.2.52|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13832437 (13M) [application/octet-stream]
Saving to: ‘ngrok-stable-linux-amd64.zip’

ngrok-stable-linux- 100%[===================>]  13.19M  36.5MB/s    in 0.4s

2021-05-09 12:55:51 (36.5 MB/s) - ‘ngrok-stable-linux-amd64.zip’ saved [13832437/13832437]

Archive:  ngrok-stable-linux-amd64.zip
  inflating: ngrok
{"tunnels":[{"name":"command_line","uri":"/api/tunnels/command_line","public_url":"https://58108f442b7c.ngrok.io","proto":'
```

Spark User Interface from our Public URL:



Loading data into PySpark

First thing first, we need to load the dataset. We will use the read.csv module. The inferSchema parameter provided will enable Spark to automatically determine the data type for each column but it has to go over the data once. After this we have shown our Dataframe schema.

Understanding the Data

We have the covid19_tweets dataset from the Kaggle Platform. The dataset contains numerous tweets using hashtags of covid19 all over the world. Our goal from this dataset is to show the countries tweeting more often.

Show column details.

The first step in an exploratory data analysis is to check out the schema of the dataframe. This will give us a bird's-eye view of the columns in the dataframe along with their data types.

```
[20] #read dataset to dataframe
     df = spark.read.csv("/content/drive/MyDrive/covid19/covid19_tweets.csv", header=True, inferSchema=True)
```

```
[21] #Show column details
     df.printSchema()
```

```
root
 |-- user_name: string (nullable = true)
 |-- user_location: string (nullable = true)
 |-- user_description: string (nullable = true)
 |-- user_created: string (nullable = true)
 |-- user_followers: string (nullable = true)
 |-- user_friends: string (nullable = true)
 |-- user_favourites: string (nullable = true)
 |-- user_verified: string (nullable = true)
 |-- date: string (nullable = true)
 |-- text: string (nullable = true)
 |-- hashtags: string (nullable = true)
 |-- source: string (nullable = true)
 |-- is_retweet: string (nullable = true)
```

Display Rows

Now we would obviously want to have a view of the actual data as well.

```
#Just like in Pandas Dataframe we have the df.head() function, here we have the show() function.
df.show(5)
```

```
+---------------+--------------------+--------------------+------------------+--------------+------------+---------------+
|      user_name|       user_location|    user_description|      user_created|user_followers|user_friends|user_favourites|
+---------------+--------------------+--------------------+------------------+--------------+------------+---------------+
|        ‘Ý°i☻L☾T|          astroworld|wednesday addams ...|2017-05-26 05:46:42|           624|         950|          18775|
|Tom Basile us|        New York, NY|Husband, Father, ...|2009-04-16 20:06:23|          2253|        1677|             24|
|Time4fisticuffs|    Pewee Valley, KY|#Christian #Catho...|2009-02-28 18:57:41|          9275|        9525|           7254|
|    ethel mertz|Stuck in the Middle |#Browns #Indians ...|2019-03-07 01:45:06|           197|         987|           1488|
|       DIPR-J&K|   Jammu and Kashmir|🖉Official Twitt...|2017-02-12 06:45:15|        101009|         168|            101|
+---------------+--------------------+--------------------+------------------+--------------+------------+---------------+
only showing top 5 rows
```

Number of rows in Dataframe:

```
[23] #number of rows in the dataframe, which we would, just use the count() function.
     df.count()
```

```
323158
```

Display the specific columns

```
#Using the select() function we can mention any columns we want to view.
df.select("user_name","user_location","user_followers","user_favourites").show(5)
```

```
+---------------+--------------------+--------------+---------------+
|      user_name|       user_location|user_followers|user_favourites|
+---------------+--------------------+--------------+---------------+
|        ‘Ý°i☻L☾T|          astroworld|           624|          18775|
|Tom Basile us|        New York, NY|          2253|             24|
|Time4fisticuffs|    Pewee Valley, KY|          9275|           7254|
|    ethel mertz|Stuck in the Middle |           197|           1488|
|       DIPR-J&K|   Jammu and Kashmir|        101009|            101|
+---------------+--------------------+--------------+---------------+
only showing top 5 rows
```

Describing the columns

Often when we are working with numeric features, we want to have a look at the statistics regarding the dataframe. The describe() function is best suited for such purposes.

It is pretty similar to Panda's describe function but the statistical values are far less and the string columns are described as well.

```
[25] #have a look at the statistics regarding the dataframe. The describe() function is best suited for such purposes.
     df.describe().show()
```

```
+-------+--------------------+------------------+--------------------+--------------------+------------------+-------------
|summary|           user_name|     user_location|    user_description|        user_created|   user_followers|        user_
+-------+--------------------+------------------+--------------------+--------------------+------------------+-------------
|  count|              323145|            223456|              252729|              235222|            185656|
|   mean| 9.512645922259457E9|3.501254379280664E7| 9.751959375058082E7|  13926.520461454516|97222.29477488212|   2351.32161
| stddev|1.390882557242068...| 5.49802029187421E8|1.501099233009359E10|  265865.19227823557| 783647.697905641|  10501.61219
|    min|                    |                  |                    |                    |       #NoMAGAts"|  stupidity
|    max|  Protecting People|                 🍠|BSN, RN. 🎓UC A...|🌊🌊🌊 Blue wave ...|                 🚴|
+-------+--------------------+------------------+--------------------+--------------------+------------------+-------------
```

Distinct values for Categorical columns

```
[26] #The distinct() will come in handy when we want to determine the unique values in the categorical
     #columns in the dataframe.
     df.select("user_location").distinct().show()
```

```
+--------------------+
|       user_location|
+--------------------+
|     Gainesville, FL|
| 2011-06-12 18:18:43|
| 2020-03-30 02:14:49|
|      ['dorathians']|
|            Zirakpur|
|I aint go?...|
|             Ahch-To|
| Mumbai, Maharashtra|
|Perth, Western Au...|
|           Bangalore|
|ÜT: -26.532499,28...|
| 2015-02-01 18:00:41|
| 2020-05-04 09:37:29|
| 2011-07-25 19:44:07|
|    ['alreadyvideo']|
|['mask', 'bighead...|
|['WithYouEverySte...|
|            Talagang|
|Marburg-Biedenkop...|
| 2020-01-28 13:04:13|
+--------------------+
only showing top 20 rows
```

Counting and Removing Null values

Now we all know that real-world data is not oblivious to missing values. Therefore, it is prudent to always check for missing values and remove them if present.

```
[28] #Counting and Removing Null values
     from pyspark.sql import functions as F
     df.select([F.count(F.when(F.isnull(c), c)).alias(c) for c in df.columns]).show()
```

```
+---------+-------------+---------------+------------+-------------+------------+---------------+-------------+------+---
|user_name|user_location|user_description|user_created|user_followers|user_friends|user_favourites|user_verified|  date|  t
+---------+-------------+---------------+------------+-------------+------------+---------------+-------------+------+---
|       13|        99702|          70429|       87936|       137502|      142431|         143662|       143944|159247|156
+---------+-------------+---------------+------------+-------------+------------+---------------+-------------+------+---
```

Geographical Distribution of Tweets Using Pandas DataFrame:

We have loaded our tweets csv file that is covid19_tweets.csv to our DataFrame and its info shows the result like this.

```
[4]  #reading csv fie of dataset
     df = pd.read_csv('covid19_tweets.csv')
```

```
[5]  #getting info of dataframes
     df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 179108 entries, 0 to 179107
Data columns (total 13 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   user_name         179108 non-null  object
 1   user_location     142337 non-null  object
 2   user_description  168822 non-null  object
 3   user_created      179108 non-null  object
 4   user_followers    179108 non-null  int64
 5   user_friends      179108 non-null  int64
 6   user_favourites   179108 non-null  int64
 7   user_verified     179108 non-null  bool
 8   date              179108 non-null  object
 9   text              179108 non-null  object
 10  hashtags          127774 non-null  object
 11  source            179031 non-null  object
 12  is_retweet        179108 non-null  bool
dtypes: bool(2), int64(3), object(8)
memory usage: 15.4+ MB
```

Now we want to get the description of data for which we got the below results.

```
[13] #describing our dataframes
     df.describe()
```

	user_followers	user_friends	user_favourites
count	9.913800e+04	99138.000000	9.913800e+04
mean	1.550841e+05	2374.224334	1.517543e+04
std	1.012846e+06	10019.923883	4.490681e+04
min	0.000000e+00	0.000000	0.000000e+00
25%	3.220000e+02	210.000000	3.330000e+02
50%	1.546000e+03	666.000000	2.184000e+03
75%	7.576000e+03	1951.000000	1.038475e+04
max	2.435916e+07	497363.000000	2.047197e+06

The tweets which are null will be discarded and it shows the results like below for all the tweets counting the number of entries.

```
#counting the total enteries against each column
df.isnull().count()
```

```
user_name           99138
user_location       99138
user_description    99138
user_created        99138
user_followers      99138
user_friends        99138
user_favourites     99138
user_verified       99138
date                99138
text                99138
hashtags            99138
source              99138
is_retweet          99138
dtype: int64
```

Now our task to perform some sentiment analysis on this dataframe so for that we have planned to process the geographical information in the form of graph stating the tweets we got in a specific month from a certain region.

```
[15] #dropping na enteries
     df = df.dropna()
```

```
[16] #get a unique month
     pd.DatetimeIndex(df['date']).month.unique()
```

```
     Int64Index([7, 8], dtype='int64', name='date')
```

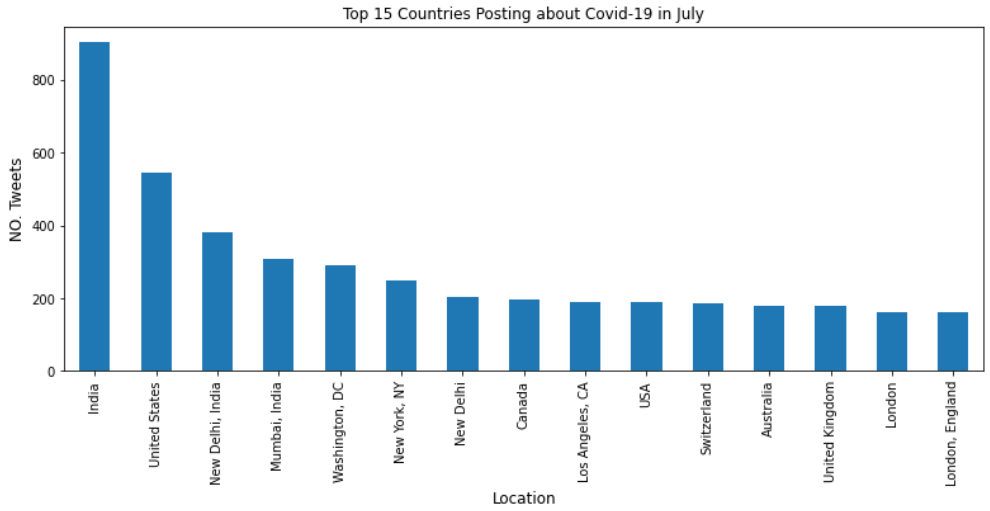
```
[17] #creating method to read the location values count for 7th and 8th month
     top_july = df['user_location'][pd.DatetimeIndex(df['date']).month == 7].value_counts()
     top_august = df['user_location'][pd.DatetimeIndex(df['date']).month == 8].value_counts()
     top_all_the_time = (top_august + top_july).sort_values(ascending = False)
```

```
[18] #plotting july tweets
     fig, ax = plt.subplots(figsize = (13,5))
     plt.xlabel("Location", fontsize = 12)
     plt.ylabel("NO. Tweets", fontsize = 12)
     top_july[0:15].plot(kind='bar', title = "Top 15 Countries Posting about Covid-19 in July", )
```

The tweets from month July are below:

```
18] fig, ax = plt.subplots(figsize = (13,5))
   plt.xlabel("Location", fontsize = 12)
   plt.ylabel("NO. Tweets", fontsize = 12)
   top_july[0:15].plot(kind='bar', title = "Top 15 Countries Posting about Covid-19 in July", )
```

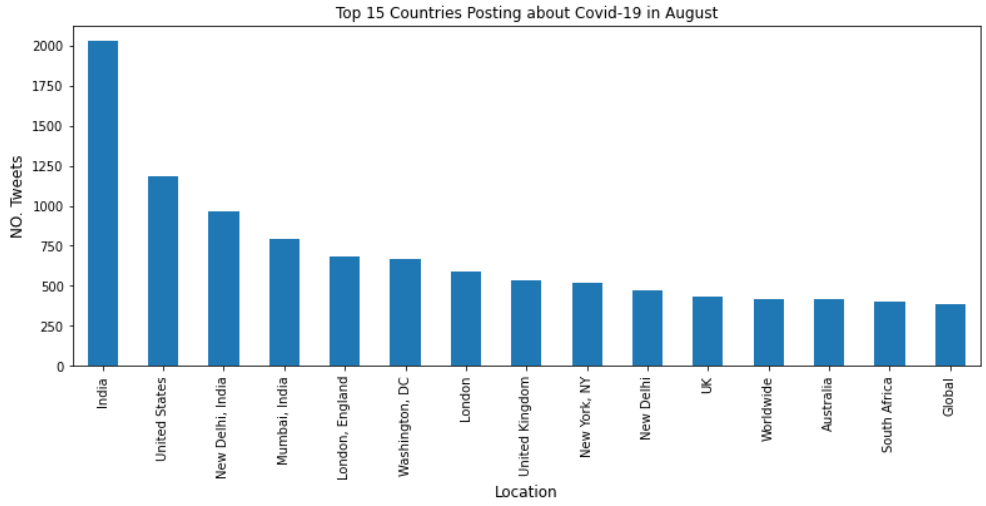
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d597744d0>
```



The tweets from month August are below:

```
[19] fig, ax = plt.subplots(figsize = (13,5))
     plt.xlabel("Location", fontsize = 12)
     plt.ylabel("NO. Tweets", fontsize = 12)
     top_august[0:15].plot(kind='bar', title = "Top 15 Countries Posting about Covid-19 in August", )
```

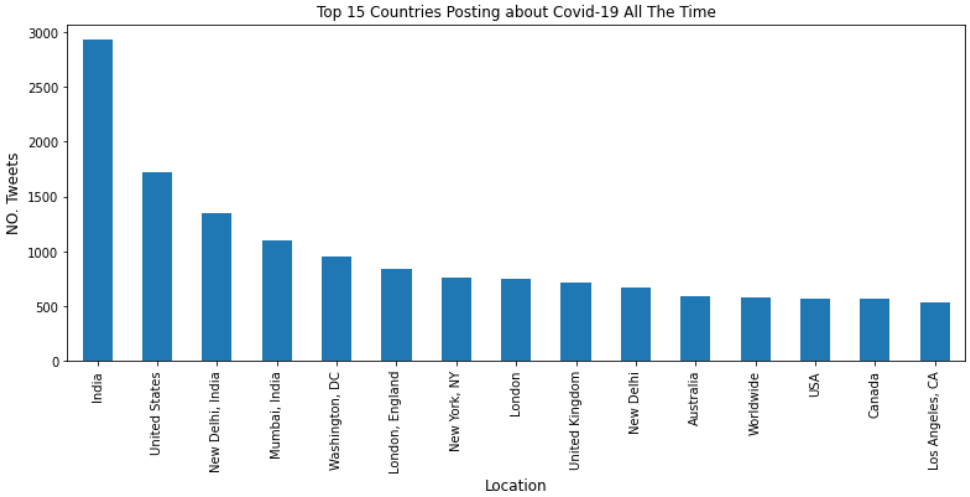
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d5bde0b50>
```



The all the time tweets are given below.

