# SENTIMENT ANALYSIS OF COVID-19 TWEETS

**TEAM NAME: A2M**
**APPLICATION ID: SPS_CH_APL_2020000903**
**PROJECT ID: SPS_PRO_1525**

**Team Members:**
1. Aastha Singh
2. Megha Gupta
3. Akshita Agarwal

## INTRODUCTION

### 1.1 Overview

Our aim is to develop a COVID-19 sentiment analysis visualization dashboard which depicts the emotions of people over the current situation and the results of analysis of news articles related to COVID-19.

Basic language that we used for coding is Python and implementation is done with the help of IBM Watson Tone Analyzer, Google BERT, MALLET based topic modelling, Dash. The former two technologies are used for analysing tweets and labelling them with five kinds of emotions. MALLET is used for news analysis and at the end the result of both BERT and MALLET is represented in pictorial form using DASH.

The visualization dashboard can be viewed [here](here) .

### 1.2 Purpose

The Corona Virus has put the entire world at a standstill and governments all around the world are employing various methods to keep the situation under control. At this point, analysing the sentiments and emotions of people can help provide an insight into how a particular decision will influence the public. The situation also poses a threat to the mental health of an individual, understanding how the public reacts to a decision can be further used to help people overcome anxiety and stress. In today's word of technology the best way to do so is by analysis of people's reactions or

comments on social media platform because nowadays most of the public express their concern or emotions through social media. For our project we have taken twitter as our source of data.

## LITERATURE SURVEY

### 2.1  Existing Problem

Analysing the sentiments of the public during the Corona Virus Pandemic can help in a variety of ways to both the government and instilling a sense of unity among the people, that they are not alone.

Most of the existing sentiment analysis make the use of TextBlob to calculate polarity and subjectivity of a preprocessed text. This only allows us to analyse if given is positive, negative or neutral. Polarity is a float value which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1].

### 2.2 Proposed Solution

For the Covid-19 sentiment analysis visualization dashboard, the proposed solution is as follows:

For creating our dataset, we made the use of IEEE database of COVID 19 Tweets. We obtain the corresponding Tweet with the help of Hydrator and store the tweets into a CSV file after extraction of tweets. The tweets obtained cannot be used

directly. Hence, we'll pre-process the tweets using a combination of pre-processing techniques and extract the intensity of sentiments using TextBlob (polarity and subjectivity). After this, we use the IBM Watson Tone Analyzer along with manual tags so that we can label the sampled tweets with the above five sentiments. We perform some fine-tuning with the help of Google BERT based models. We will also use MALLET for further topic modelling of news articles. This way we can analyse the major topics in talk during the Corona Virus pandemic. The dashboard is built with the help of DASH framework by Python. The dashboard contains graphs of different recorded metrics that'll help us analyse the sentiment of the public.

## THEORITICAL ANALYSIS

### 3.1 Block diagram
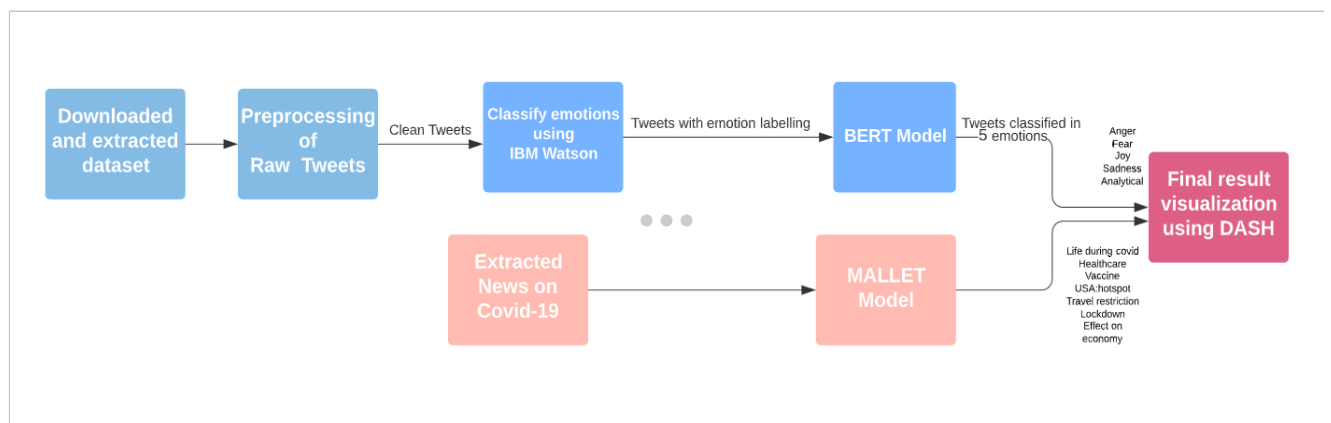


### 3.2 Software designing

We have used various software and API to make a model which predicts emotion of a tweet with very less error. The ones which are the base for the project are IBM Watson Tone Analyser, BERT, MALLET and Dash. Using IBM Watson we created data for creating a training dataset through which we can teach our BERT model. In BERT we created a ML model which took the training set and learned and predicted

for a larger dataset the emotions of each tweet. We used MALLET for news scrapping and its output tells us the statistics of words commonly used in everyday basis in context of Covid-19. At last Dash was used to create a final dashboard which presented all the output we got from tweets and news in a graphical form.

