Part B – Context Based Sentiment Analysis

Team Details 1B

Student	Email	Github ID	Class ID
Attaluri, Lalith Chandra	la4kf@mail.umkc.edu	LalithChandraAttaluri	4
Karumanchi, Pranitha Saroj	pkt59@mail.umkc.edu	pranithakarumanchi99	7

Goals & Objectives:

Motivation:

In the data-driven competitive environment, in different sectors, such as Telecomsector, Bankingsector, Financial & Health service sectors, data handling had became vital in decision taking process. The key factors will be to handle the humongous amounts of data and get benefits from it. One of the impressive systems, use Apache Spark, which can manage big data on real time & perform various analyses.

Objectives:

The main idea of our project is to use Spark Streaming to do ETL process & to implement machine-learning concepts on that data in real time. Our device source is the Twitter data and we will be using the Streaming Content, which is the real time data processing, and using streaming API we'd be collecting data for a collection of specified keywords in a near real time process. Then we'd do transformations on RDD's streaming set & load data into Hive framework which is similar to ETL method. We will also be executing EDA on the twitter data while capturing data background. Our project will also highlight the Sentiment Analysis where the tweets are populated with real-time sentiments. It also describes the keyword reasons for categorizing a tweet into positive & negative sentiment.

Significance:

For the sentiment analysis, we use an predefined ML (TextBlob) method to predict the sentiment of tweets, & based on this, we will analyze keywords that are critical to sentiment prediction & we will train the new Machine learning model with this analysis for predicting tweet feeling that will improve prediction accuracy.

Features:

The characteristics of the project include collecting real time tweets using twitter streaming APIs, performing ETL (pre processing data, extracting required information & loading data into Hive), & use TextBlob to predict sentiment for every tweet and thereby train new Machine learning model with tweets as input and type of sentiment as output.

Description of Dataset:

We use twitter streaming API to capture tweets on a particular collection of keywords, almost in real time. Each & every tweet is in JSON format, which are pairs with key value. It had different details about tweet such as tweet text, user who tweeted, user information, geographical location where tweet was originated, tweet source such as Android, Iphone etc. Using Spark, we capture real-time data through streaming API The streaming background with a 20-second window and streaming API is able to download 500 kb of data from which we search keywords such as **Coronavirus, COVID-19, Pandemic, COVID19, covid19, covid-19, coronavirus, pandemic** and extract nearly 80 kb from that.

Design Flow:

We originally built a account in the Twitter Developer API. Using the Spark Streaming program, we downloaded the tweets from provided API tokens & credentials. Our project have a sort of client & server model where we collected tweets through our application's Tweepy PY library that served as server level. Now we used the program Spark Streaming to submit the request and, the application will receive the server tweets on a window based model upon completion.

If customer receives tweets on window based setup, then two operations will be performed. First, we'd show feeling on fly for any tweet that streamed on the stream info. Second, after carrying out the series of transformations on the streaming tweets, we will be storing tweets into the HIVE system. Having saved the data in HIVE, by training on stored tweets we would be building a classification model. If model was trained then we'd run model on tweets in real time & predict the feeling.

The next level of the project is to compare scores on model and direct sentiment scores on tweets. This also describes the keyword reasons for categorizing a tweet into either positive or negative sentiment.

Step by Step Implementation of the design:

Data Extraction:

We had created twitter developer account for downloading twitter info, and have used tweepy python library & spark streaming background to download tweets. Every tweet is a raw json sample, which also had complex json format, we pre processed data & converted complex json to simple text format that can be managed and sent by socket to the client side system.

We transformed them from the client side into a table structure where we could write hive queries for some research. Results of the study are briefly clarified in the preliminary report.

Server Client Implementation:

We adopted the client-serve approach to accessing the streaming twitter tweets. On the server, we used tweepy library to connect to the twitter streaming API with the appropriate credentials that we got from the twitter developer account. To pass data from server to client we used socket link. In browser, the data we received from streaming twitter.

The API is in the JSON format, which we have pre-processed on the server side, & sends simple text data to client side with appropriate fields. The tweepy will start to stream twitter data and put the data in socket.

On the client side we used the spark streaming context to collect the data from socket, configuring spark streaming context with a window of 20 units time and a sliding interval of 20 seconds which means that the next 20 windows of data will be read by the spark reader for every 20 seconds.