```
[20] fig, ax = plt.subplots(figsize = (13,5))
     plt.xlabel("Location", fontsize = 12)
     plt.ylabel("NO. Tweets", fontsize = 12)
     top_all_the_time[0:15].plot(kind='bar', title = "Top 15 Countries Posting about Covid-19 All The Time", )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d596c5590>
```



CASSANDRA:

Sentiment analysis deals with extracting sentiments from the input file. Here we are using Twitter API to pull out all the required data from the source.
For my implementation, I have focus on text analysis with many of the combinations of technologies. These are as follows:

1. **Technologies** - Cassandra, Spark , PySpark, DataStax Enterprise Analytics, Python, Tweepy
2. **Notebooks/Env** - Jupyter Notebook, Docker
3. **Visualization Tools** - Tableau

DATASET:

I have used the live Twitter API to pull all tweets. Based on these tweets I have defined a variable title which will pull all tweets based on COVID19. The terms used for positive are recovery, vaccinated, relax, strong, cheers, blessed, care, quarantine and chill, social distancing and for negative tweets I have used deaths, frustrated, terrible, unfocussed, fear, no vaccine, stress, anxiety, worry, hardship, covididiots.
Since the Tweepy Twitter API allows me to pull only 100 tweets per hour. I have reduced the terms of positive and negative. Just worked on recovery , vaccinated, death and frustrated as respectively.

IMPLEMENTATION:

Steps involved in Implementations

1. Setup the Docker environment in my local system, this will help me to run the entire application of packages in a container [4].
2. Docker allows me to compose the jupyter shell and I can now add and run my logics inside the jupyter environment [7].
3. Added and Installed pySpark and py4J library in Jupyter.
4. Time to import all the packages. I have added pandas, cassandra, tweepy, Spark package, Tokenizer, StopWordsRemover to extract the functions of the entire code.
5. Get twitter keys to access twitter's API to extract tweets.
6. Integrating spark cassandra with datastax I have connected into DSE cluster which will help us to connect to cassandra database. This will indeed generate a session, and based on this session object we can run our own queries [5].

7. Now creating Cassandra keyspace, creating tables in Apache cassandra.
8. After pulling all the tweets, I have cleaned and preprocess the data based on emoji, special characters, retweets.
9. Once the cassandra table has been generated I have created a spark session and created dataframes [6].
10. For implementing tweets tokens, I have used word tokenizer and stopwordsremover to help me get single words from the tweets.
11. By using the python package i have derived sentiment analysis based on the score (value above 0 is positive and value below 0 is negative) of the positive and negative words.
12. Positive function will return positive tweets , negative function will return negative tweets and assessment will give defined words based on to judge the score of each word [8].
13. Based on the record results, I have plotted the graphs in Tableau.

ALGORITHM:
Algorithm/PseudoCode

1. Importing all libraries

Importing all the Libraries ¶

```
In [ ]: # !pip install geotext

In [1]: pysparkzip = "/opt/dse/resources/spark/python/lib/pyspark.zip"
        py4jzip = "/opt/dse/resources/spark/python/lib/py4j-0.10.4-src.zip"

In [2]: # Needed to be able to find pyspark libaries
        import sys
        sys.path.append(pysparkzip)
        sys.path.append(py4jzip)

In [233]: import pandas
          import cassandra
          import pyspark
          import tweepy
          import re
          import os
          from IPython.display import display, Markdown
          from pyspark.sql import SparkSession
          from pyspark.ml.feature import Tokenizer, RegexTokenizer, StopWordsRemover
          from pyspark.sql.functions import col, udf
          from pyspark.sql.types import IntegerType
          from pattern.en import sentiment, positive
          from geotext import GeoText
          import unicodedata
```

2. Importing Twitter API Keys and Tokens

Importing Twitter API keys and tokens

In [4]:
```
consumer_key = os.environ['CONSUMER_KEY']
consumer_secret = os.environ['CONSUMER_SECRET']
access_token = os.environ['ACCESS_TOKEN']
access_token_secret = os.environ['ACCESS_TOKEN_SECRET']


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

api = tweepy.API(auth)
```

3. Function to format Spark Dataframes

Function to format Spark DataFrames ¶

In [5]:
```
def showDF(df, limitRows =  5, truncate = True):
    if(truncate):
        pandas.set_option('display.max_colwidth', 50)
    else:
        pandas.set_option('display.max_colwidth', -1)
    pandas.set_option('display.max_rows', limitRows)
    display(df.limit(limitRows).toPandas())
    pandas.reset_option('display.max_rows')
```

4. Creating Cassandra Table, Loading Tables and Pulling Tweets

Creating Tables, Pulling Tweets, and Loading Tables

Connect to Cluster

In [6]:
```
from cassandra.cluster import Cluster

cluster = Cluster(['dse'])
session = cluster.connect()
```

Keyspace

In [7]:
```
session.execute("""
    CREATE KEYSPACE IF NOT EXISTS sentimentanalytics
    WITH REPLICATION =
    { 'class' : 'SimpleStrategy', 'replication_factor' : 1 }"""
)
```

Out[7]:
```
<cassandra.cluster.ResultSet at 0x7ff68c0c9b50>
```

Setting keyspace

In [8]:
```
session.set_keyspace('sentimentanalytics')
```

In [9]:
```
tweetTitle = "covid19"
```

In [10]:
```
positiveNegative = ["positive", "negative"]
```

In [100]:
```
query = "DROP TABLE sentiment_%s_%s" % (tweetTitle, positiveNegative[0])
session.execute(query)
query = "DROP TABLE sentiment_%s_%s" % (tweetTitle, positiveNegative[1])
session.execute(query)



for emotion in positiveNegative:
    query = "CREATE TABLE IF NOT EXISTS sentiment_%s_%s (twitterid bigint, tweet text, city text, country text,\
    user text, createdat text, hashtagkeys text, source text, retweets int, userstatuscount int, PRIMARY KEY \
    ((twitterid, tweet, city, country, user, hashtagkeys, source, retweets, userstatuscount), createdat)) with \
    clustering order by (createdat desc)" % (tweetTitle, emotion)
    print query
    session.execute(query)
```

```
CREATE TABLE IF NOT EXISTS sentiment_covid19_positive (twitterid bigint, tweet text, city text, country text,    user
text, createdat text, hashtagkeys text, source text, retweets int, userstatuscount int, PRIMARY KEY     ((twitterid,
tweet, city, country, user, hashtagkeys, source, retweets, userstatuscount), createdat)) with     clustering order by
(createdat desc)
CREATE TABLE IF NOT EXISTS sentiment_covid19_negative (twitterid bigint, tweet text, city text, country text,    user
text, createdat text, hashtagkeys text, source text, retweets int, userstatuscount int, PRIMARY KEY     ((twitterid,
```

5. Preprocess the Data

```
def cleanUpTweet(tweet):
    if tweet == None:
        return 'unknown'

    emoji_pattern = re.compile(
    u"(\ud83d[\ude00-\ude4f])|"
    u"(\ud83c[\udf00-\uffff])|"
    u"(\ud83d[\u0000-\uddff])|"
    u"(\ud83d[\ude80-\udeff])|"
    u"(\ud83c[\udde0-\uddff])"
    "+", flags=re.UNICODE)

#     removeSpecial = re.compile ('[\n|#|@|!|.|?|,|\"]')
    removeSpecial = re.compile ('[\n|@|!|.|?|,|\"]')
    removeHttp = re.compile("http\S+ | https\S+")
    removeRetweet = re.compile("RT")

    noemoji = emoji_pattern.sub(r'', tweet)
    nospecial = removeSpecial.sub(r'', noemoji)
    nohttp = removeHttp.sub(r'', nospecial)
    noretweet = removeRetweet.sub(r'', nohttp)

    cleanTweet=noretweet
#     cleanTweet = ''.join([x for x in cleanTweet if x.isalnum()])
#     cleanTweet = re.sub(r"[^a-zA-Z0-9:.,@#&]+", ' ', cleanTweet)
    cleanTweet = unicodedata.normalize('NFKD', cleanTweet).encode('ascii', 'ignore')
    return cleanTweet


def cleanUpPlace(tweet):
    if tweet == None:
        return 'unknown'
    tweet = tweet.title()
    places = GeoText(tweet)
    cities = list(places.cities)
    countries = list(places.countries)
    city, country = 'unknown', 'unknown'
    if len(cities) > 0:
        city = str(cities[0])
    if len(countries) > 0:
        country = str(countries[0])
    return (city, country)

def cleanHashtags(hashtag_dict):
    hashtags = []
    for h_dict in hashtag_dict:
        for h in h_dict.values():
            if type(h) == unicode:
                decoded = unicodedata.normalize('NFKD', h).encode('ascii', 'ignore')
                hashtags.append(decoded)

    if len(hashtags) == 0:
        hashtags = ['unknown']

    hashtags = ','.join(hashtags)
    return hashtags
```

6. Pulling data from twitter ID and inserting into Cassandra

In [104]:
```
# Query 1
for emotion in positiveNegative:
    print("\n... Checking the conents of {} table".format(emotion))
    query = 'SELECT * FROM sentiment_%s_%s limit 2' % (tweetTitle, emotion)
    rows = session.execute(query)
    for user_row in rows:
        print(user_row)
```

```
... Checking the conents of positive table
Row(twitterid=1390766965292101633, tweet=u" dw_europe: EU leaders are meeting in Portugal to discuss the bloc's socia
l affairs strategy against the backdrop of the #COVID19 pandem...", city=u'unknown', country=u'unknown', user=u'JM Ha
milton', hashtagkeys=u'COVID19', source=u'Twitter Web App', retweets=9, userstatuscount=621164, createdat=u'2021-05-0
7')
Row(twitterid=1390759589310865415, tweet=u" DrAmbrishMithal: Don't chase #PCest after recoveryRepeat #PCR NOT necessa
ry&gt;10 days after onset of symptoms&gt;3 days after absen...", city=u'Noida', country=u'India', user=u'Nikhil katiy
ar', hashtagkeys=u'RTPCRTest,RTPCR', source=u'Twitter for iPhone', retweets=127, userstatuscount=494, createdat=u'202
1-05-07')

... Checking the conents of negative table
Row(twitterid=1390610581359562756, tweet=u' foogatwo: #NJ #COVID19 Case update for 5/5 Mass Deletions continuing Grou
ndhog day Growing increasingly frustrated with inability of...', city=u'unknown', country=u'unknown', user=u'Sunny mo
onshine', hashtagkeys=u'NJ,COVID19', source=u'Twitter Web App', retweets=76, userstatuscount=885, createdat=u'2021-05
-07')
Row(twitterid=1390776623432179715, tweet=u' DrEricDing: HUGE EXCESS DEATHSglobal #COVID19 death toll is 2x higher tha
n official data69 mil worldwide Deaths:US 905kIndi...', city=u'unknown', country=u'unknown', user=u'Pablo Miguel', ha
shtagkeys=u'COVID19', source=u'Twitter Web App', retweets=1394, userstatuscount=1364, createdat=u'2021-05-07')
```

7. Performing Different Query Analysis
   a) Get all data from sentiment table

In [104]:
```
# Query 1
for emotion in positiveNegative:
    print("\n... Checking the conents of {} table".format(emotion))
    query = 'SELECT * FROM sentiment_%s_%s limit 2' % (tweetTitle, emotion)
    rows = session.execute(query)
    for user_row in rows:
        print(user_row)
```

```
... Checking the conents of positive table
Row(twitterid=1390766965292101633, tweet=u" dw_europe: EU leaders are meeting in Portugal to discuss the bloc's socia
l affairs strategy against the backdrop of the #COVID19 pandem...", city=u'unknown', country=u'unknown', user=u'JM Ha
milton', hashtagkeys=u'COVID19', source=u'Twitter Web App', retweets=9, userstatuscount=621164, createdat=u'2021-05-0
7')
Row(twitterid=1390759589310865415, tweet=u" DrAmbrishMithal: Don't chase #PCest after recoveryRepeat #PCR NOT necessa
ry&gt;10 days after onset of symptoms&gt;3 days after absen...", city=u'Noida', country=u'India', user=u'Nikhil katiy
ar', hashtagkeys=u'RTPCRTest,RTPCR', source=u'Twitter for iPhone', retweets=127, userstatuscount=494, createdat=u'202
1-05-07')

... Checking the conents of negative table
Row(twitterid=1390610581359562756, tweet=u' foogatwo: #NJ #COVID19 Case update for 5/5 Mass Deletions continuing Grou
ndhog day Growing increasingly frustrated with inability of...', city=u'unknown', country=u'unknown', user=u'Sunny mo
onshine', hashtagkeys=u'NJ,COVID19', source=u'Twitter Web App', retweets=76, userstatuscount=885, createdat=u'2021-05
-07')
Row(twitterid=1390776623432179715, tweet=u' DrEricDing: HUGE EXCESS DEATHSglobal #COVID19 death toll is 2x higher tha
n official data69 mil worldwide Deaths:US 905kIndi...', city=u'unknown', country=u'unknown', user=u'Pablo Miguel', ha
shtagkeys=u'COVID19', source=u'Twitter Web App', retweets=1394, userstatuscount=1364, createdat=u'2021-05-07')
```

   b) Find out maximum number of times a particular hashtags has been used and the user

In [106]:
```
# Query 2: Find out the maximum number of times a particular hashtag has been used and the user
# which has maximum retweets with that keyword in both the positive and negative table
for emotion in positiveNegative:
    print emotion
    query = "SELECT max(retweets), max(user) FROM sentiment_%s_%s where hashtagkeys='COVID19' ALLOW FILTERING" % (twee
    rows = session.execute(query)
    for user_row in rows:
        print(user_row)
```

```
positive
Row(system_max_retweets=1132, system_max_user=u'sic')
negative
Row(system_max_retweets=1394, system_max_user=u'stay blazin')
```

c) Finding Distinct values based on Country

In [107]:
```
# Query 3: Finding distinct values based on country
for emotion in positiveNegative:
    print '\n',emotion
    query = "SELECT DISTINCT twitterid, tweet, city, country, user, hashtagkeys, source, retweets, userstatuscount \
    FROM sentiment_%s_%s where country='India' LIMIT 2 ALLOW FILTERING" % (tweetTitle, emotion)
    rows = session.execute(query)
    for user_row in rows:
        print(user_row)
```