## EXPERIMENTAL INVESTIGATION

For creation of databases, we initially extracted tweets with the help of Tweepy. We could obtain tweets from Tweepy for a short period of time. Hence, it did not provide any information as to why there is a change in sentiment values. To provide a better analysis as to what effect the lock down will have, we used IEEE database of corona virus Tweet IDs from March 21 to May30, the period of lock down in India. With data from over three months, we were able to analyse why people reacted the way they did at different times of the lockdown.

Analysing the most common words used during the lockdown gave us a better insight into the mindset of the public. The initial analysis of polarity and subjectivity of tweets gave us an idea about how the mood of the public is. By using IBM Watson Tone Analyser, we were able to further classify it into different emotions such as Joy, Sadness, Fear, Anger and Analytical. We also worked on improving the accuracy of the BERT model to detect and predict emotions with higher accuracy.

In the MALLET model, manual labelling of keywords helped us analyse what are the major topics of interest during the COVID 19 pandemic.
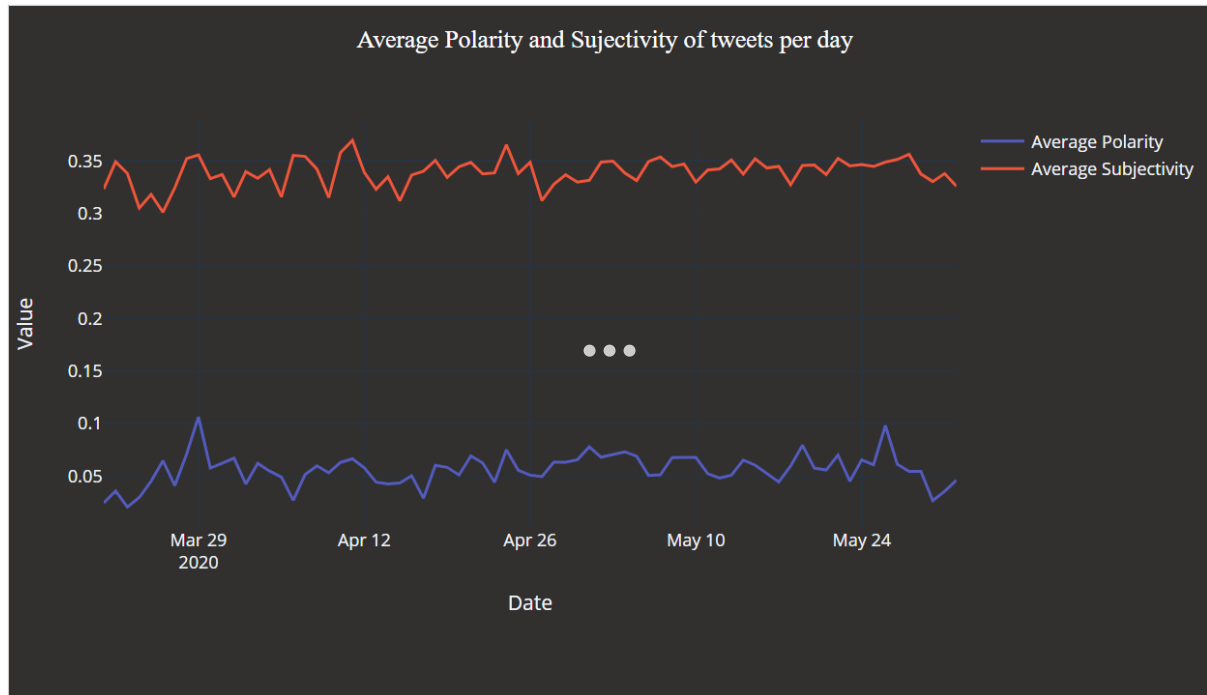
# **FLOWCHART**
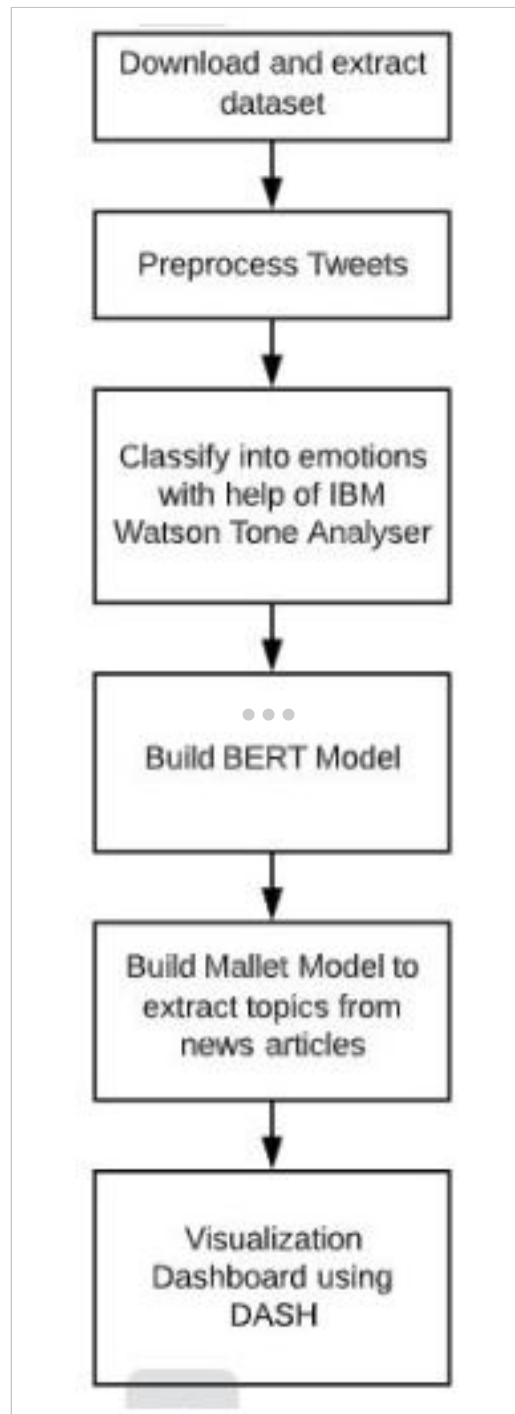
**RESULTS**