Data Pre-processing:

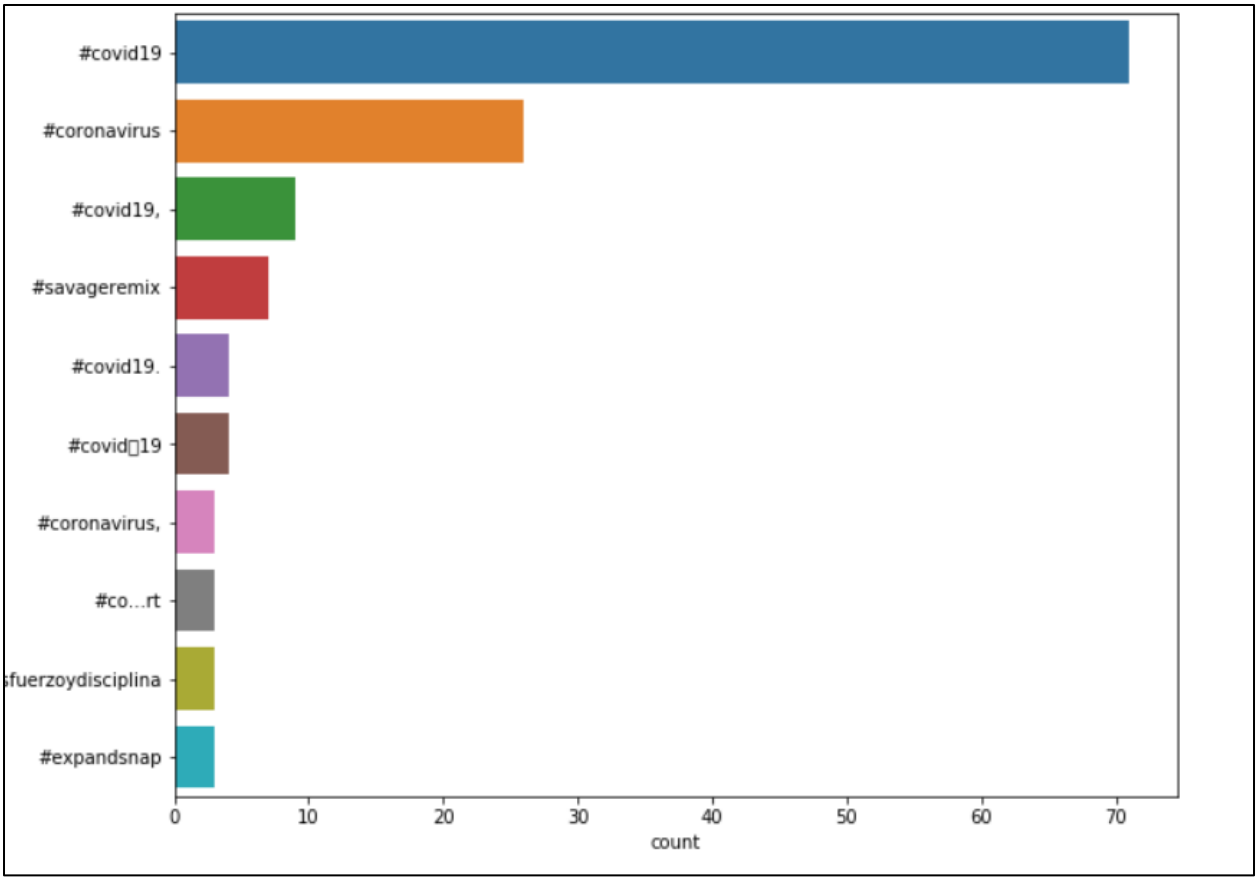
If the client observes the data, we apply map and lower it to turn the data into table form and store it in a hive database. It's easier to write the Spark SQL queries from the hive, and to do some data analysis.

Data Visualization:

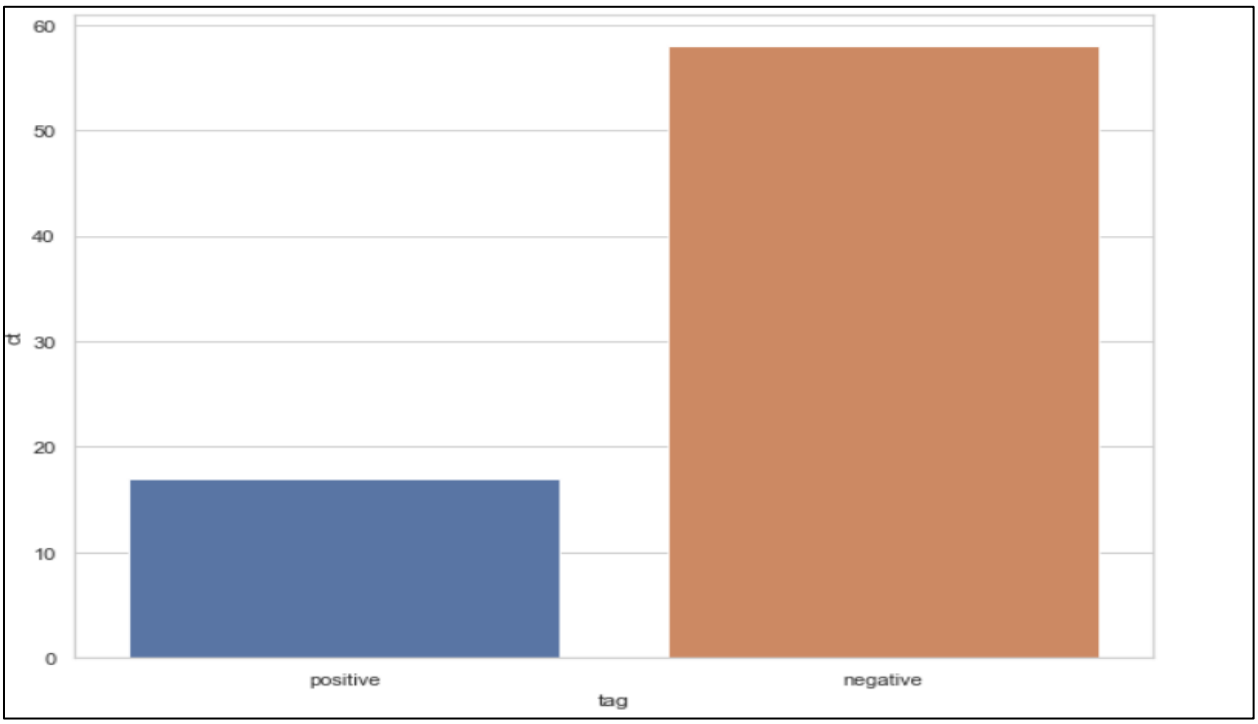
- **Most Widely used URL's:** The below figure shows the distinct URL's present in the time frame

```
+-------------------------------------------------+---+
|url                                              |ct |
+-------------------------------------------------+---+
|https://t.co/kq6pjltosert                        |1  |
|https://t.co/kzvxb8svpd                          |1  |
|https://t.co/awn5ffzyzyel                        |1  |
|https://t.co/toyotopm39rt                        |1  |
|https://t.co/oruil2ueidrt                        |1  |
|https://t.co/c9czei681vhttps://t.co/r9vwnanp4trt|1  |
|https://t.co/kwzortp3q1rt                        |1  |
|https://t.co/fcgg0o5oxbrt                        |1  |
|https://t.co/jy8jtyqftart                        |1  |
|https://t.co/dvuzqxewbr@maddow                   |1  |
+-------------------------------------------------+---+
```

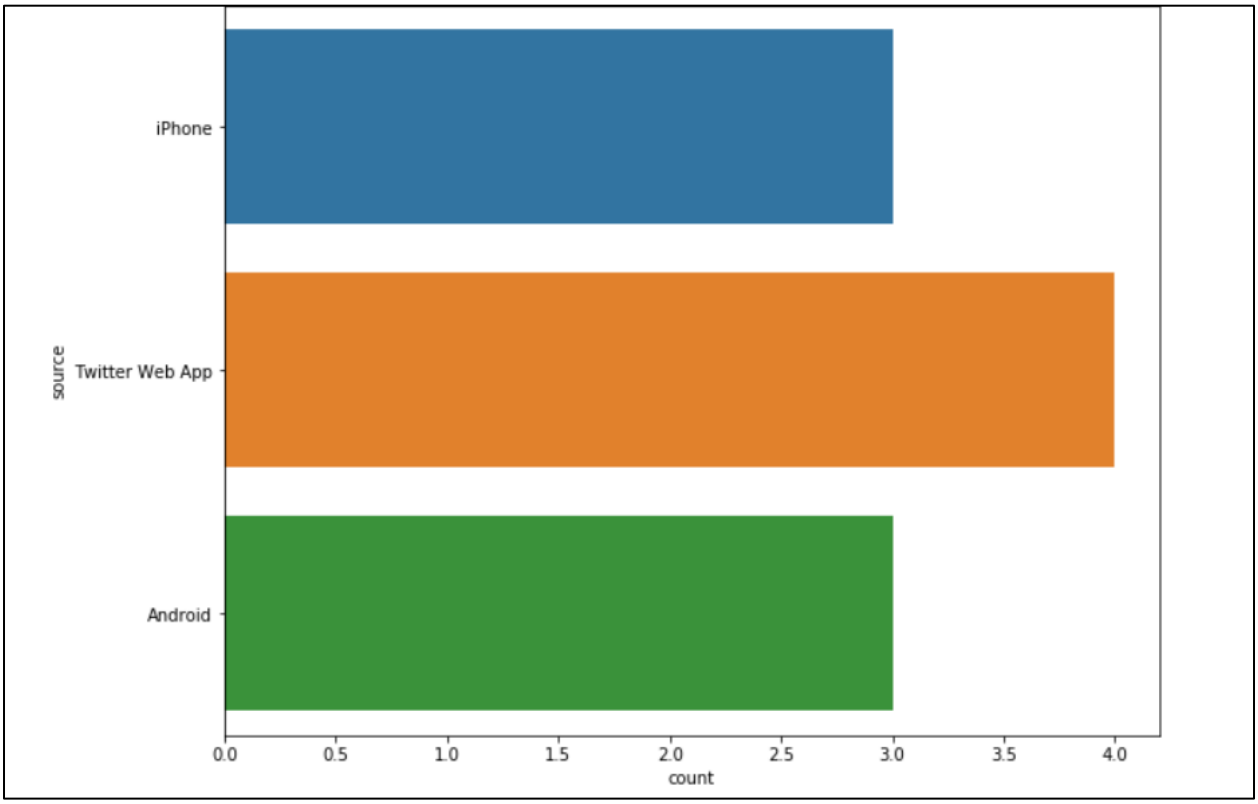
- **Top 10 hashtags:** The below figure shows the top 10 hashtags used in the particular timeframe