```
positive
Row(twitterid=1390759589310865415, tweet=u" DrAmbrishMithal: Don't chase #PCest after recoveryRepeat #PCR NOT necessa
ry&gt;10 days after onset of symptoms&gt;3 days after absen...", city=u'Noida', country=u'India', user=u'Nikhil katiy
ar', hashtagkeys=u'RTPCRTest,RTPCR', source=u'Twitter for iPhone', retweets=127, userstatuscount=494)
Row(twitterid=1390777601547931648, tweet=u' sanjuydv: Shamefully Bihar stands at 8th rank in wastage of life saving C
ovid19 vaccines in IndiaMerely 766 lakh out of 13 Cr hv be...', city=u'Katihar', country=u'India', user=u'Suraj kuma
r', hashtagkeys=u'unknown', source=u'Twitter for Android', retweets=854, userstatuscount=533)

negative
Row(twitterid=1390688541802258436, tweet=u'shalmida drakchaurasia Sir He is in depressionFrustrated guyIgnore himMay
God bless him with Covid19 and...', city=u'Ahmedabad', country=u'India', user=u'KUNDALINIYOGA #BabubhaiThakkar', hash
tagkeys=u'unknown', source=u'Twitter for Android', retweets=0, userstatuscount=3110)
Row(twitterid=1390607576828624898, tweet=u'Young Thais take to social media to vent against their government while st
ressing in post after post that they were...', city=u'unknown', country=u'India', user=u'UCAN India', hashtagkeys=u'u
nknown', source=u'SocialPilot.co', retweets=0, userstatuscount=19576)
```

d) Finding number of people tweeting positive and negative for two different countries

In [221]:
```
# Query 4: Finding number of people tweeting positive and negative stuff from two different countries
for emotion in positiveNegative:
    print emotion
    query = "SELECT count(country) as Indians FROM sentiment_%s_%s where country='India' ALLOW FILTERING" % (tweetTitl
    rows = session.execute(query)
    for user_row in rows:
        print(user_row)

    query = "SELECT count(country) as Canadians FROM sentiment_%s_%s where country='Canada' ALLOW FILTERING" % (tweetT
    rows = session.execute(query)
    for user_row in rows:
        print(user_row)
```

```
positive
Row(indians=13)
Row(canadians=8)
negative
Row(indians=3)
Row(canadians=3)
```

e) Finding the number of tweets from a particular platform

In [222]:
```
# Query 5: Finding the number of tweets from a particular platform
for emotion in positiveNegative:
    print emotion
    query = "SELECT count(*) FROM sentiment_%s_%s where source='Twitter Web App' ALLOW FILTERING" % (tweetTitle, emoti
    rows = session.execute(query)
#     print(rows)
    for user_row in rows:
        print(user_row)
```

```
positive
Row(count=65)
negative
Row(count=53)
```

f) Finding number of tweets with retweets>25

In [223]:
```
# Query 6: Finding number of tweets with retweets > 25
for emotion in positiveNegative:
    print emotion
    query = "SELECT count(*) FROM sentiment_%s_%s where retweets>25 ALLOW FILTERING" % (tweetTitle, emotion)
    rows = session.execute(query)
#     print(rows)
    for user_row in rows:
        print(user_row)
```

```
positive
Row(count=80)
negative
Row(count=124)
```

8. Creating Spark session which is connected to Cassandra and Counting the number of rows in each

Creating a spark session which is connected to Cassandra and counting the number of rows in each.

In [225]:
```
countTokens = udf(lambda words: len(words), IntegerType())

# spark = SparkSession.builder.appName('demo').master("local").getOrCreate()
spark = SparkSession.builder.appName('demo').master("dse://dse:9042").getOrCreate()

tableNamePos = "sentiment_%s_positive" % (tweetTitle.lower())
tableNameNeg = "sentiment_%s_negative" % (tweetTitle.lower())

tablepos = spark.read.format("org.apache.spark.sql.cassandra").options(table=tableNamePos, keyspace="sentimentanalytic
tableneg = spark.read.format("org.apache.spark.sql.cassandra").options(table=tableNameNeg, keyspace="sentimentanalytic

print "Postive Table Count: "
print tablepos.count()
print "Negative Table Count: "
print tableneg.count()
print(tablepos)
display(tablepos)
```

```
Postive Table Count:
220
Negative Table Count:
209
DataFrame[twitterid: bigint, tweet: string, city: string, country: string, user: string, hashtagkeys: string, source:
string, retweets: int, userstatuscount: int, createdat: string]

DataFrame[twitterid: bigint, tweet: string, city: string, country: string, user: string, hashtagkeys: string, source:
string, retweets: int, userstatuscount: int, createdat: string]
```

9. Converting a sentence into word tokenizer

Sentence to words tokenizer

```
In [227]: okenizerPos = Tokenizer(inputCol="tweet", outputCol="tweetwords")
          okenizedPos = tokenizerPos.transform(tablepos)
          fPos = tokenizedPos.select("tweetwords", "city", "country", "createdat", "hashtagkeys", "source", "retweets", "userstat
          howDF(dfPos)

          okenizerNeg = Tokenizer(inputCol="tweet", outputCol="tweetwords")
          okenizedNeg = tokenizerNeg.transform(tableneg)
          fNeg = tokenizedNeg.select("tweetwords", "city", "country", "createdat", "hashtagkeys","source", "retweets", "userstatu
          howDF(dfNeg)

          rint(dfPos.count())
          rint(dfNeg.count())
```

	tweetwords	city	country	createdat	hashtagkeys	source	retweets	userstatuscount	tokens
0	[, binancebcf:, #binance, charity's, year-long...	unknown	unknown	2021-05-07	Binance	Twitter for iPhone	4	617	18
1	[, cdcgov:, at, the, start, of, 2021, people, ...	unknown	unknown	2021-05-07	COVID19	Twitter Web App	37	186560	24
2	[, wef:, two, key, conditions, for, a, post-pa...	unknown	unknown	2021-05-07	covid19,economics	Twitter for iPad	12	15526	13
3	[#corona, info, for, #usa:new, cases:, 27163to...	Karlsruhe	Germany	2021-05-07	Corona,USA	corona-tracker-app	0	13927	14
4	[, ethicalsid:, one, can, feel, in, saurabh_ml...	Dubai	unknown	2021-05-07	COVID19	Twitter for iPhone	151	2770	23

	tweetwords	city	country	createdat	hashtagkeys	source	retweets	userstatuscount	tokens
0	[05/07/2021, update:people, tested:, 905825, (...	unknown	unknown	2021-05-07	unknown	ri_covid_19	0	315	11
1	[, jkwan_md:, may, 7:, #covid19, in, #ontario3...	unknown	Canada	2021-05-07	COVID19,Ontario	Twitter Web App	102	348030	15
2	[robking65, natwittee, ctvnews, none, it's, a,...	Toronto	Canada	2021-05-07	unknown	Twitter for Android	0	1540	17
3	[#alberta, restaurant, owners, frustrated, as,...	unknown	unknown	2021-05-06	Alberta,COVID19	Twitter for iPhone	0	43243	14
4	[look, what, u, voted, 4, a, man, without, an,...	unknown	unknown	2021-05-07	unknown	Twitter for Android	2	232906	21

```
220
209
```

10. Removing all StopWords

Removing stop words

```
In [228]: removerPos = StopWordsRemover(inputCol="tweetwords", outputCol="tweetnostopwords")
          removedPos = removerPos.transform(dfPos)
          dfPosStop = removedPos.select("tweetwords", "tweetnostopwords", "city", "country", "createdat", "hashtagkeys","source"
          showDF(dfPosStop)

          removerNeg = StopWordsRemover(inputCol="tweetwords", outputCol="tweetnostopwords")
          removedNeg = removerNeg.transform(dfNeg)
          dfNegStop = removedNeg.select("tweetwords", "tweetnostopwords", "city", "country", "createdat", "hashtagkeys","source"
          showDF(dfNegStop)

          print(dfPosStop.count())
          print(dfNegStop.count())
```

	tweetwords	tweetnostopwords	city	country	createdat	hashtagkeys	source	retweets	userstatuscount	tokens	notokens
0	[, binancebcf:, #binance, charity's, year-long...	[, binancebcf:, #binance, charity's, year-long...	unknown	unknown	2021-05-07	Binance	Twitter for iPhone	4	617	18	15
1	[, cdcgov:, at, the, start, of, 2021, people, ...	[, cdcgov:, start, 2021, people, 65+, half, #c...	unknown	unknown	2021-05-07	COVID19	Twitter Web App	37	186560	24	12
2	[, wef:, two, key, conditions, for, a, post-pa...	[, wef:, two, key, conditions, post-pandemic, ...	unknown	unknown	2021-05-07	covid19,economics	Twitter for iPad	12	15526	13	11
3	[#corona, info, for, #usa:new, cases:, 27163to...	[#corona, info, #usa:new, cases:, 27163today, ...	Karlsruhe	Germany	2021-05-07	Corona,USA	corona-tracker-app	0	13927	14	13
4	[, ethicalsid:, one, can, feel, in, saurabh_ml...	[, ethicalsid:, one, feel, saurabh_mlagk's, vo...	Dubai	unknown	2021-05-07	COVID19	Twitter for iPhone	151	2770	23	14

	tweetwords	tweetnostopwords	city	country	createdat	hashtagkeys	source	retweets	userstatuscount	tokens	notokens
0	[05/07/2021, update:people, tested:, 905825, (...	[05/07/2021, update:people, tested:, 905825, (...	unknown	unknown	2021-05-07	unknown	ri_covid_19	0	315	11	11
1	[, jkwan_md:, may, 7:, #covid19, in, #ontario3...	[, jkwan_md:, may, 7:, #covid19, #ontario3166,...	unknown	Canada	2021-05-07	COVID19,Ontario	Twitter Web App	102	348030	15	14
2	[robking65, natwittee, ctvnews, none, it's, a,...	[robking65, natwittee, ctvnews, none, fair, qu...	Toronto	Canada	2021-05-07	unknown	Twitter for Android	0	1540	17	14
3	[#alberta, restaurant, owners, frustrated, as,...	[#alberta, restaurant, owners, frustrated, #co...	unknown	unknown	2021-05-06	Alberta,COVID19	Twitter for iPhone	0	43243	14	11
4	[look, what, u, voted, 4, a, man, without, an,...	[look, u, voted, 4, man, without, ounce, consc...	unknown	unknown	2021-05-07	unknown	Twitter for Android	2	232906	21	14

11. Finding positive negative and neutral tweets

Finding the positive, negative and neutral tweets

Negative Tweets

In [229]:
```
pandaNeg = dfNegStop.toPandas()
sentimentScoreNeg = 0
countNeg = 0
numTweets = 0
negList = list()
netralList = list()

for index, row in pandaNeg.iterrows():
    if positive(row["tweetnostopwords"], .1):
        countNeg = countNeg + 1
    scoreNeg = sentiment(row['tweetnostopwords'])[0]
    if scoreNeg <= 0:
        negList.append((row['country'], row['city'], row['createdat'], row['hashtagkeys'], sentiment(row["tweetnostopw
                        sentiment(row['tweetnostopwords']).assessments, row['source'], row['retweets'], row['userstatu
        sentimentScoreNeg = scoreNeg + sentimentScoreNeg
    else:
        netralList.append((row['country'], row['city'], row['createdat'], row['hashtagkeys'], sentiment(row["tweetnost
                           sentiment(row['tweetnostopwords']).assessments, row['source'], row['retweets'], row['userst

labels = ['Country', 'City', 'Create_At', 'Hashtags Keyword', 'Sentiment Score', 'Postive', 'Assessments', 'Source', '
negativeTweetScores = pandas.DataFrame.from_records(negList, columns=labels)
negativeTweetScores
```

ut[229]:

	Country	City	Create_At	Hashtags Keyword	Sentiment Score	Postive	Assessments	Source	Retweets	S
0	unknown	unknown	2021-05-07	unknown	(0.0, 0.0)	False	[]	ri_covid_19	0	
1	Canada	unknown	2021-05-07	COVID19,Ontario	(0.0, 0.0)	False	[]	Twitter Web App	102	34
2	unknown	unknown	2021-05-06	Alberta,COVID19	(-0.7, 0.2)	False	[([frustrated], -0.7, 0.2, None)]	Twitter for iPhone	0	4
3	unknown	unknown	2021-05-07	unknown	(0.0, 0.0)	False	[]	Twitter for Android	2	23

12. Positive Tweets

Positive Tweet

Also adding up all the sentiment scores of all the tweets

In [230]:
```
pandaPos = dfPosStop.toPandas()
sentimentScore = 0
countPos = 0
poslist = list()

for index, row in pandaPos.iterrows():
    if not positive(row["tweetnostopwords"]) and sentiment(row["tweetnostopwords"])[0] != 0.0:
        countPos = countPos + 1
    score = sentiment(row['tweetnostopwords'])[0]
    if score > 0:
        poslist.append((row['country'],row['city'], row['createdat'], row['hashtagkeys'], sentiment(row["tweetnostopwo
                        sentiment(row['tweetnostopwords']).assessments, row['source'], row['retweets'], row['userstat
        sentimentScore = score + sentimentScore
    else:
        netralList.append((row['country'], row['city'], row['createdat'], row['hashtagkeys'], sentiment(row["tweetnost
                           sentiment(row['tweetnostopwords']).assessments, row['source'], row['retweets'], row['userst

postiveTweetScores = pandas.DataFrame.from_records(poslist, columns=labels)
postiveTweetScores
```

	Country	City	Create_At	Hashtags Keyword	Sentiment Score	Postive	Assessments	Source	Retweets	User Status	sentiment score
0	unknown	unknown	2021-05-07	Binance	(0.116666666667, 0.133333333333)	True	[([social], 0.0333333333333, 0.0666666666667, ...	Twitter for iPhone	4	617	0.116667
1	unknown	unknown	2021-05-07	covid19,economics	(0.1, 0.6)	True	[([key], 0.0, 1.0, None), ([economic], 0.2, 0....	Twitter for iPad	12	15526	0.100000
2	unknown	Dubai	2021-05-07	COVID19	(0.103703703704, 0.477777777778)	True	[([tough], -0.388888888889, 0.833333333333, No...	Twitter for iPhone	151	2770	0.103704
3	Canada	Toronto	2021-05-07	Toronto,COVID19	(0.136363636364, 0.5)	True	[([live], 0.136363636364, 0.5, None)]	Twitter for Android	7	2182	0.136364

13. Neutral Tweets and concatenating and saving all dataframes into one CSV

Neutral Tweets and concatenating and saving all 3 dataframes into one csv

```
In [231]: neutralTweetScores = pandas.DataFrame.from_records(nutralList, columns=labels)
          neutralTweetScores
```

Out[231]:

	Country	City	Create_At	Hashtags Keyword	Sentiment Score	Postive	Assessments	Source	Retweets	User Status	sentiment score
0	Canada	Toronto	2021-05-07	unknown	(0.166666666667, 0.533333333333)	True	[([fair], 0.7, 0.9, None), ([anger], -0.7, 0.2...	Twitter for Android	0	1540	0.166667
1	unknown	unknown	2021-05-07	COVID19	(0.466666666667, 0.716666666667)	True	[([easily], 0.433333333333, 0.833333333333, No...	Twitter for iPhone	398	64373	0.466667
2	unknown	unknown	2021-05-07	unknown	(0.3, 0.2)	True	[([fun], 0.3, 0.2, None)]	Twitter for Android	2	74	0.300000
3	unknown	unknown	2021-05-07	unknown	(0.0833333333333, 0.55)	False	[([fresh], 0.3, 0.5, None), ([active], -0.1333...	Twitter for Android	18	242872	0.083333
4	unknown	Leeds	2021-05-07	COVID19	(0.333333333333, 0.5)	True	[([pregnant], 0.333333333333, 0.5, None)]	Twitter for Android	291	215	0.333333

14. By using Pandas extract all data in the form of list and get the analysis of the overall data by giving the summary of the defined output records.