## 5.1 Sentiment Analysis of #COVID-19 Tweets

### 5.1.1  Average polarity and subjectivity of the Tweets



| Metric Name | Description | Equation |
|---|---|---|
| Average polarity /Average Subjectivity | Calculates the average polarity and sentiment value of tweets per day | Mean of polarity/subjectivity of all the tweets in a single day |

Average polarity recorded = 0.056

Average subjectivity recorded = 0.339

According to the chart above, with the development of COVID-19, the related tweets' expression became more subjective. From average subjectivity being 0.330 in the month of March, it increased to 0.342 in the month of May and the Polarity of tweets also saw a trend of becoming more positive gradually over the time. The average polarity rose from 0.509 in the month of March to 0.059 in the month of May. The question that arises after such an analysis is Why with more and more people being infected with Coronavirus, the sentiment of related tweets went positive? We try to answer this question in the subsequent analysis.

## 5.1.2 *Positive Negative and Neutral analysis*



Positive Sentiment        Negative Sentiment        Neutral Sentiment
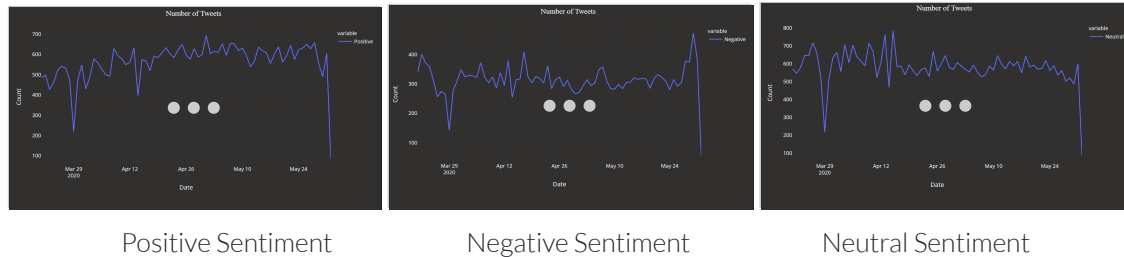
Diving deep into the previous analysis, we differentiated positive, negative and neutral tweets. As mentioned about the polarity becoming increasingly positive meaning that the rise in positive tweets should be significantly larger than the rise in other two. So here we see the average number of positive tweets per day increased from being 471 in march to 593 in May (25.9%) whereas the percentage rise in neutral and negative tweets are -1.9% (minus represents drop) and 3.33% respectively. Thus supporting previous analysis.
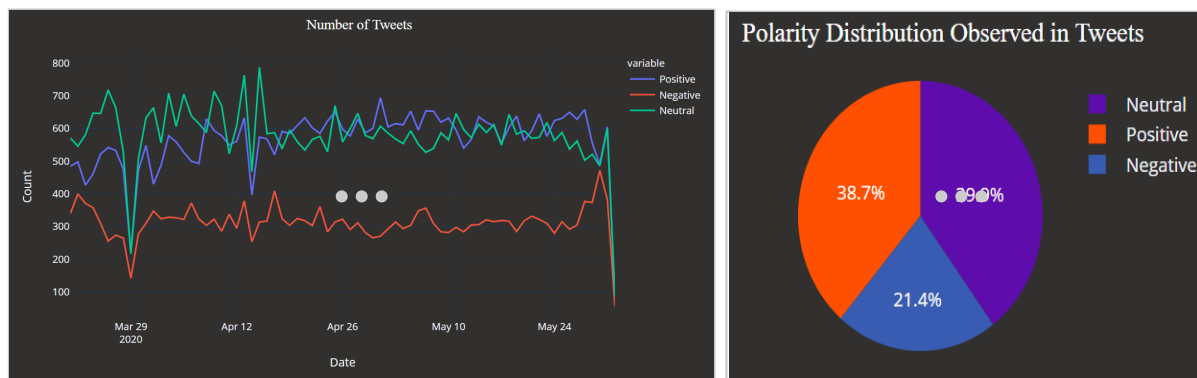


For positive,Negative,Neutral Sentiments  a) Number of tweets  b)Polarity Observed in Tweets

Here we see, the majority of tweets (39.9%) have a negative sentiment. This supports the current havoc of the mind state of the public in the current scenario. At the start of lockdown, the negative sides of people were dominant but with time, the adjustment with the 'new normal' can be seen as the graph of negative tweets dips. The occasional high peaks of all the tweets occur on the days of announcement of lockdown extension. Hence showing that all sentiments are stronger at the announcement of extension however negativity is the most recorded emotion. However, it is eased with the duration of time.