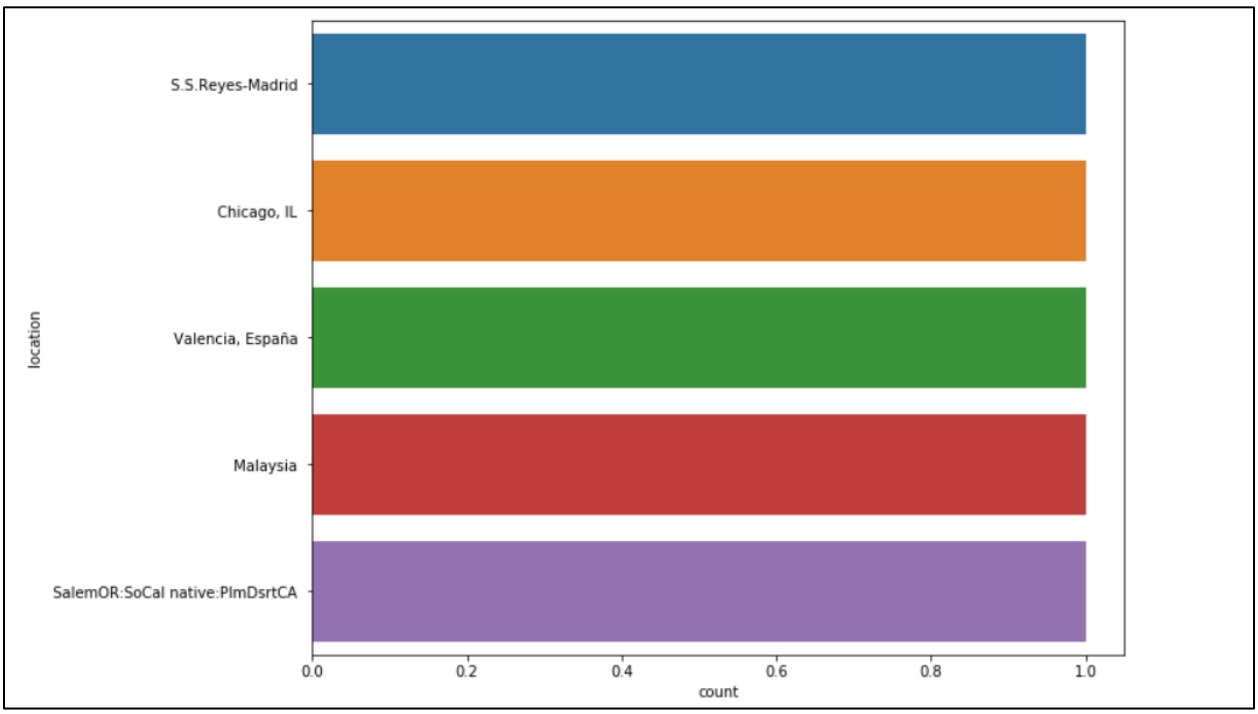
- **Positive vs Negative tweets:** As the tweets we have collected are based on Covid-19 pandemic negative tweets are higher than positive tweets.