```
In [254]: df = pandas.concat([negativeTweetScores, postiveTweetScores, neutralTweetScores])
          csv_name = '/project_folder/allTweetSentimentsfinal.csv'
          df.to_csv(csv_name, index = False, header=True, encoding='utf-8-sig')
```

```
In [258]: df = pandas.read_csv(csv_name)
          final_dict = {}
          for i, group in df.groupby(['Country']):
              # print(group)
              # break
              p_grp = group[group['Postive'].apply(lambda x: x == True)]
              n_grp = group[group['Postive'].apply(lambda x: x == False)]

              if i not in final_dict:
                  final_dict[i] = {}
              pos = len(p_grp)
              neg = len(n_grp)
              pos_key = 'positive_count'
              if pos_key not in final_dict[i]:
                  final_dict[i][pos_key] = pos
              neg_key = 'negative_count'
              if neg_key not in final_dict[i]:
                  final_dict[i][neg_key] = neg

              sources = list(set(group.Source))
              num_sources = len(sources)
              s_key = 'sources'
              n_s_key = 'num_sources'
              if s_key not in final_dict[i]:
                  final_dict[i][s_key] = sources
              if n_s_key not in final_dict[i]:
                  final_dict[i][n_s_key] = num_sources

              p_hashtags = list(set(group['Hashtags Keyword']))
              p_all_hashtags = []
              for hashtag in p_hashtags:
                  for x in hashtag.split(','):
                      if x.strip() != 'unknown':
                          p_all_hashtags.append(x.strip())
              p_hashtag_key = 'hashtags'
              if p_hashtag_key not in final_dict[i]:
                  final_dict[i][p_hashtag_key] = list(set(p_all_hashtags))
              n_p_hashtag_key = 'num_hashtags'
              if n_p_hashtag_key not in final_dict[i]:
                  final_dict[i][n_p_hashtag_key] = len(set(p_all_hashtags))

              clean_words = group.Assessments.apply(
                  lambda x: re.findall(r"\[([A-Za-z0-9_]+)\]", x))
              all_clean_words = [y for x in clean_words for y in x]
              # all_clean_words = []
              # for x in clean_words:
              #     for y in x:
              #         all_clean_words.append(y)
              # print(all_clean_words)

              s_a_key = 'sentiment_assessment_words'
              if s_a_key not in final_dict[i]:
                  final_dict[i][s_a_key] = all_clean_words
```

RESULT EVALUATION:

Tabular format of results