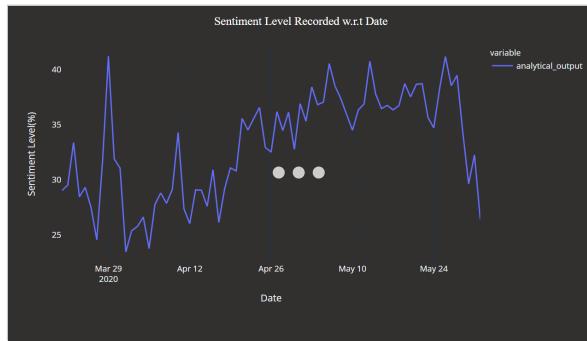## 5.2 Five Emotions Analysis of #COVID 19 Tweets

### 5.2.1  *Sentiment level of all the emotions of the tweets*
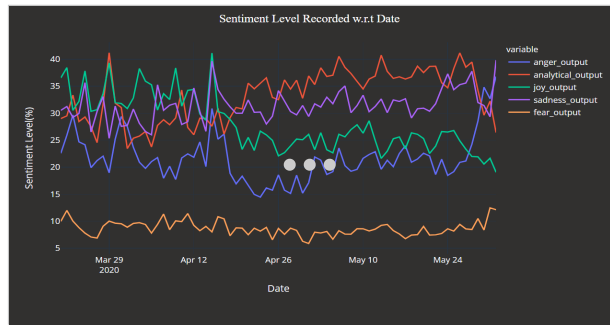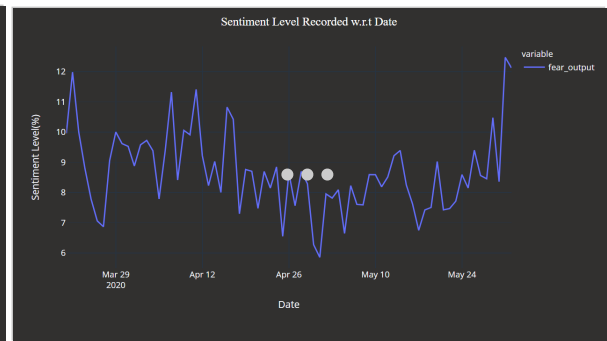
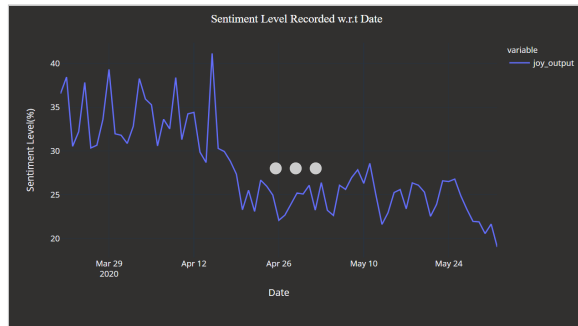