- **Tweets from different sources:**

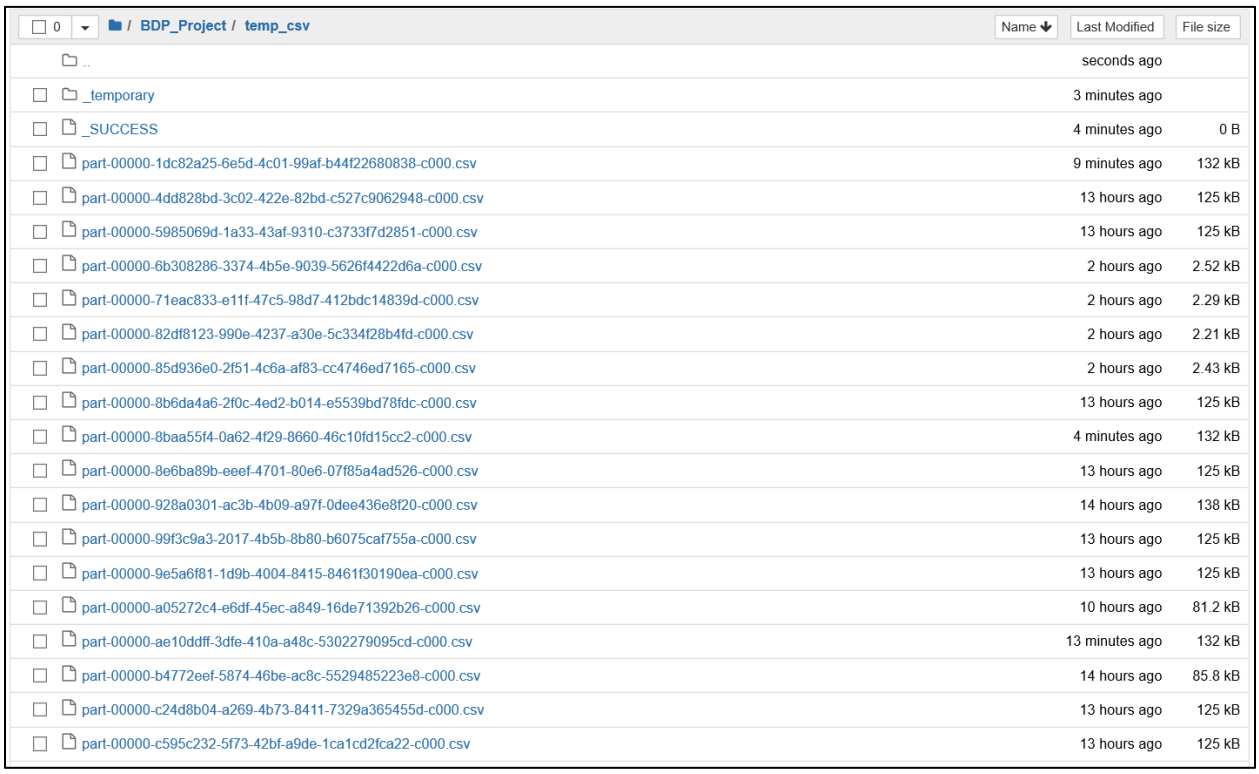


- **Tweets from different locations:**



Saving and loading the streamed data:

Using the Spark Streaming Background, we gather tweets for every 20 seconds and save them to the local folder with following options:- coalesce=1, which stores data in single file; headers are true, store file with headers. After we have collected the enough tweets to train the model, we merge all csv files through Globalize and load merged file to the Spark SQL sense, and create external hive tables. We've gathered about 50k tweets. We used TextBlob on the data for predicting feelings for each tweet and with the data we are training the ML model for predicting feelings about the meaning of sports. Below are the csv files that are saved.



0 / BDP_Project / temp_csv

Name	Last Modified	File size
..	seconds ago	
_temporary	3 minutes ago	
_SUCCESS	4 minutes ago	0 B
part-00000-1dc82a25-6e5d-4c01-99af-b44f22680838-c000.csv	9 minutes ago	132 kB
part-00000-4dd828bd-3c02-422e-82bd-c527c9062948-c000.csv	13 hours ago	125 kB
part-00000-5985069d-1a33-43af-9310-c3733f7d2851-c000.csv	13 hours ago	125 kB
part-00000-6b308286-3374-4b5e-9039-5626f4422d6a-c000.csv	2 hours ago	2.52 kB
part-00000-71eac833-e11f-47c5-98d7-412bdc14839d-c000.csv	2 hours ago	2.29 kB
part-00000-82df8123-990e-4237-a30e-5c334f28b4fd-c000.csv	2 hours ago	2.21 kB
part-00000-85d936e0-2f51-4c6a-af83-cc4746ed7165-c000.csv	2 hours ago	2.43 kB
part-00000-8b6da4a6-2f0c-4ed2-b014-e5539bd78fdc-c000.csv	13 hours ago	125 kB
part-00000-8baa55f4-0a62-4f29-8660-46c10fd15cc2-c000.csv	4 minutes ago	132 kB
part-00000-8e6ba89b-eeef-4701-80e6-07f85a4ad526-c000.csv	13 hours ago	125 kB
part-00000-928a0301-ac3b-4b09-a97f-0dee436e8f20-c000.csv	14 hours ago	138 kB
part-00000-99f3c9a3-2017-4b5b-8b80-b6075caf755a-c000.csv	13 hours ago	125 kB
part-00000-9e5a6f81-1d9b-4004-8415-8461f30190ea-c000.csv	13 hours ago	125 kB
part-00000-a05272c4-e6df-45ec-a849-16de71392b26-c000.csv	10 hours ago	81.2 kB
part-00000-ae10ddff-3dfe-410a-a48c-5302279095cd-c000.csv	13 minutes ago	132 kB
part-00000-b4772eef-5874-46be-ac8c-5529485223e8-c000.csv	14 hours ago	85.8 kB
part-00000-c24d8b04-a269-4b73-8411-7329a365455d-c000.csv	13 hours ago	125 kB
part-00000-c595c232-5f73-42bf-a9de-1ca1cd2fca22-c000.csv	13 hours ago	125 kB