```
In [260]: pandas.DataFrame.from_dict(final_dict).T
```

Out[260]:

	hashtags	negative_count	num_hashtags	num_sources	positive_count	sentiment_assessment_words	sources
Argentina	[economics, covid19]	0	2	1	1	[key, economic]	[Twitter Web App]
Australia	[COVID19]	0	1	2	2	[older, easily, love]	[Twitter for iPhone, Twitter for Android]
Canada	[Toronto, Ontario, COVID19]	2	3	3	3	[live, live, fair, anger, many]	[Twitter for iPhone, Twitter Web App, Twitter ...
France	[COVID19, Massachusetts, Covid19]	1	3	2	1	[last, huge, higher]	[#COVID19, Twitter for Android]
Georgia	[COVID19]	0	1	1	2	[easily, love, easily, love]	[Twitter for iPad]
Germany	[USA, Corona]	1	2	1	0	[active]	[corona-tracker-app]
Greece	[coronavirus, Greece]	0	2	1	1	[new, confirmed]	[Twitter for Android]
India	[COVID19, Covid19]	3	2	2	4	[beloved, personally, positive, positive, posi...	[SocialPilot.co, Twitter for Android]
Ireland	[COVID19]	0	1	3	3	[incredibly, new, spanish, economic, new, old]	[Twitter for iPad, Twitter Web App, Twitter fo...
Japan	[Binance, COVID19, JAPAN]	0	3	2	2	[social, economic, many]	[Twitter for iPhone, TweetDeck]
Nepal	[]	1	0	1	0	[angry]	[Twitter for iPhone]
New Zealand	[]	0	0	1	2	[older, older]	[Twitter Web App]
Nigeria	[COVID19]	0	1	1	2	[new, new]	[Twitter for Android]
Peru	[IndiaFightsCorona, COVID19]	1	2	1	0	[]	[Twitter for Android]
South Africa	[COVID19]	0	1	1	2	[total, confirmed, total, total, confirmed, to...	[Twitter for Android]
South Korea	[COVID19]	1	1	1	0	[special, hard]	[Twitter for Android]
Sri Lanka	[]	1	0	1	0	[young, social]	[SocialPilot.co]
Switzerland	[economics, covid19]	0	2	1	1	[key, economic]	[Buffer]
United Kingdom	[]	1	0	1	0	[last]	[Twitter for iPad]
United States	[COVID19]	0	1	2	2	[easily, love, easily, love]	[Twitter for iPhone, Twitter for Android]
unknown	[Polio, Lockdown, tunisie, COVID19, Alberta, s...	77	33	19	179	[frustrated, fourth, frustrated, fourth, cruci...	[Naattuvartha, Twitter for iPad, ContentStudio...

Fig 1: This is the generated output of the table where we can actually compare the outline of the tweets.

Based on the country, I have extracted all the hashtags keywords, number of negative tweets counts, number of hashtags keywords count, number of sources from where people are tweeting, number of positive counts, sentiment analysis assessments based on the judgement of words from the tweets, and finally type of source from which people are tweeting from.

VISUALIZATION DIAGRAM WITH DETAILED EXPLANATION:

Visualization of data by using Tableau

1. Count of Sources, the view is filtered on Country, which excludes unknown

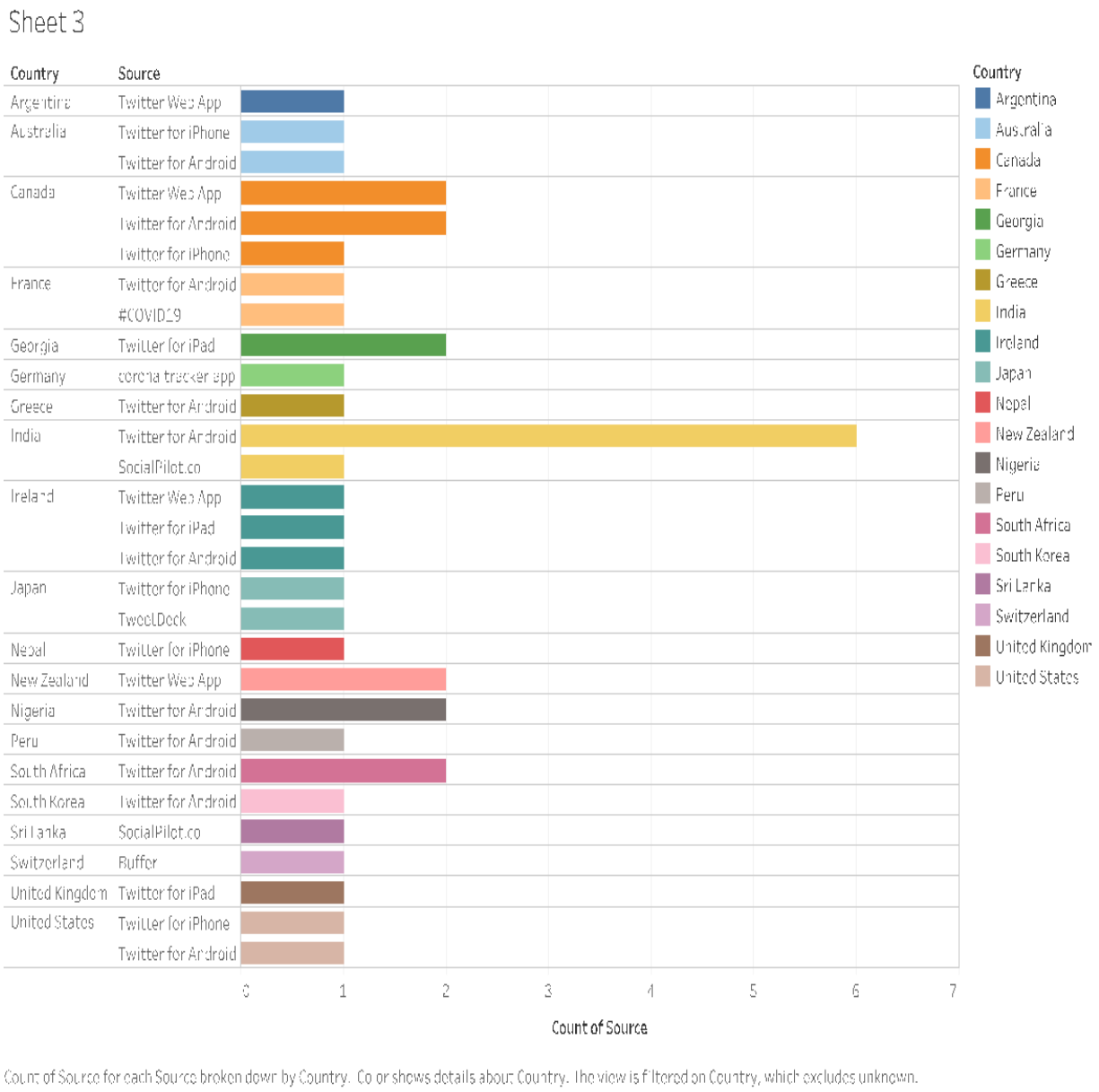


Fig 2: Graphs based on source of tweets based on countries

In the above Fig 2, we can see the number of sources as per the country have been shown in the graph. Where India - using Twitter for Android being the highest source and Switzerland been the lowest source count

2. WordCloud of Hashtags Keywords Tweeted by People

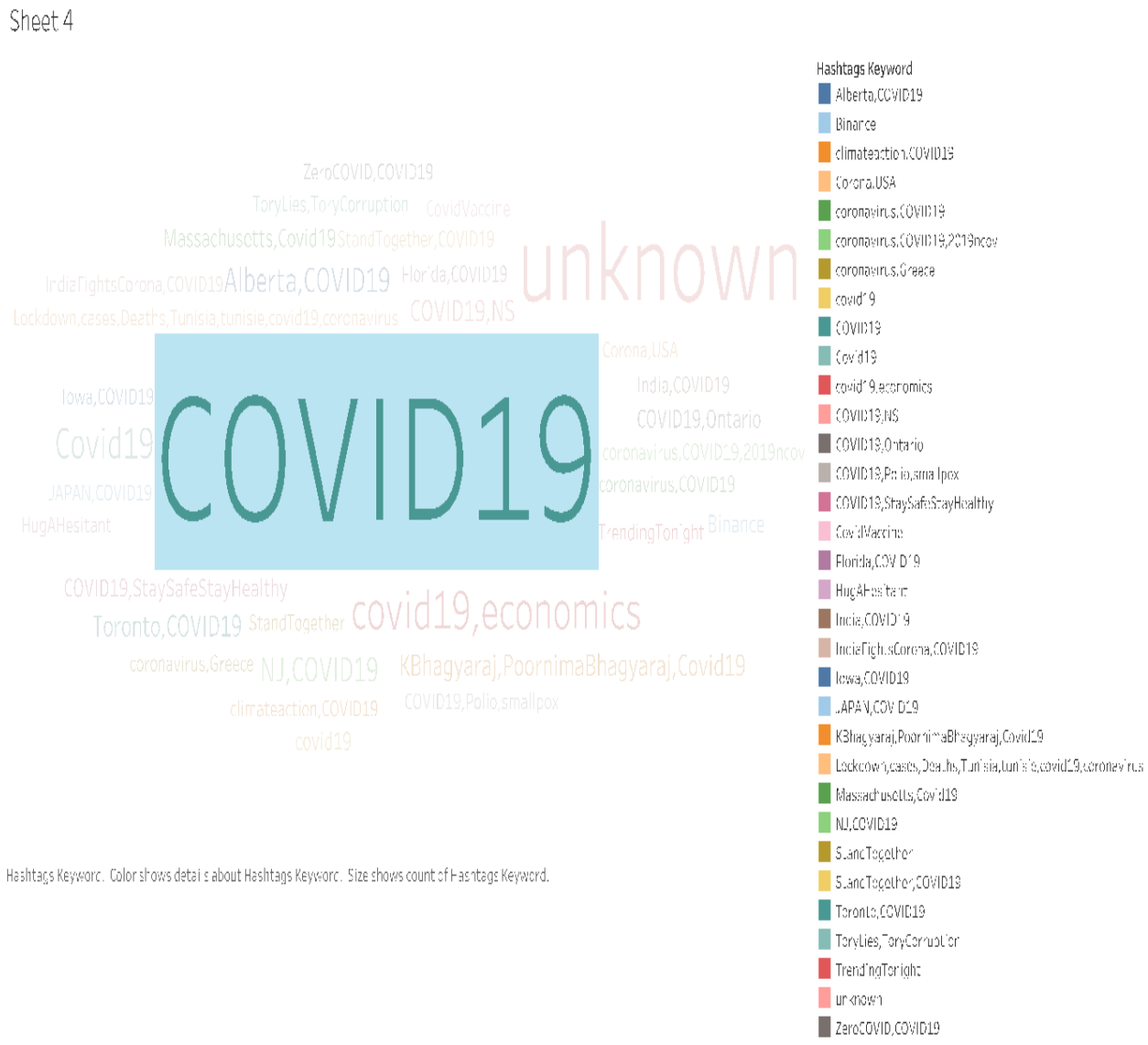


Fig 3: Hashtags keywords WordCloud

The above Fig 3 , depicts all the hashtags keywords used by different continents of the world. This demonstrates that COVID19 (all letters capital has the highest , most common hashtag key) used all over the world.

3. Total value of Retweets based on Country

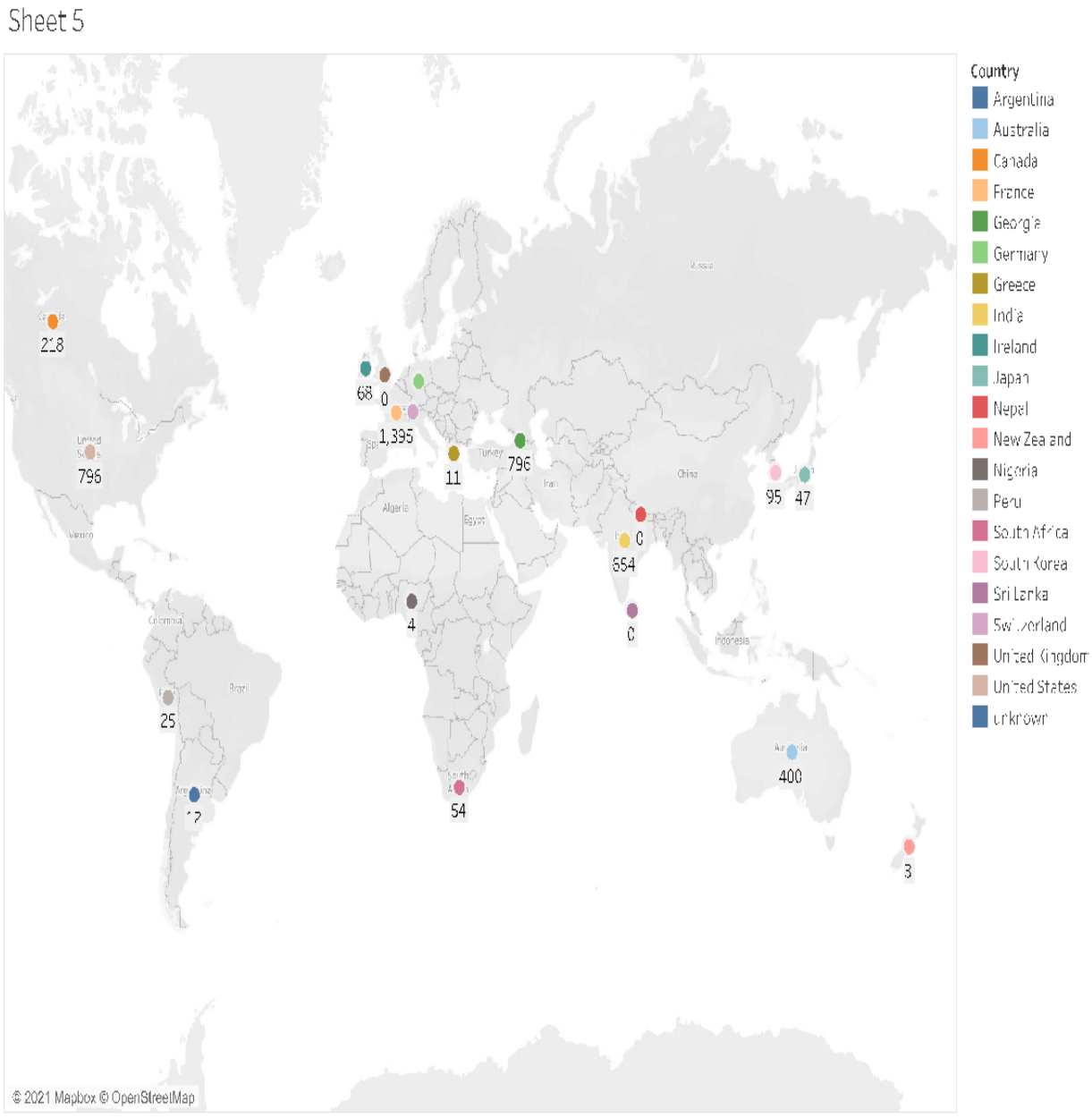


Fig 4: The marked Labels are sum of Retweets based on Country

The above Fig 4, tells us that from all over the world these many countries have max retweets , out of which France being highest - 1395 and New Zealand being lowest - 3.

4. Which cities have the maximum number of positive and which cities have the maximum number of negative sentiments.

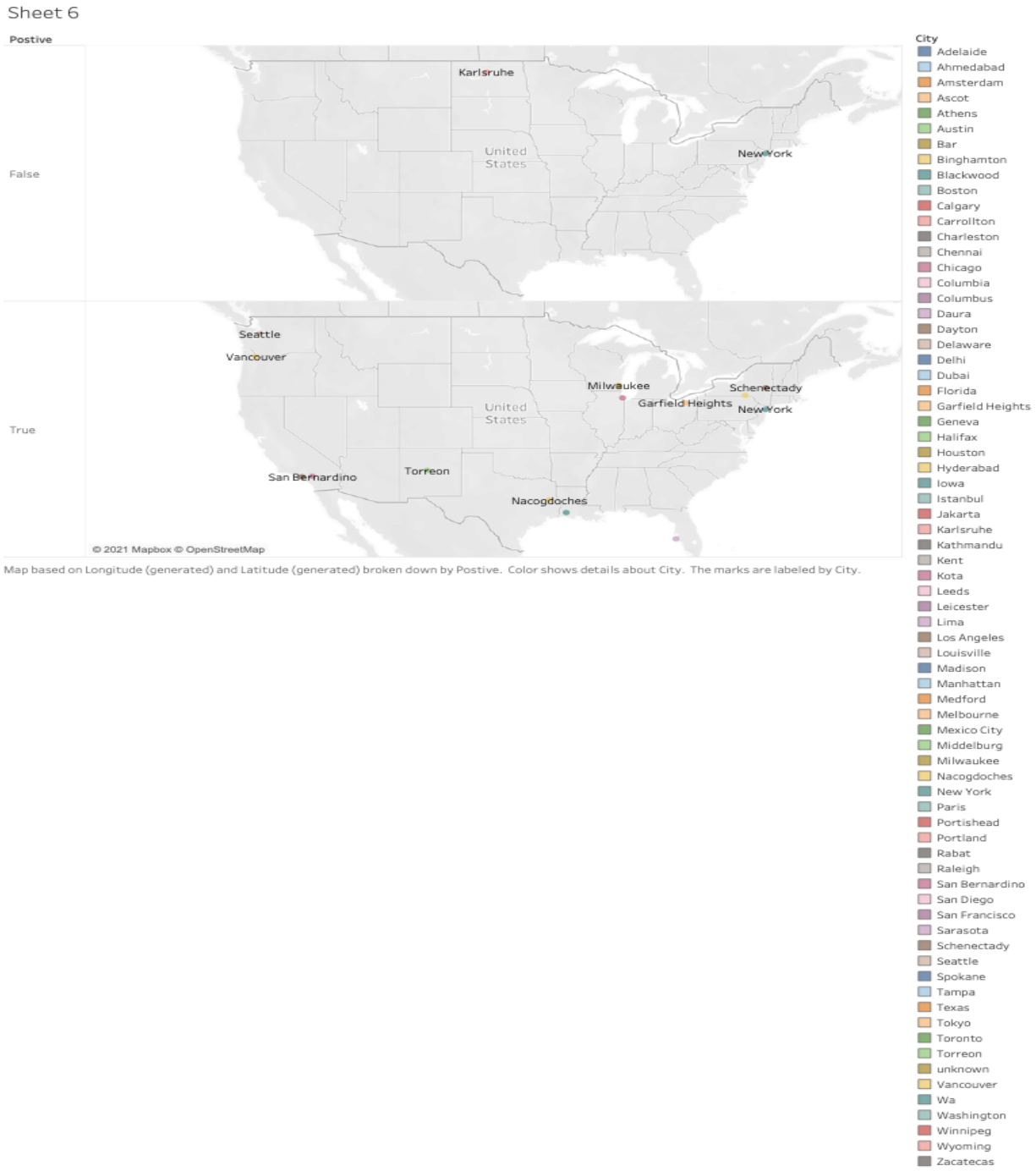


Fig 5: Positive and Negative Sentiments based on Cities

Fig 5, depicts the Covid19 sentiment analysis based on positive and negative score. Maximum cities have tweeted positive tweets and hardly 1 or 2 have tweeted negative tweets viz new york.

5. Count of Retweets for each city

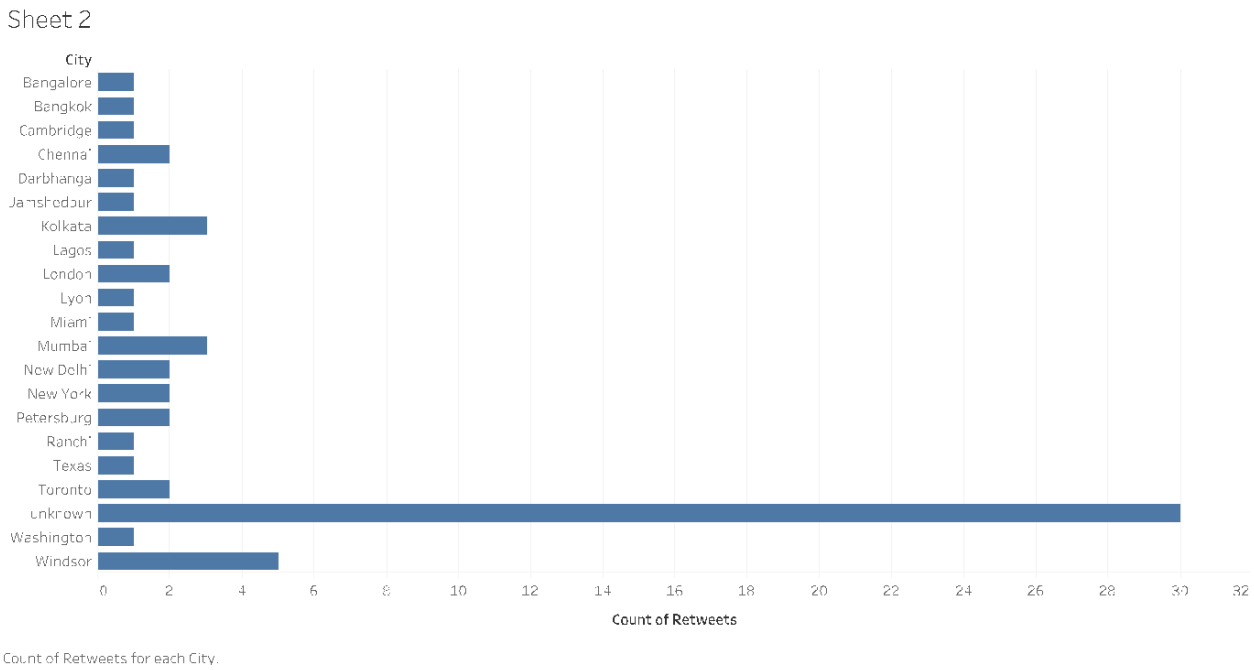


Fig 6 : Count of Retweets based on each city.

Here we can see that there are many tweets from unknown parameter where city is not defined as 30K,Windsor being highest 5K.

6. Group selected country and count the values of hashtags

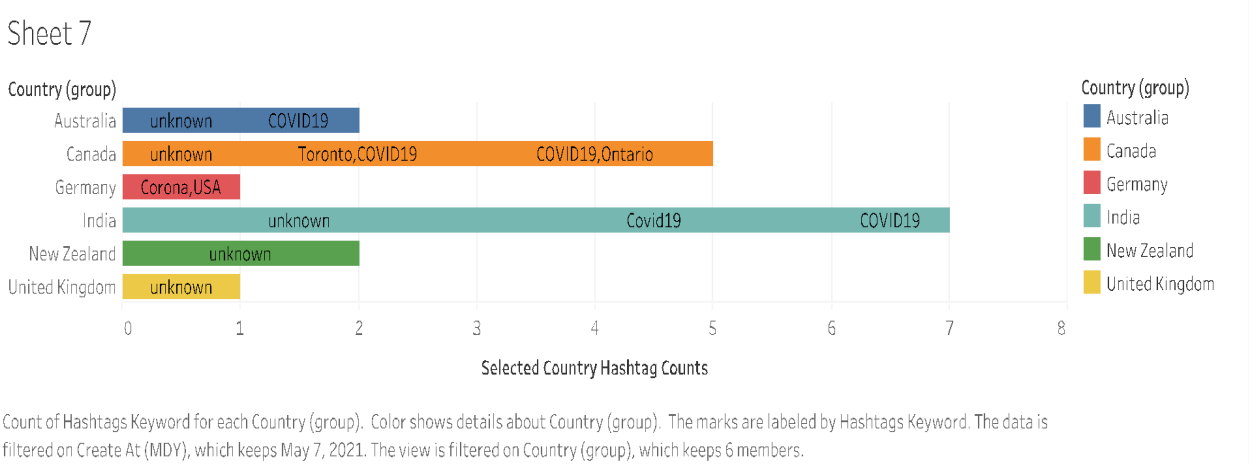
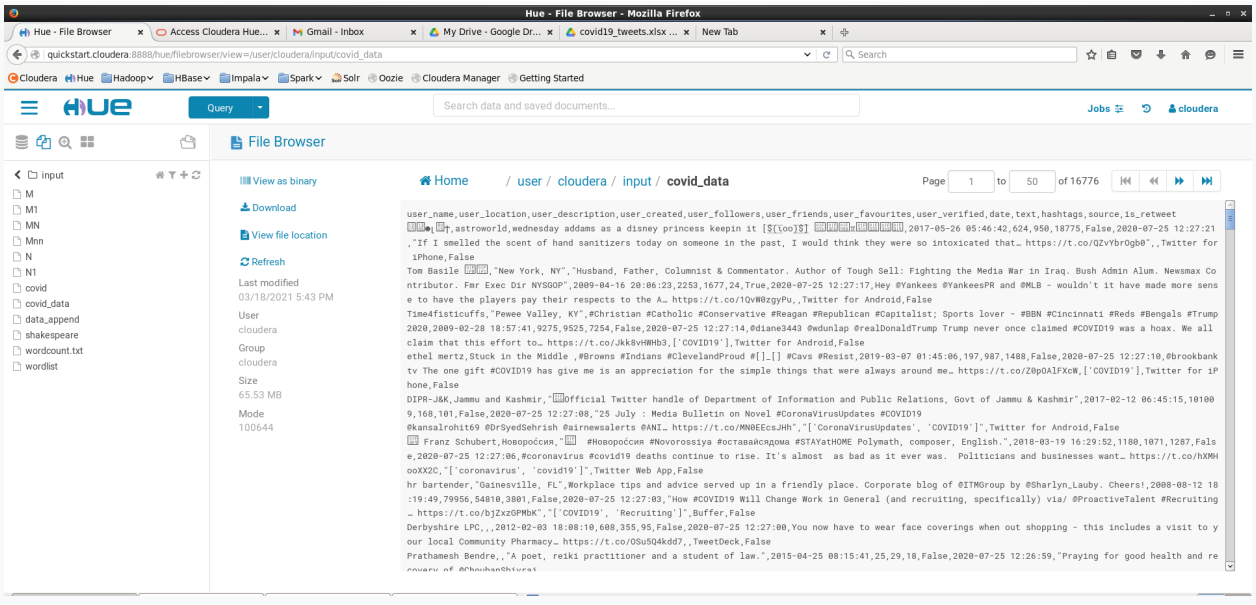


Fig 7: This figure depicts selected country hashtags count. India being highest and Germany being lowest count.

Hue:

Apache hue is used for visualization and querying databases. It is an open source which helps look HDFS file system and manage databases like hive, etc. Where you can visualize your queries. We have added our database to the HDFS and visualize it using hue.



Hive:

Hive is a data warehouse built on top of Hadoop for structured data. It has SQL-like query and hence is very popular and simple to operate. We have used it to analyze our dataset to find trends in it. We run these queries in hive and visualize it using tableau.

Pseudo-Code:

1. Starting of Hive in the terminal and creating a table named 'covid19_tweets' along with all its attributes and their data types. The command specifies that each variable is seperated by ',' as it is a csv file and thats how the software should differentiate between different variables and their data.
2. The csv file for covid 19 is loaded into the table by specifying the file path on the device.

3. Running the queries and getting the results.

The queries operated on this dataset are:

To see from where tweets are coming from majorly.