All emotions Sentiment level
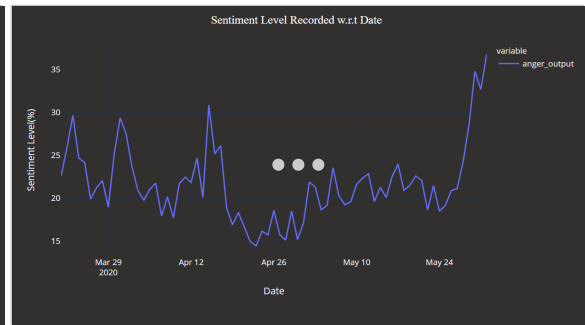


'Anger' Sentiment level


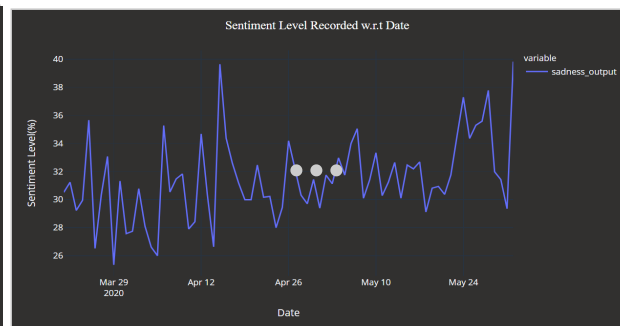
'Analytical'  Sentiment level



'Fear' Sentiment level



'Joy' Sentiment level



'Sadness' Sentiment level

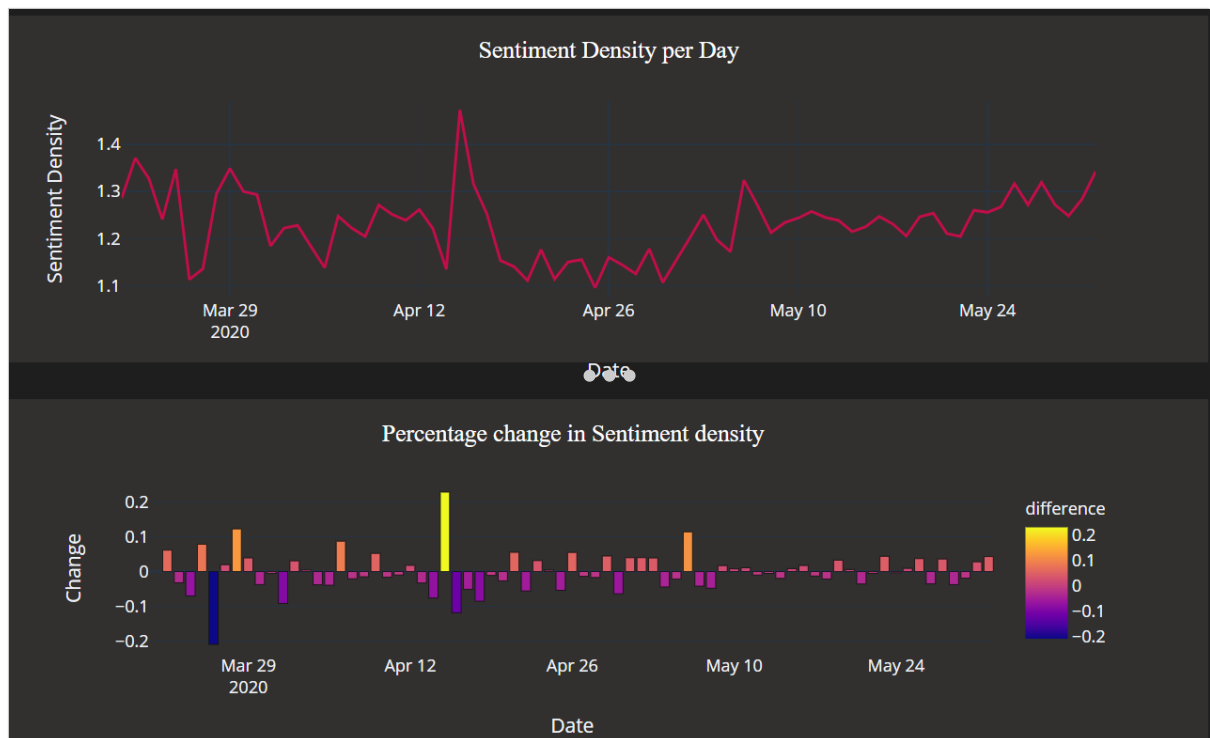| Metric Name | Description | Equation |
|---|---|---|
| Sentiment level (%) | A proxy for of the intensity of a certain sentiment | count (tweets with certain sentiment) / count (total tweets) |

Before the first announcement of lockdown (or lockdown 1.0), surprisingly, sentiment 'Joy' was detected in most of the tweets with 'Sadness' and 'Analytical' closely competing for second position. However, after the announcement, the sentiment 'Joy' in the tweets saw a decline and 'Analytical' and 'Sadness' were on a rise. Right after the announcement of lockdown, 'Anger' and 'Sadness' saw a sudden rise while 'Joy' and 'Fear' reduced significantly. Most rise is seen in the sentiment 'Analytical'. With every subsequent announcement of extension of lockdown, 'Analytical' sentiment levels have risen. Interesting trend is observed in 'Fear', with the extension of lockdown, the 'fear' component started appearing less in the tweets.Though when the lockdown was finally lifted (May 30th) a peak is observed indicating a sudden rise in fear in public. Overloaded with information seems to have made people less sensitive.'Anger' has shown a significant decline until the end of may. Probably, the news of lockdown being lifted instead of infected cases rising causes a restlessness in the public. 'Sadness' is the sentiment which shows a constant increase.
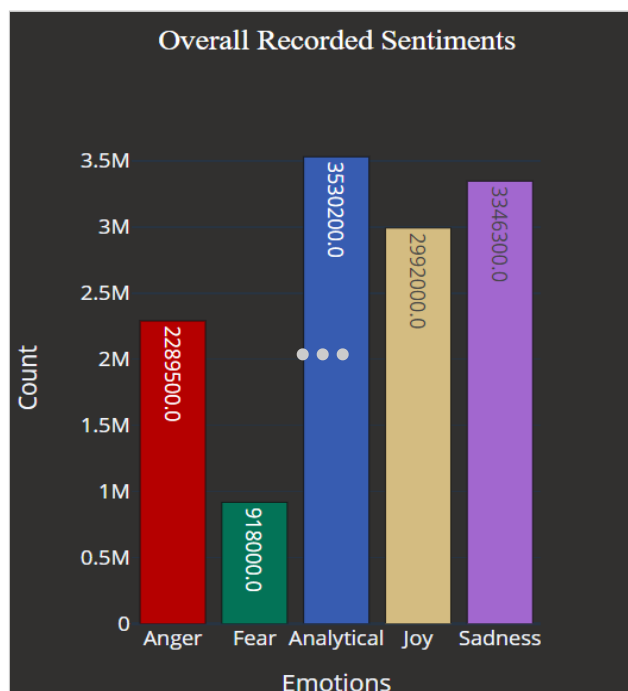


From above it is noted that the sentiment 'Analytical' was most recorded in all the tweets, and 'Sadness' was the next most discovered sentiment. 'Fear' was recognised the lowest.