Model Training:

To train the ML sentiment model we followed the below procedure

- Data Processing
- Initialization of model variables / Data division / Fit and Validation of mode

Data Processing:

- Using the GLOB library we loaded all files into single DF.
- Because tweets were free text we only tested percentage of non alpha tweets that was about 6 perrcentage.

- We used a feature to clean up tweets as we feed tweets into a model of ML. We need the needless words removed from tweets. We had implemented the approach using regex.

- Used Textblob model in order to obtain the tweets' polarity-scores and allocated a new columns in pandas dataframe
- Created two new columns to obtain Sentiment score & definition that is described based on polarity score. If < = 0 we tagged it to negative feeling and scored it to 0, while if > 0 we tagged it to positive feeling and scored it to 1

Initialization of model variables & Data division / Fit and Validation of mode:

- We had divided data into train and test splits using the test train split
- Initialized TFIDF vector & performed fit & transformed the train data in single step.
- We have train features and test features separately in various TFIDF variables
- Logistic Regression & Decision Tree Classification, Random Forest Classification and Gradient initialized
- Generated pipeline structure for each model for respective hyperparameters
- Now we fit TFIDF training features for each classifier along with the positive & negative sentiment definition
- Next we have predicted the test features of the ML models
- All models observed the accuracy.
- Now we have downloaded the code TFIDF & ML(LR) using the jsonl

Sentiment Comparison Model (User-Defined Functions):

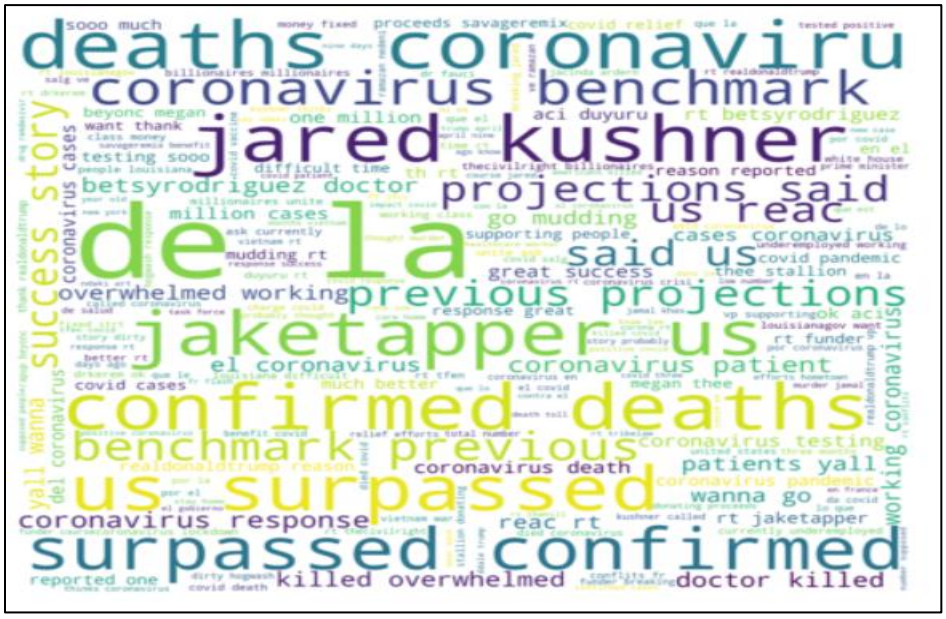
3 User Defined Functions have been created to serve the purpose

- **Data Cleaning**: Removing the junk characters such as smileys & other language characters, etc. that that affect the output of ML model being trained.
- **TextBlob Prediction**: This UDF will take the parameter of tweeting and call the TextBlob feature with the tweet, while the TextBlob will give the tweet a polarity of feel. We defined the threshold of 0 when polarity is less than 0, the feeling would be 'Negative' then 'Positive.'
- **Model Prediction**: We store the trained ML model in a folder using jsonlib after we have trained an ML model. In this UDF we load model using the jsonlib and predict feeling for tweet and give the feeling back.