```
hive> select user_location,count(user_location) as coun from covid19_tweets  GROUP BY user_location order by coun des
c limit 100;
Query ID = cloudera_20210322105656_4c0f4fd3-febf-4203-bd17-805562e2b3e2
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
```

```
       36772
India   3741
United States   2455
New Delhi; India        1721
Mumbai; India   1401
Washington; DC  1354
London; England 1269
United Kingdom  1206
London  1103
New York; NY    1082
Australia       964
Worldwide       879
Canada  835
New Delhi       822
USA     817
UK      803
WORLDWIDE       777
Los Angeles; CA 776
South Africa    766
Lagos; Nigeria  731
California; USA 709
Global  688
Nairobi; Kenya  618
Nigeria 579
Atlanta; GA     578
Chicago; IL     561
Switzerland     551
Mumbai  523
Earth   516
Johannesburg; South Africa      493
Boston; MA      489
Chennai; India  481
Ireland 470
Toronto; Ontario        468
Singapore       466
Hyderabad; India        451
Texas; USA      444
```

To see from which source are tweets majorly being generated from.

```
hive> select source,count(source) as coun from covid19_tweets  GROUP BY source order by coun desc limit 100;
Query ID = cloudera_20210322110404_fa00df44-45f8-4513-b4b7-7ad3ad150002
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1612400960231_0073, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1612400960231
_0073/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1612400960231_0073
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-03-22 11:05:09,181 Stage-1 map = 0%,  reduce = 0%
```

```
OK
Twitter Web App 56891
Twitter for Android     40179
Twitter for iPhone      35472
TweetDeck       8543
Hootsuite Inc.  7321
Twitter for iPad        4336
Buffer  2728
Sprout Social   1833
Instagram       1759
IFTTT   1545
dlvr.it 681
GlobalPandemic.NET      679
LinkedIn        628
COVID19-Updates 625
Twitter Media Studio    596
WordPress.com   537
HubSpot 513
Sprinklr        497
FS Poster       414
Dynamic Signal  391
Resistbot Open Letters  390
Cheap Bots; Done Quick! 335
Paper.li        322
Twitter for Advertisers 317
Blood Donors India      281
IAMBLOG2TWITTER 266
Alexander Higgins       225
Zoho Social     224
The Social Jukebox      216
Fabrik.fm       209
SocialPilot.co  208
coronaData_Test 208
HN_Comments     191
Tweetbot for iOS        181
Twitter Media Studio - LiveCut  175
Orlo    168
Oktopost        163
ContentStudio.io        162
SocialFlow      162
Twitter for Mac 162
Social Media Publisher App      159
Salesforce - Social Studio      153
Khoros  149
LaterMedia      148
Global Citizen Mobile App       136
TweetCaster for Android 133
AgoraPulse Manager      130
```

To see from which users have the greatest number of tweets.

```
hive> select user_name,count(user_name) as coun from covid19_tweets   GROUP BY user_name order by coun desc limit 100
;
Query ID = cloudera_20210322105252_935b0b5a-4b22-4db8-a69d-af7a6aad40ae
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1612400960231_0069, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1612400960231
_0069/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1612400960231_0069
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-03-22 10:52:35,508 Stage-1 map = 0%,  reduce = 0%
2021-03-22 10:52:42,872 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 2.0 sec
2021-03-22 10:52:51,236 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 4.23 sec
MapReduce Total cumulative CPU time: 4 seconds 230 msec
```

```
GlobalPandemic.NET      679
Coronavirus Updates     625
covidnews.ch    402
Open Letters    390
Blood Donors India      282
Hindustan Times 280
IAM Platform    266
Paperbirds_Coronavirus  259
IANS Tweets     244
ANI     233
Coronavirus Updates - Alexander Higgins 225
Dushyant Vachhani       220
OTV     220
ABS-CBN News    218
Covid Data      208
#IndiaFightsCorona      199
The New Indian Express  197
COVID Scale     191
APO Group English       180
Sambad English  177
Coronavirus_Scotland    150
ABS-CBN News Channel    145
National Cyber Security 143
Hotpage News    142
ActivistBowen   140
Novel Coronavirus - Covid19     137
Deccan Herald   136
Integral Overview       133
LatestLY        132
The Pioneer     132
Business Standard       129
NewsMobile      128
Conservative in Utah    126
Countries Without Coronavirus Bot       126
Isolation Bot   126
Global Times    124
Covid-19 Bot    124
8min 46sec      123
Kathryn Guylay; MBA; PhD        123
moneycontrol    121
China Xinhua News       116
Suke    116
NEWS9   115
The Times Of India      111
VoiceToData     109
```

To see which users, have the greatest number of followers.

```
hive> select user_name, max(user_followers) as maxim from covid19_tweets  group by user_name order by maxim desc limit 100;
Query ID = cloudera_20210322115858_dce09d64-5f14-4f37-93a7-d3a017e4f9a4
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1612400960231_0082, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1612400960231_0082/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1612400960231_0082
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2021-03-22 11:58:10,808 Stage-1 map = 0%,  reduce = 0%
```