### 5.2.1  *Emotion Density*



| Metric Name | Description | Equation |
|---|---|---|
| Sentiment Density | The average count of sentiment that a tweet posses | COUNT (all the sentiments) / COUNT (all the tweets) |
| % Change in Sentiment Density | The percentage change in daily Sentiment density | [ Sentiment Density on day(n) / Sentiment density on day (n-1) ]-1 |

Keeping in mind that there could be more than one sentiment present in a tweet, we computed the Sentiment Density to show that on average how many different sentiments a tweet had on a single day. This figure will give us a direct impression of how much the tweets were "packed with" different emotions on a day. We then computed the day-on-day change of these metrics and formed the delta metrics.

Through the general trend of Sentiment Density in the above dashboard, we can infer that from late March till mid-April, people were undergoing less dense sentiments, especially in terms of the positive feelings.Before that people were going

through densest emotions.In May, the Sentiment Density started increasing and towards the end of may it reached to the same level as the sentiment density before the lockdown was announced. This could be interpreted that people started properly adjusting to the situation in May or after.

**5.3 Topic Modelling on News**
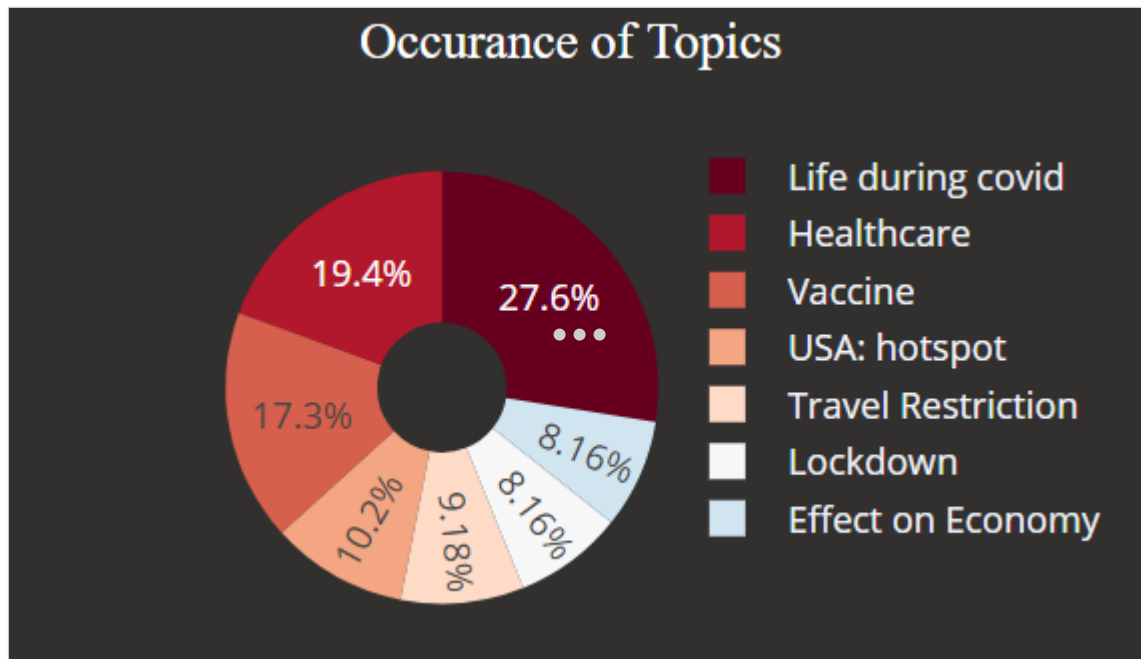


We utilized Mallet, a natural language processing toolkit, to perform Latent Dirichlet Allocation (LDA) topical modelling, and summarized 7 topics. We named these topics by summarizing the topic keywords returned by the model, and they are as follows (following the descending sequence of frequency): Life during COVID-19, Healthcare, Vaccine, USA: hotspot, Travel Restrictions,Lockdown, Effect on Economy. Equipping with the TextBlob's sentiment analysis, the trending of these topics over time are as follows:

**Polarity and Subjectivity of News Articles**
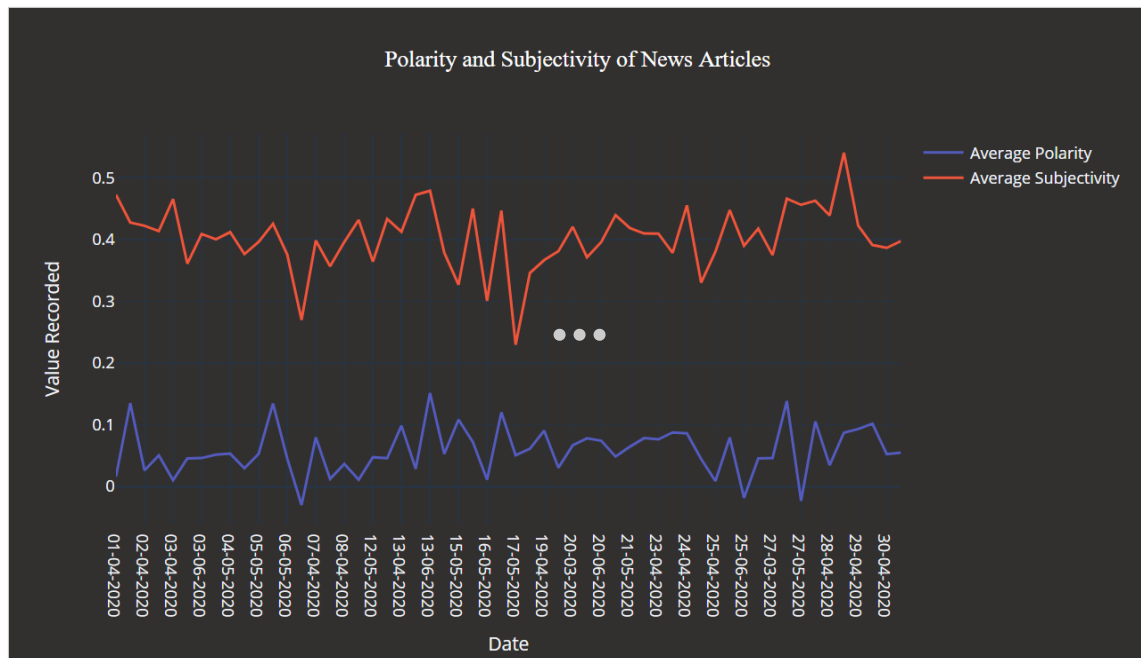
For the sentiment of all topics in news, we saw that average polarity is around 0.059 which indicates that on average the news reports contain positive news. Although the news articles on the day after the announcement of lockdown extension are made, show negative polarity denoting 'Lockdown' in general is conceived as a 'negative' news. Although after the announcement of lockdown 3.0, the polarity is maintained at a positive side.