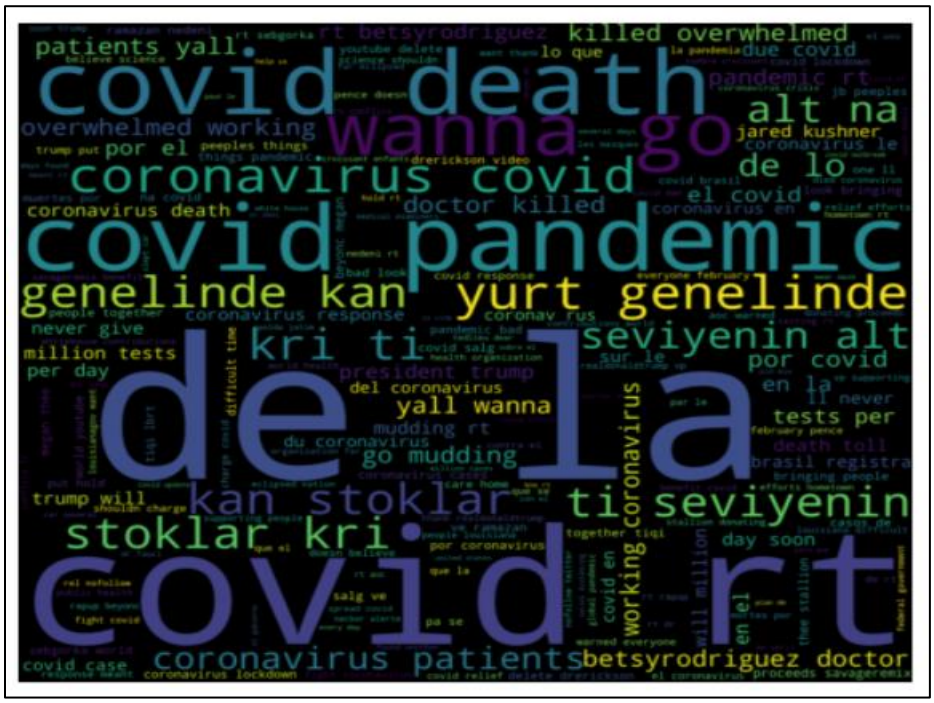
Stop Words:

```
stopwords.add('co')
stopwords.add('https')
stopwords.add('hey')
stopwords.add('hello')
stopwords.add('school')
```

Positive Words:



Negative Words:



Words with more importance:

	Word	Importance
17154	rt	0.032737
8915	https	0.021289
4067	coronavirus	0.018957
13310	new	0.015749
4186	covid	0.008859
3793	confirmed	0.008136
6164	en	0.008105
2868	cases	0.008035
11351	live	0.006637
14353	pandemic	0.006565
15184	positive	0.006190
10353	just	0.004882
4686	deaths	0.004159
5808	earth	0.004084
19001	surpassed	0.004070
15953	que	0.003942
8156	great	0.003768
9252	important	0.003670
1995	best	0.003613
6199	energy	0.003557

Words with no importance:

	Word	Importance
0	aa	0.0
1	aad	0.0
3	aafaizli	0.0
7	aaionwgdte	0.0
8	aajkamrankhanrt	0.0
...	...	...
22204	zxq	0.0
22208	zyyoif	0.0
22209	zyywanedy	0.0
22210	zz	0.0
22213	zzoeiqdmoh	0.0

Project Management:

Lalith Chandra Attaluri(50%):

- Server Implementation
- Data Visualisation: Tweets from different sources, Tweets from different counties
- Data Pre processing & UDF's creation
- Building the model
- Documentation

Pranitha Karumanchi(50%):

- Client Implementation
- Data Visualisation: Top 10 hash tags & Top 10 URL's
- Training and Testing the model
- Word Cloud Implementation

Work to be completed:

- Predicting the score by training the model with real time tweets
- Comparing the accuracy and differences between UDF and Custom model.

Remaining work percentage (15%)

Report for Final submission:

Finally our model is built and we have to make the predictions by using our already built model for the live streaming tweets.

Here we displayed the table in which first column contains user ID, second column contains the sentiment prediction of our custom build model and the final column contains the sentiment predicted by the user defined model.