```
OK
CNN     49442559
National Geographic     24359165
CGTN    13892841
NDTV    13568396
The Times Of India      13057020
United Nations  12806543
China Xinhua News       12681192
CNN International       11081219
WIRED   10380131
ABP News        10363089
Ivanka Trump    9222125
UNICEF  8363171
World Health Organization (WHO) 8207835
Hindustan Times 7730317
Shashi Tharoor  7663695
Anonymous       7379859
People's Daily; China   7114528
ABS-CBN News    7062663
Shivraj Singh Chouhan   6740445
Bloomberg       6582855
The Hindu       6368904
SkyNews 6322221
Department of Defense       6190147
Hamid Mir       5926839
GMA News        5785454
El Universal    5723883
IndiaToday      5334662
American Red Cross      5290712
Zee News English        5248572
Filmfare        5109282
Mumbai Police   5085882
ABS-CBN News Channel    5016797
VANITY FAIR     4855471
ANI     4721276
@zoomtv 4671626
Suresh Prabhu   4515088
CNNNews18       4511223
Ravi Shankar Prasad     4507530
China Daily     4398378
Cricbuzz        4288833
POLITICO        4248153
Vasundhara Raje 4192882
State Bank of India     4132904
Dr. Mehmet Oz   4036082
World Economic Forum    3846197
New Scientist   3736401
Economic Times  3692878
```

Spark

RDD:

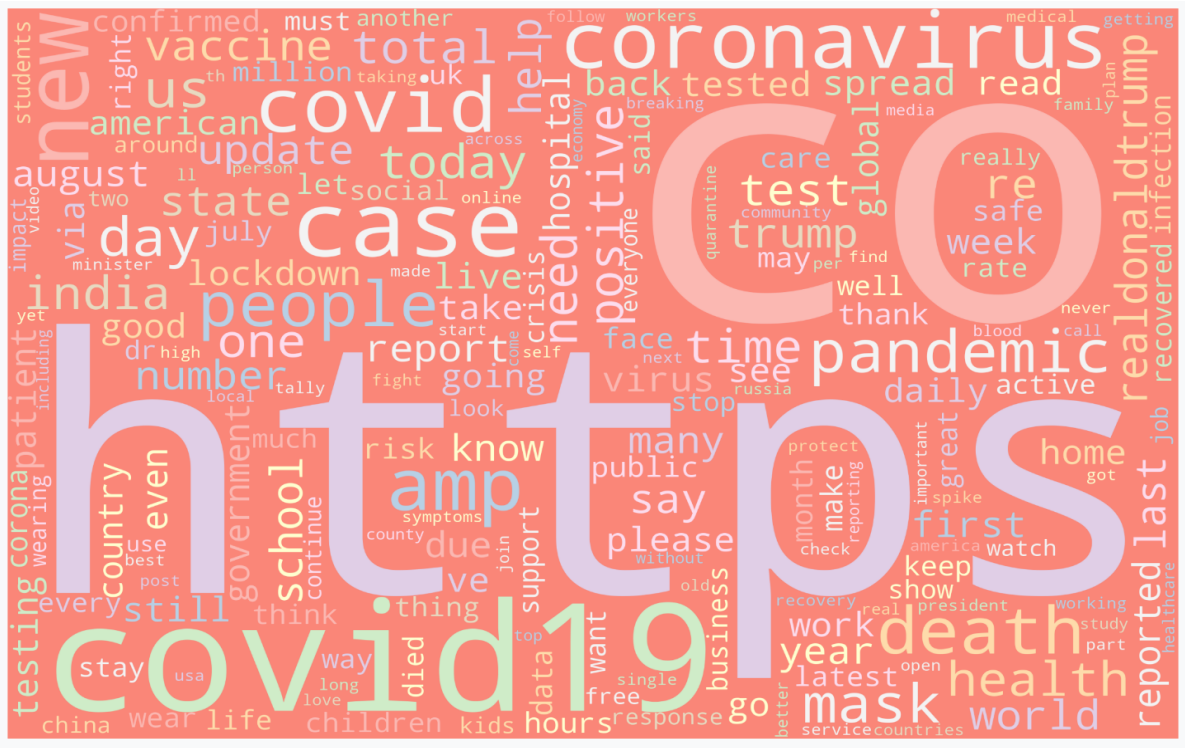
The code was executed on Google Colab using Apache Spark version 3.0.2. The dataset file and all the intermediate files created were stored in Google Drive. Word Count was done using the help of Apache Spark, and python library ‘pyspark’. Other important python libraries used were ‘nltk’ for

removing punctuations, ’re’ for regular expression, and ‘matplotlib’, ‘wordcloud’ for plotting the Word-cloud.

Pseudo-Code:

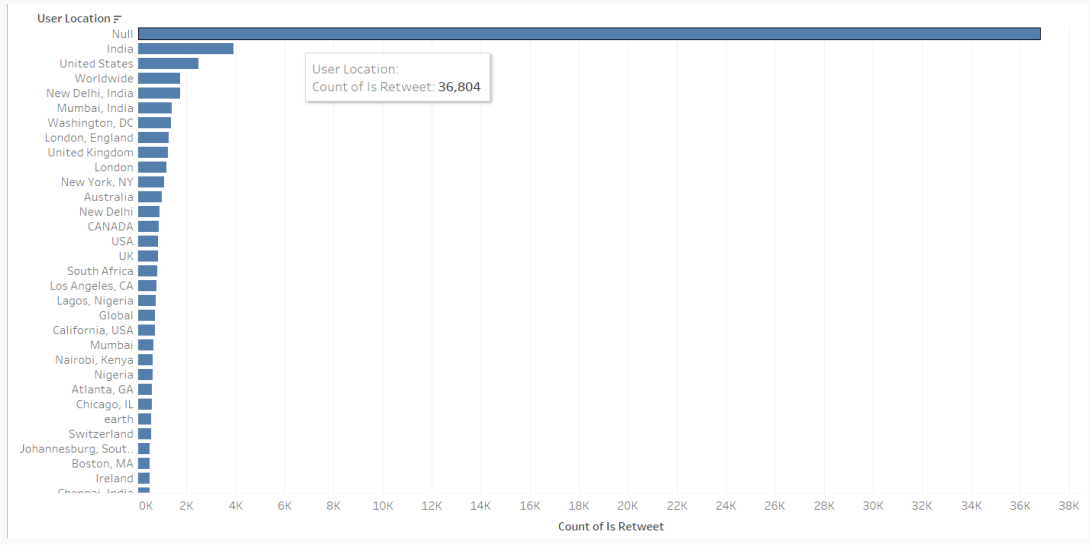
- The dataset is imported from the google drive and all the punctuations are removed and the output is stored in an intermediate file.
- Spark is loaded in system and the MapReduce function is performed on the intermediate file.
- Before performing Word Count on the data all the stop words are removed using *‘splitRDD_no_stop = words.filter(lambda x: x.lower() not in stop_words)’* command.
- The Word Count is then performed on the data which returns words as key and their count as value.
- The words are then sorted on basis of their count in the descending order.
- The word and count of the top 200 words are stored in the ‘out_task1’ file.
- The word cloud is made for the top 200 words using the ‘out_task1’ file.

Preliminary Results



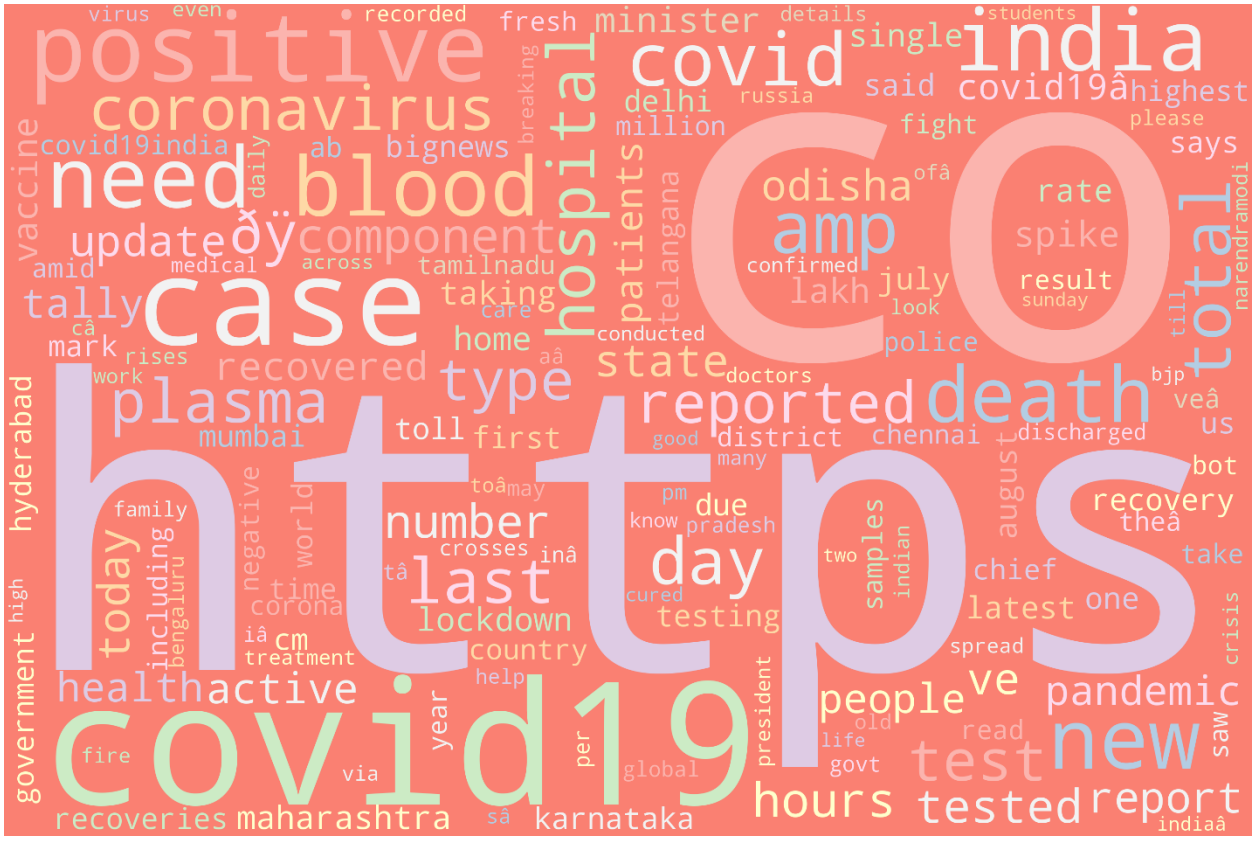
Word Count for all tweets

We have generated Word Cloud from our text dataset from which we can view that letter https occurs most of the time and then CO after that covid19 and coronavirus, list goes on.



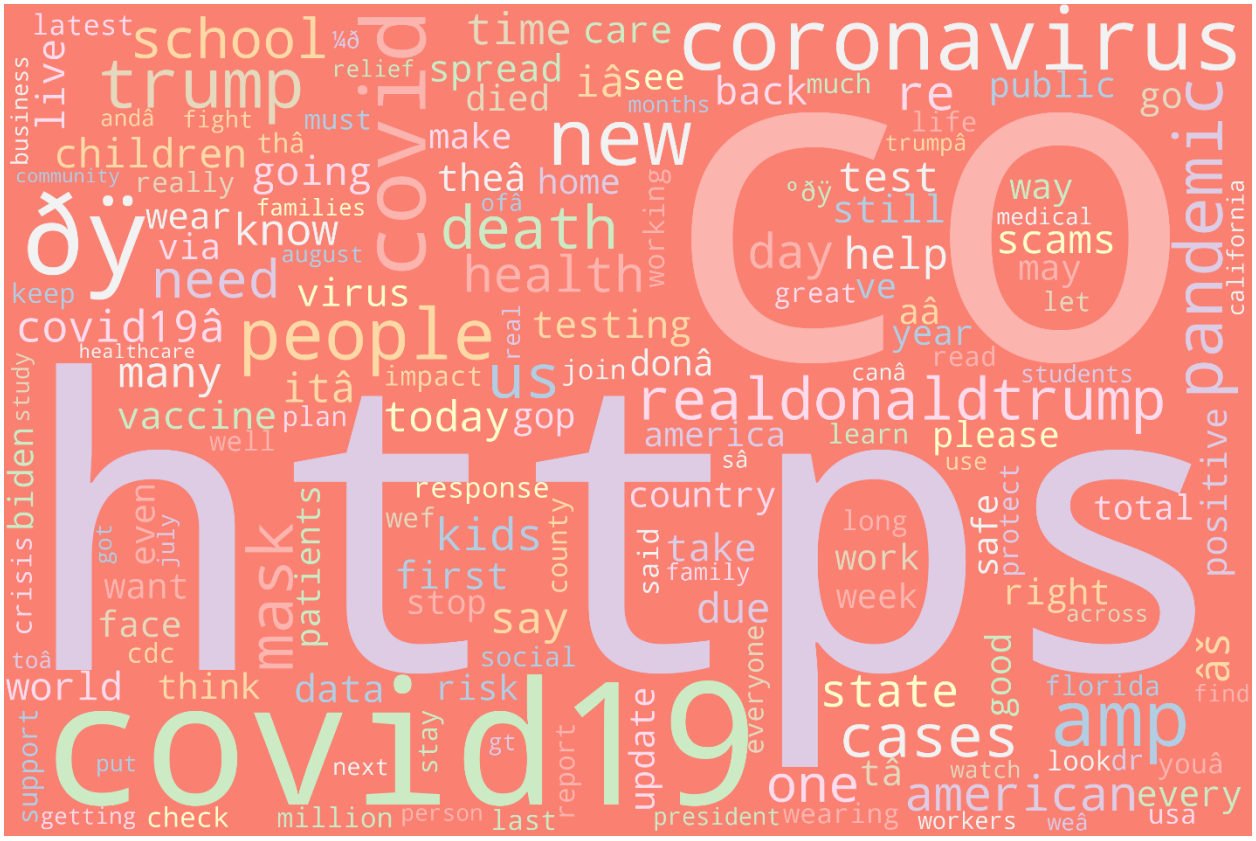
To see from which locations tweets are coming from (Viz for hive queries).

There are maximum number of tweets from India followed by United States. The reason behind this maybe due to the population and popularity of the social media application twitter among that location.



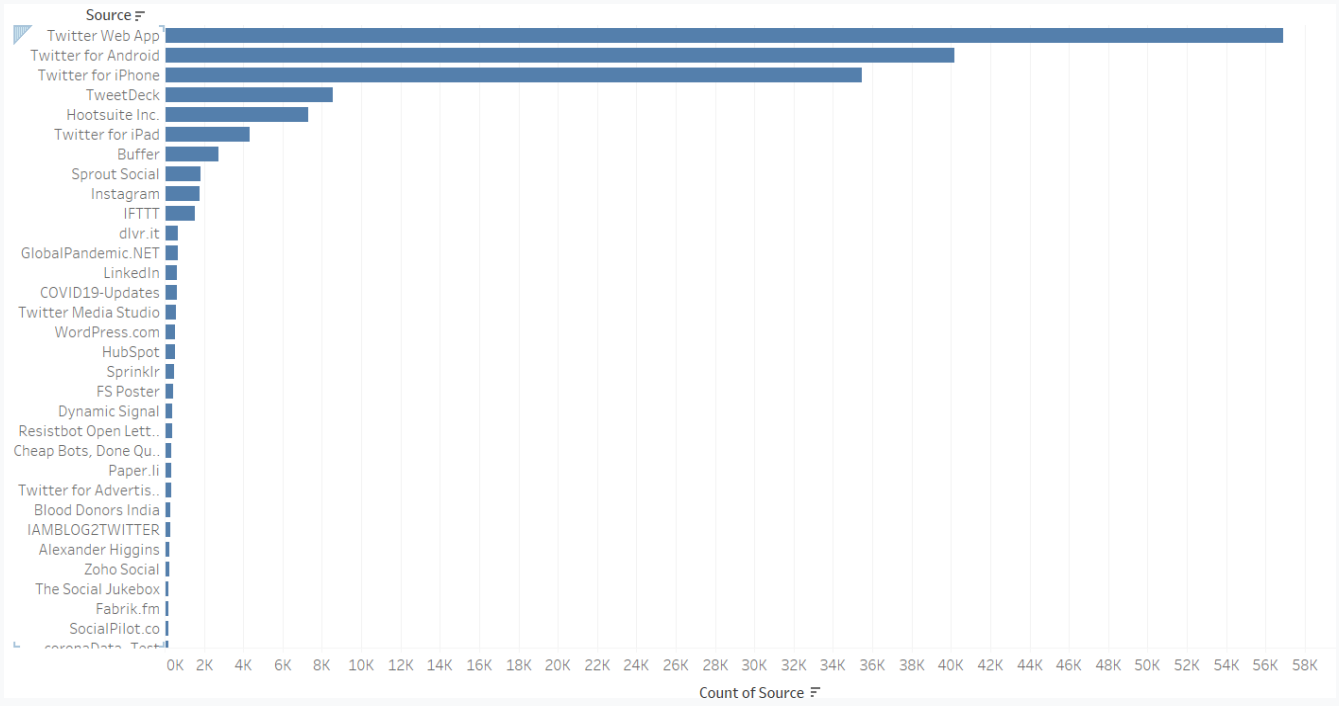
Word Cloud for tweets originating from India

The tweets originating from India have words with states name arising pre-dominantly reporting on the cases in each state. Also, words like hospital, cases, death are pre-dominantly occurring which highlight the conditions in the country. These are terms which indirectly are highlighting how people are feeling in the country. These negative terms have been surrounding people's life. Also, there are terms like government which were used in the tweets to talk about what the rules and regulations are and how they have been successful or unsuccessful in curbing the problem.



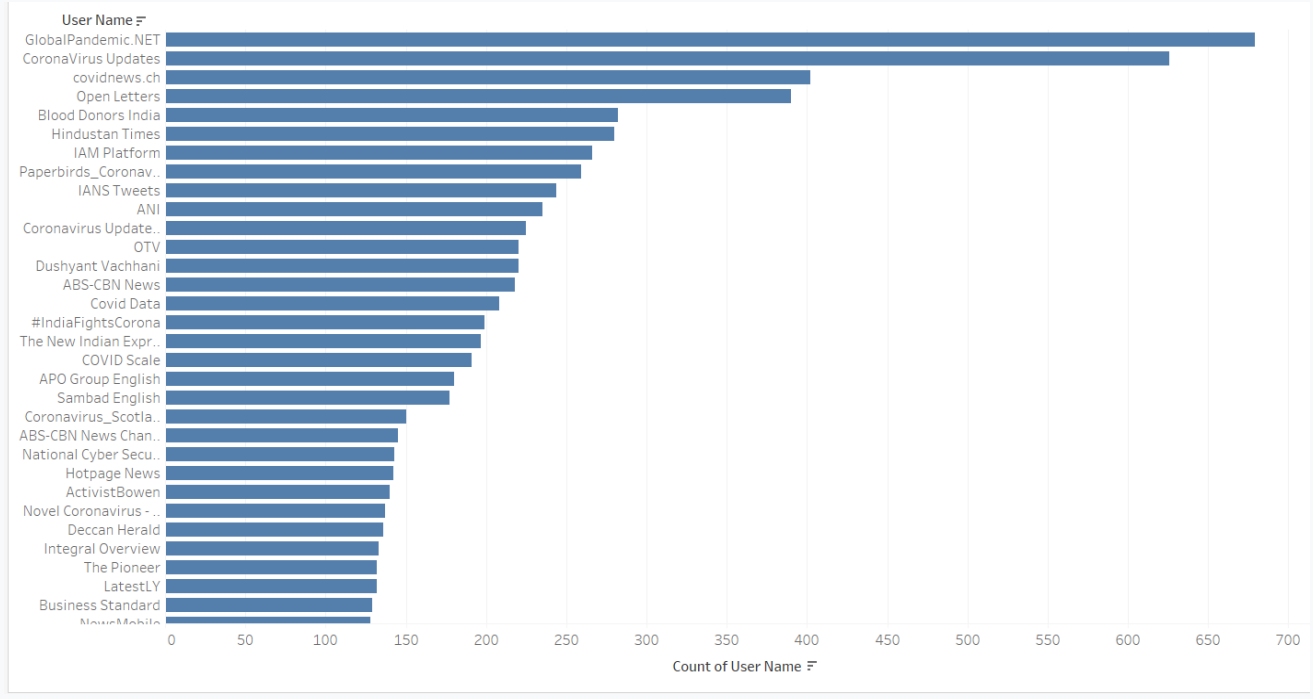
Word Cloud for tweets originating from USA

The tweets originating from USA have terms like Biden and Trump showing people tweeting about the election. Also, the term like business is there which shows how the market is going on. There are also, covid related terms like mask, pandemic, cases to show how people are being affected by it and how they are handling it. Also, the term vaccine is present which shows people first expecting for vaccines to be made and later tweeting about how they felt about talking it.



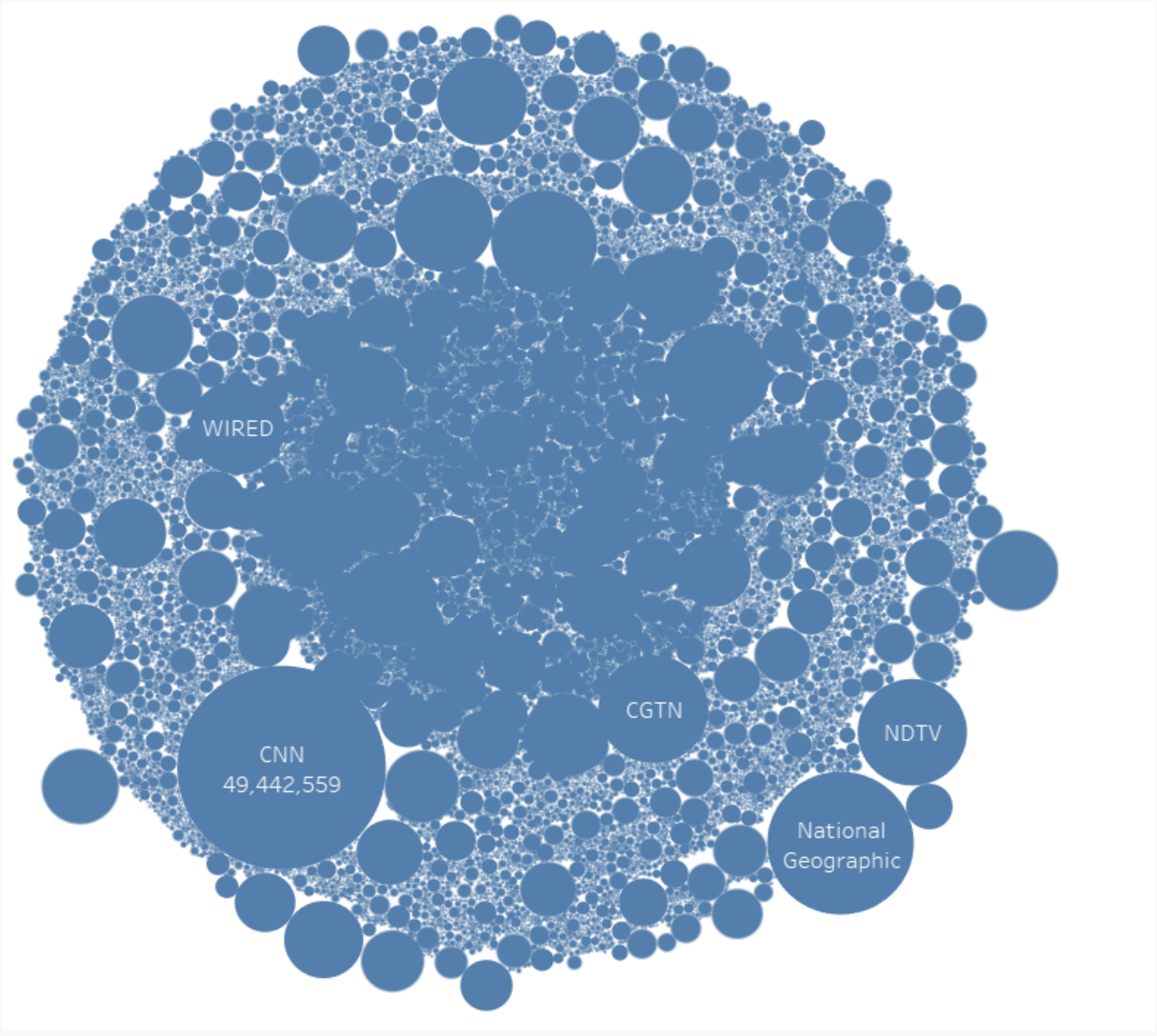
To see from which source are tweets majorly being generated from (Viz for hive queries)

Maximum number of tweets have been from web application followed by android and twitter users. This maybe because a lot of people use their laptop or computers to tweet. Also, there are more android users than iPhone and hence we see this pattern.



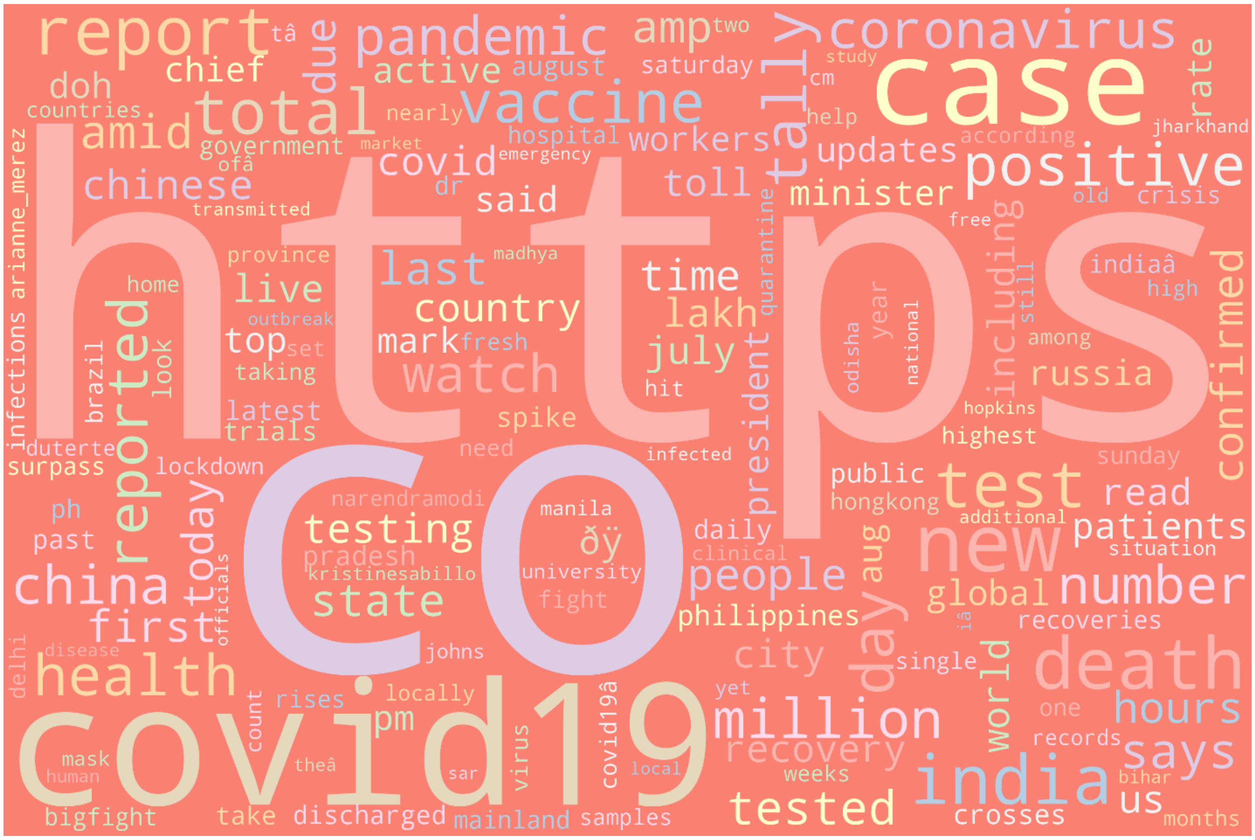
To see from which users have the greatest number of tweets (Viz for hive queries)

GlobalPandemic.NET has greatest number of tweets on this pandemic followed by CoronaVirus Updates and covidnews.ch. These users continuously give updates on what is going on around the globe with respect to the current pandemic. How people are reacting and how the pandemic is being administered.



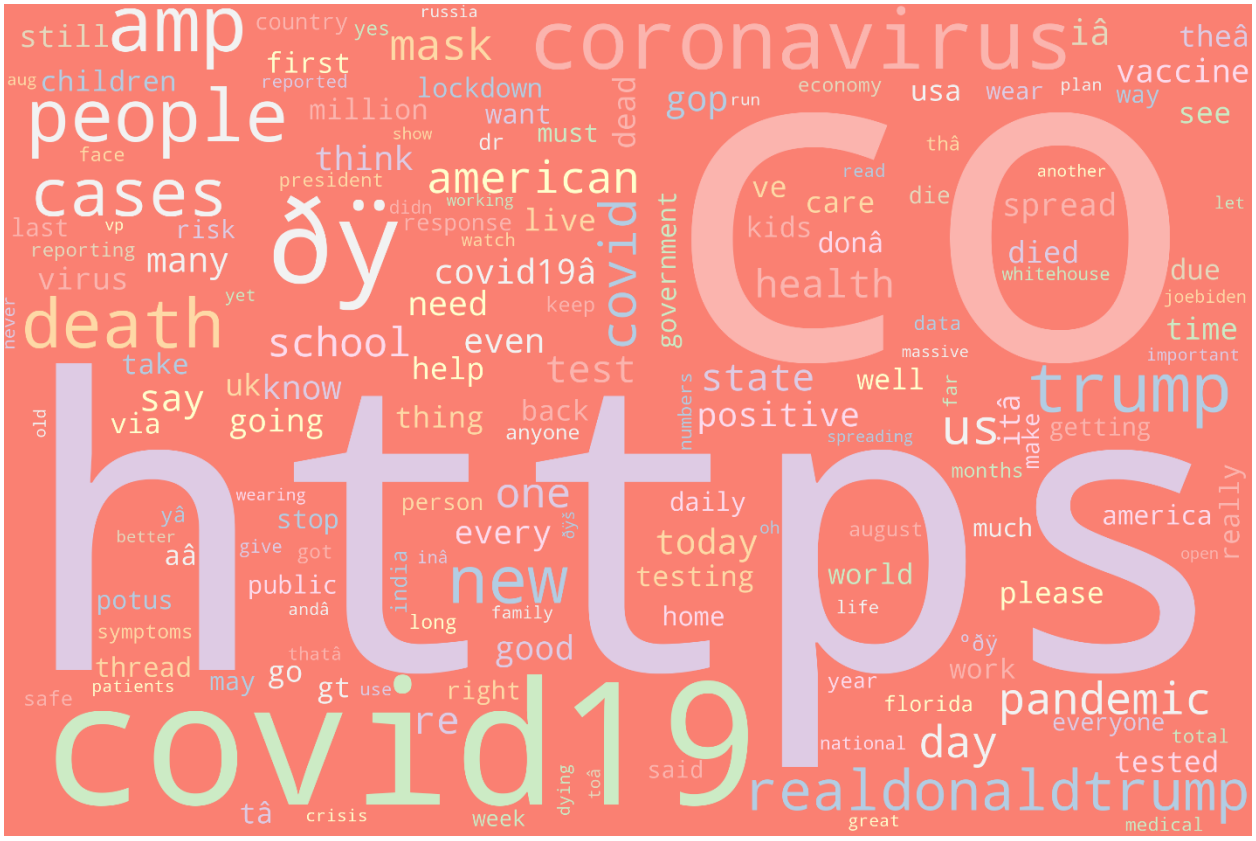
To see which users, have the greatest number of followers (Viz for hive queries)

These are the channels or people with maximum number of followers who have been reporting or tweeting on the covid pandemic. The following shows that these are prominent users who people have faith in on reporting the scenarios correctly.



Word Cloud showing tweets from users having more than 7M followers

The word cloud shows terms like country name where the tweeter handle is tweeting on the situations in various country, pre-dominant one being China and India. They are also talking about the cases that are being reported, how many of them are recovering and how many are not. The term vaccine also comes up, tracking of the progress of it across the globe and talking about its effectiveness in its purpose.



Word Cloud of tweets that are most popular

The word cloud shows tweets with a lot of negative terms like death, crisis, lockdown, and many other terms showing what the people are pre-dominantly feeling. Also, there is name Donald Trump who was the last US president and used to tweet a lot on this pandemic. He has been popular in people's tweet because of the election and also because of his ways on handling the spread of the virus in US. Other terms are related to covid like mask, patients which again are negative. There is also vaccine occurring again showing people to be interested in talking about it and closely monitoring its manufacturing and its effectiveness.

CHAPTER 1(LIFE)

WHO: Covid 19 pandemic has affected everyone around the world in one or the other way.

1. Cities, with their dense international setups were particularly affected which includes markets, business, travel, tourism. These were the common entry points for the virus.
2. Rural areas and regions with high numbers of elderly people were affected as well.
3. The pandemic has deeply impacted jobs and businesses. In the developing world,micro-level, small-level and medium-level enterprises are under intense pressure.

WHAT:

1. The pandemic has taken a lot of lives.
2. COVID-19 has rapidly affected businesses, world trade, our day to day life and movements.
3. Presently the impacts of COVID-19 in daily life are extensive and have put tremendous pressure on the healthcare sector and affected the social lives of the people.

WHEN: The early traces of the virus were found in November 2019, but it spread rapidly across the world and was declared as a pandemic in March 2020 by the World Health Organization and is still going on.