## APPLICATIONS

1. The project can be used to analyse the sentiments of the people and help find solutions to uplift the general mood of the citizens.

2. The government can further use the analysis information to decide the impact of a particular policy and which policy to roll out next.

3. Helpline numbers and other programs can also be started to help the public deal with anxiety and stress. Timely action in such cases can help the public tremendously.

4. The data can be further studied to decide the general reaction and sway of the public.

# CONCLUSION

The project helps us in analysing the trend of tweets and news articles related to the COVID-19 pandemic. This helps us realise the major thoughts and concerns of the public and their reaction to the lockdown. The project helps:

1. Analyse the polarity (positive, neutral and negative) and subjectivity of tweets

2. Detect tones present in tweets, namely Anger, Fear, Joy, Sadness and Analytical.

3. Analyse the rise and fall of positive, neutral and negative sentiments during the lockdown.

4. Identify major topics of interest during the lockdown.

5. Calculate Sentiment Density to indicate on average how many different sentiments a tweet has on a given day.

# FUTURE SCOPE

The project can be integrated with other applications to provide an all round analysis of how COVID-19 has impacted the world.

1. Display the count of positive and recovered corona virus patients.

2. Display status of medicines and other government notices.

3. Mask detection and number of people wearing masks.

# BIBLIOGRAPHY

1. https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset

2. https://pandas.pydata.org/

3. https://towardsdatascience.com/twitter-sentiment-analysis-based-on-news-topics-during-covid-19-c3d738005b55