Please find the code snippet below:

```
import re,pandas as pd
from joblib import dump,load
from textblob import TextBlob


model=load('lr.joblib')
tfidf_temp=load('tfid.joblib')


def data_cleansing(corpus):
    letters_only = re.sub("[^a-zA-Z]", " ", corpus)
    words = letters_only.lower().split()
    return( " ".join( words ))

def model_predict(text):
    t_a=tfidf_temp.transform(pd.Series(text).astype(str))
    pred1 = model.predict(t_a)
    print(pred1)
    return (pred1[0])

def model_textblob(text):
    score=TextBlob(str((text).encode('ascii', 'ignore'))).sentiment.polarity
    if score <= 0.0 :
        return ('negative')
    else:
        return ('positive')

from pyspark.sql.types import StringType
spark.udf.register("custom_udf", data_cleansing, StringType())
spark.udf.register("custom_predict",model_predict,StringType())
spark.udf.register("texblob",model_textblob,StringType())
```

```
<function __main__.model_textblob(text)>
```

```
spark.sql("SELECT _1,custom_predict(custom_udf(_1)),texblob(custom_udf(_1)) FROM tweets2").show()
```

Please find the output below:

```
+--------------------+-----------------------------+----------------------+
|                  _1|custom_predict(custom_udf(_1))|texblob(custom_udf(_1))|
+--------------------+-----------------------------+----------------------+
|rt @yamiche: conf...|                     positive|              positive|
|rt @iacceso: ชาว...|                     negative|              negative|
|rt @danrather: a ...|                     negative|              positive|
|rt @quantum_light...|                     negative|              positive|
|rt @unian: плати ...|                     positive|              negative|
|rt @yomex_nooni: ...|                     negative|              negative|
|rt @chennaicorp: ...|                     negative|              positive|
|rt @bipinchauhan:...|                     negative|              negative|
|rt @rtbfinfo: "de...|                     negative|              negative|
|rt @tg67210: preu...|                     negative|              negative|
+--------------------+-----------------------------+----------------------+
```

Accuracy comparison for Custom and Predefined model:

Please find the performance report of our custom model and Text Blob model.

	Custom Model	Textblob model
Accuracy	**97.01%**	**21.81%**

Please find the snippets that show the accuracy of custom model and Textblob model during execution.

Accuracy for Custom Model:

```
def ML_Pipeline(clf_name):
    clf = Classifiers[clf_name]
    fit = clf.fit(train_features,train['sentiment_description'])
    pred = clf.predict(test_features)
    Accuracy = accuracy_score(test['sentiment_description'],pred)
    Confusion_matrix = confusion_matrix(test['sentiment_description'],pred)
    print('==='*35)
    print('Accuracy of '+ clf_name +' is '+str(Accuracy))
    print('==='*35)
    print(Confusion_matrix)
```

```
ML_Pipeline('lg')
```

```
=====================================================================================
Accuracy of lg is 0.9701393497013935
=====================================================================================
[[3496   68]
 [  67  890]]
```

Accuracy for Textblob Model:

```
df['textblob_score']=np.where(df.sentiment_value<=1,0,0.0)
```

```
df.head()
```

	tweet_txt	sentiment_value	sentiment_score	sentiment_description	textblob_score
0	covid salg n ve ramazan nedeni servus tv jetzt...	-0.200000	1	negative	0.0
1		0.000000	1	negative	0.0
2	https t co rifmjewnkbrt vanityfair obsessed wi...	0.150000	0	positive	0.0
3		0.000000	1	negative	0.0
4	experts from wto wtoddgwolff fao maximotorero ...	0.158333	0	positive	0.0

```
df['tbsentiment_description']=np.where(df.textblob_score<=0.0,'positive','negative')
```

```
df.head()
```

	tweet_txt	sentiment_value	sentiment_score	sentiment_description	textblob_score	tbsentiment_description
0	covid salg n ve ramazan nedeni servus tv jetzt...	-0.200000	1	negative	0.0	positive
1		0.000000	1	negative	0.0	positive
2	https t co rifmjewnkbrt vanityfair obsessed wi...	0.150000	0	positive	0.0	positive
3		0.000000	1	negative	0.0	positive
4	experts from wto wtoddgwolff fao maximotorero ...	0.158333	0	positive	0.0	positive

```
accuracy_score( df['tbsentiment_description'], df['sentiment_description'],)
```

```
0.21818181818181817
```

References:

- https://spark.apache.org/docs/latest/streaming-programming-guide.html
- https://towardsdatascience.com/almost-real-time-twitter-sentiment-analysis-with-tweep-vader-f88ed5b93b1c
- https://developer.twitter.com/en/docs/tweets/data-dictionary/overview/intro-to-tweet-json

Project Management:

Lalith Chandra Attaluri(50%):

- Server Implementation
- Data Visualisation: Tweets from different sources, Tweets from different counties
- Data Pre processing & UDF's creation
- Building the model
- Documentation
- Prediction comparision of the models on live tweets

Pranitha Karumanchi(50%):

- Client Implementation
- Data Visualisation: Top 10 hash tags & Top 10 URL's
- Training and Testing the model
- Word Cloud Implementation
- Accuracy comparision for Custom and pre defined models