WHERE: 200+ Countries and Territories around the world have reported a total of 113M+cases of the coronavirus COVID-19 that originated from Wuhan, China, and a death toll of2.5M+ (as of this writing)

WHY: There is no definite answer to this question. The primary cause of this virus has been linked to the animal “bat” by the mainstream media.

CHAPTER 2(DATA)

WHO: The dataset covers all the twitter users who have used hashtag Covid19 or used any covid or related material to it in their text. It covers people from every corner across the globe.

WHAT: The dataset records all the basic information of the user's social media handle, which in this case is twitter and along with that it covers the twitter text that the user is sending which is one of the major analysis points in our dataset.

WHEN: The collection of datasets began from November 2019 when Covid 19 started spreading from china to everywhere across the globe. The dataset is collected from twitter.

WHERE: It was a local topic trending in China, but as it started spreading across the world, people started tweeting about it and that is how the dataset was collected.

WHY: The dataset was collected to get an understanding of how people are reacting to this pandemic and how they are being affected by it.

CHAPTER 3(THE SCIENTIST AND AI)

WHO: The main character in this dataset are the people across the globe who are using tweets to express their feeling? The data scientist that is us, are trying to understand these feelings and report on it.

WHAT: Big data tools like Hadoop and Spark were used in the analysis of the dataset. Various modules in Hadoop like map-reduce, hue, hive and Cassandra were used to analyze the data. Also, various spark modules were employed to analyze the data. Tableau was used as one of the visualizing tools.

WHEN: The project has been in work for past three months where different tools were used to get perspective of the situation. For better analysis different tools were used so that all the angles were effectively covered, and better analysis can be presented.

WHERE: The project was part of the coursework Programming in Hadoop and Spark.

CHAPTER 4(USERS)

WHO: The main target audience are both the people who are interested in these kind of analysis and programmers or data scientist who are interested in looking into how tools were used to get various answers.

WHAT: The tools and visualization gave us the understanding on how people are feeling about this pandemic and situation going around.

WHEN: The project is available for anyone to be viewed on GitHub.

WHY: The tools can help other data scientist to understand how certain tools can be used to get some inferences. The visualization in turn can help people see what are people tweeting about and how they are feeling.

CHAPTER 5(THE SOCIETY)

WHO: People around the globe are affected. All the users who use the twitter account and have tweeted on topic of covid are being sampled. Twitter users are majorly from USA and India and hence were oversampled in the database while the rest of the countries were not much actively covered because of the population it had and the twitter users. The data scientist who did this analyzing were Vyoma Desai, Affan Charolia and Ali Alyamni.

WHAT: The analysis has brought some known but disturbing facts about the sentiments of the people, about how they are feeling. The data is from a social media application, that is the twitter. On twitter people share their thought and opinions in form of tweets which are not hidden information and are available for public to see. Hence there is no issue of data privacy or security.

WHEN: The impact is going on right now as the pandemic is not over yet and people are still being affected by it. There is no issue of data privacy.

WHERE: The pandemic has spread across the globe and is still going on. It is claiming many lives and have affected many in more than one form.

WHY: The issue at hand is not seeming to slow down and is affecting a lot of people. It has affected a lot of people in more than one form.

HOW: The project can highlight the sentiments of the people and in turn highlight where the issues are lying. This can help authorities to go to people in need and help them in which ever way possible.

CONCLUSION

We have used various Hadoop, Cassandra, and Spark modules to try and understand the twitter dataset and try to analyze it. We have seen which of the

words are frequently used in the twitter texts. Also, we tried to query the tweets based on various factors like location, etc. Finally, we used hive queries and tableau to visualize our dataset and understand more about it. People are using a lot of negative terms in their tweets and they cannot be blamed for that as it is a difficult situation for everyone. They are talking about loss of lives and their livelihood.

FUTURE WORK

Further advance to this project can be use of certain deep learning tools to analyze the text further, to better understand what the tweets texts in the twitter dataset is talking about. These can help better classify the problem in more than two terms of positive and negative. It can be used to analyze whether people are angry or sad or if they are happy about something or feeling joy.

REFERENCES

[1] L. -A. Cotfas, C. Delcea, I. Roxin, C. Ioanăş, D. S. Gherai and F. Tajariol, "The Longest Month: Analyzing COVID-19 Vaccination Opinions Dynamics From Tweets in the Month Following the First Vaccine Announcement," in *IEEE Access*, vol. 9, pp. 33203-33223, 2021, doi: 10.1109/ACCESS.2021.3059821.

[2] R. F. Sear et al., "Quantifying COVID-19 Content in the Online Health Opinion War Using Machine Learning," in IEEE Access, vol. 8, pp. 91886-91893, 2020, doi: 10.1109/ACCESS.2020.2993967.

[3] N. Paul and S. S. Gokhale, "Analysis and Classification of Vaccine Dialogue in the Coronavirus Era," *2020 IEEE International Conference on Big Data (Big Data)*, Atlanta, GA, USA, 2020, pp. 3220-3227, doi: 10.1109/BigData50022.2020.9377888.

[4] https://docs.docker.com/engine/install/

[5] https://www.datastax.com/dev/academy

[6] W. Feng, P. Gu, C. Zhang and K. Zhou, "Transforming UML Class Diagram into Cassandra Data Model with Annotations," *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)*, 2015, pp. 798-805, doi: 10.1109/SmartCity.2015.165.

[7] A. Jaison, N. Kavitha and P. S. Janardhanan, "Docker for optimization of Cassandra NoSQL deployments on node limited clusters," *2016 International Conference on Emerging Technological Trends (ICETT)*, 2016, pp. 1-6, doi: 10.1109/ICETT.2016.7873643.

[8] D. K. Zala and A. Gandhi, "A Twitter Based Opinion Mining to Perform Analysis Geographically," *2019 3rd International Conference on Trends in*

Electronics and Informatics (ICOEI), 2019, pp. 59-63, doi: 10.1109/ICOEI.2019.8862548.

CONTRIBUTION OF WORK

Vyoma Desai (individual contribution: Pages 34 - 50)
Significance, Feature's part, chapter life. Extracting live data from Twitter API, CAssandra, Spark, PySpark, Datastax integration, Data preprocessing, sentence tokenization, removing stopwords, extracting positive and negative sentiments and visualization of graphs in Tableau.

Affan Asad Charolia (individual contribution: Pages 51 - 64)
Motivation, background, and objective's part. Implementing queries using Hue and Hive and representing graphs in Tableau. Doing word count is spark and visualizing it using word cloud. Chapter 3 and 4.

Ali Alyami (individual contribution: Pages 6 - 33)

Project dataset analysis, Extract Data from twitter API related to covid-19 tweets, HIVE sentiment analysis, Conclusion. Implementing dataset wordcount using Hadoop Map Reduce, Implementing Dataframes in python to analyze geographical visualization of tweets, creating spark DataFrames and perform Spark SQL queries using pyspark.