4. https://www.ibm.com/watson/services/tone-analyzer/

5. https://www.datacamp.com/community/tutorials/learn-build-dash-python

# APPENDIX

## A. SOURCE CODE

## 1. BERT MODEL

```python
import tensorflow as tf
device_name = tf.test.gpu_device_name()
import torch
!pip install transformers
import numpy as np
import pandas as pd
import os
from google.colab import files
uploaded = files.upload()
import io
df = pd.read_csv(io.BytesIO(uploaded['anger-training-github-lite.csv']))
print('Number of training sentences: {:,}\n'.format(df.shape[0]))
anger = df[['text','pred_anger']]
sentences = anger.text.values
labels = anger.pred_anger.values

from transformers import BertTokenizer
# Load the BERT tokenizer.
print('Loading BERT tokenizer...')
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
input_ids = []
# For every sentence...
```

```python
23  for sent in sentences:
24      encoded_sent = tokenizer.encode(
25                          sent,  # Sentence to encode.
26                          add_special_tokens = True,
27                              )
28    # Add the encoded sentence to the list.
29      input_ids.append(encoded_sent)
30
31  """##Padding and Trunking"""
32  from keras.preprocessing.sequence import pad_sequences
33  MAX_LEN = 250
34  # Pad our input tokens with value 0.
35  input_ids = pad_sequences(input_ids, maxlen=MAX_LEN, dtype="long",
36                          value=0, truncating="post", padding="post")
37  attention_masks = []
38  # For each sentence...
39  for sent in input_ids:
40      # Create the attention mask.
41      att_mask = [int(token_id > 0) for token_id in sent]
42      # Store the attention mask for this sentence.
43      attention_masks.append(att_mask)
44
45  """##Train/Validation Split"""
46  from sklearn.model_selection import train_test_split
47  # Use 90% for training and 10% for validation.
48  train_inputs,validation_inputs,train_labels,validation_labels=train_test_split(input_ids
    , labels, random_state=1999, test_size=0.1)
49  # Do the same for the masks.
50  train_masks, validation_masks, _, _ = train_test_split(attention_masks, labels,
51                          random_state=1999, test_size=0.1)
52  train_inputs = torch.tensor(train_inputs)
53  validation_inputs = torch.tensor(validation_inputs)
54  train_labels = torch.tensor(train_labels)
55  validation_labels = torch.tensor(validation_labels)
56  train_masks = torch.tensor(train_masks)
57  validation_masks = torch.tensor(validation_masks)
58  from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
59  batch_size = 16
60  # Create the DataLoader for our training set.
```

```python
61  train_data = TensorDataset(train_inputs, train_masks, train_labels)
62  train_sampler = RandomSampler(train_data)
63  train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)
64  # Create the DataLoader for our validation set.
65  validation_data = TensorDataset(validation_inputs, validation_masks, validation_labels)
66  validation_sampler = SequentialSampler(validation_data)
67  validation_dataloader  =  DataLoader(validation_data,  sampler=validation_sampler,
    batch_size=batch_size)
68  from transformers import BertForSequenceClassification, AdamW, BertConfig
69  model = BertForSequenceClassification.from_pretrained(
70      "bert-base-uncased",
71      num_labels = 2,
72      output_attentions = False,
73      output_hidden_states = False, # Whether the model returns all hidden-states.
74  model.cuda()
75
76  """##Set Optimizer and Learning Rate"""
77  optimizer = AdamW(model.parameters(),
78                  lr = 5e-5, # args.learning_rate
79                  eps = 1e-8 # args.adam_epsilon
80              )
81  from transformers import get_linear_schedule_with_warmup
82  epochs = 4
83  total_steps = len(train_dataloader) * epochs
84  # Create the learning rate scheduler.
85  scheduler = get_linear_schedule_with_warmup(optimizer,
86                      num_warmup_steps = 0, # Default value in run_glue.py
87                      num_training_steps = total_steps)
88  import numpy as np
89  def flat_accuracy(preds, labels):
90      pred_flat = np.argmax(preds, axis=1).flatten()
91      labels_flat = labels.flatten()
92      return np.sum(pred_flat == labels_flat) / len(labels_flat)
93  import time
94  import datetime
95  def format_time(elapsed):
96      '''
97      Takes a time in seconds and returns a string hh:mm:ss
98      '''
```

```python
99      # Round to the nearest second.
100     elapsed_rounded = int(round((elapsed)))
101     # Format as hh:mm:ss
102     return str(datetime.timedelta(seconds=elapsed_rounded))
103
104 import random
105 seed_val = 42
106 random.seed(seed_val)
107 np.random.seed(seed_val)
108 torch.manual_seed(seed_val)
109 torch.cuda.manual_seed_all(seed_val)
110 loss_values = []
111
112 # For each epoch...
113 for epoch_i in range(0, epochs):
114
115     print("")
116     print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
117     print('Training...')
118     # Measure how long the training epoch takes.
119     t0 = time.time()
120     # Reset the total loss for this epoch.
121     total_loss = 0
122     model.train()
123     # For each batch of training data...
124     for step, batch in enumerate(train_dataloader):
125
126         # Progress update every 40 batches.
127         if step % 40 == 0 and not step == 0:
128             # Calculate elapsed time in minutes.
129             elapsed = format_time(time.time() - t0)
130
131             # Report progress.
132             print('  Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format(step,
    len(train_dataloader), elapsed))
133         b_input_ids = batch[0].to(device)
134         b_input_mask = batch[1].to(device)
135         b_labels = batch[2].to(device)
136         model.zero_grad()
```

```python
137          outputs = model(b_input_ids,
138                      token_type_ids=None,
139                      attention_mask=b_input_mask,
140                      labels=b_labels)
141
142          loss = outputs[0]
143          total_loss += loss.item()
144          loss.backward()
145 torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
146          optimizer.step()
147          scheduler.step()
148      avg_train_loss = total_loss / len(train_dataloader)
149      loss_values.append(avg_train_loss)
150
151      print("")
152      print("  Average training loss: {0:.3f}".format(avg_train_loss))
153      print("  Training epcoh took: {:}".format(format_time(time.time() - t0)))
154      print("")
155      print("Running Validation...")
156
157      t0 = time.time()
158      # during evaluation.
159      model.eval()
160
161      # Tracking variables
162      eval_loss, eval_accuracy = 0, 0
163      nb_eval_steps, nb_eval_examples = 0, 0
164
165      # Evaluate data for one epoch
166      for batch in validation_dataloader:
167          batch = tuple(t.to(device) for t in batch)
168
169          # Unpack the inputs from our dataloader
170          b_input_ids, b_input_mask, b_labels = batch
171          with torch.no_grad():
172              outputs = model(b_input_ids,
173                          token_type_ids=None,
174                          attention_mask=b_input_mask)
175          logits = outputs[0]
```

```
176          # Move logits and labels to CPU
177          logits = logits.detach().cpu().numpy()
178          label_ids = b_labels.to('cpu').numpy()
179          # Calculate the accuracy for this batch of test sentences.
180          tmp_eval_accuracy = flat_accuracy(logits, label_ids)
181          # Accumulate the total accuracy.
182          eval_accuracy += tmp_eval_accuracy
183          # Track the number of batches
184          nb_eval_steps += 1
185              print("  Accuracy: {0:.3f}".format(eval_accuracy/nb_eval_steps))
186      print("  Validation took: {:}".format(format_time(time.time() - t0)))
187 print("Training complete!")
188 import matplotlib.pyplot as plt
189 import seaborn as sns
190 sns.set(style='darkgrid')
191 sns.set(font_scale=1.5)
192 plt.rcParams["figure.figsize"] = (12,6)
193 plt.plot(loss_values, 'b-o')
194 plt.title("Training loss")
195 plt.xlabel("Epoch")
196 plt.ylabel("Loss")
197 plt.show()
```

The complete code is available on github.