SENTIMENT ANALYSIS OF COVID-19 TWEETS

TEAM NAME: A2M
APPLICATION ID: SPS_CH_APL_2020000903

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 5 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 Visualizattion dashboard can be viewed from 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, analyzing 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 todays word of technology the best way to do so is by analysis peoples rections or comments on social media platform because nowdays 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

Analyisng the sentiments of the public during the Corona Virus Pandemic can help in a variety of ways to both the governments 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 float which lies in the range of [-1,1] where 1 means positive statement and -1 means a negative statement. Subjective sentences generally refer to personal opinion, emotion or judgment whereas objective refers to factual information. Subjectivity is also a float which lies in the range of [0,1].

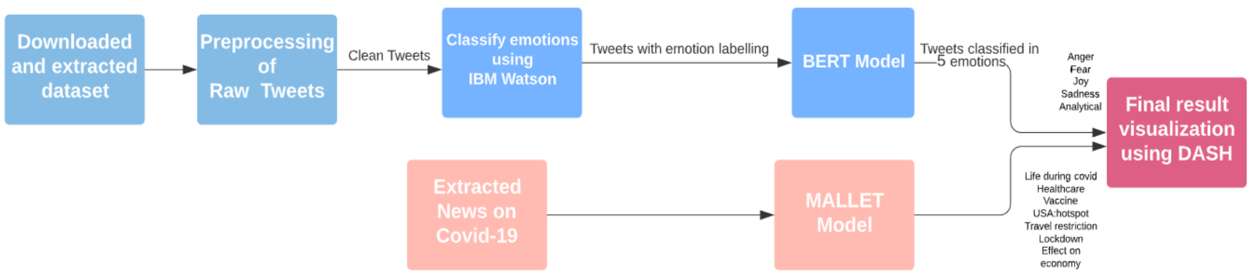
2.2 Proposed Solution

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 subjectivty). After this, we use the IBM Watson Tone Analyzer along with manual tags so that we can label the sampled tweets with the above 5 sentiments. We perform some fine tuning with the help of Google BERT based model. We will also use MALLET for further topic modelling of news articles. This way we're able to 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 anayze the sentiment of the public.

THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Software designing

We have used various softwares 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 creatind a training dataset though 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 tell us the statistics of words commonly used in everyday basis in context of covid-19. At last Dash was used to creat a final dashboard which presented all the output we got from tweets and news in a graphical form.

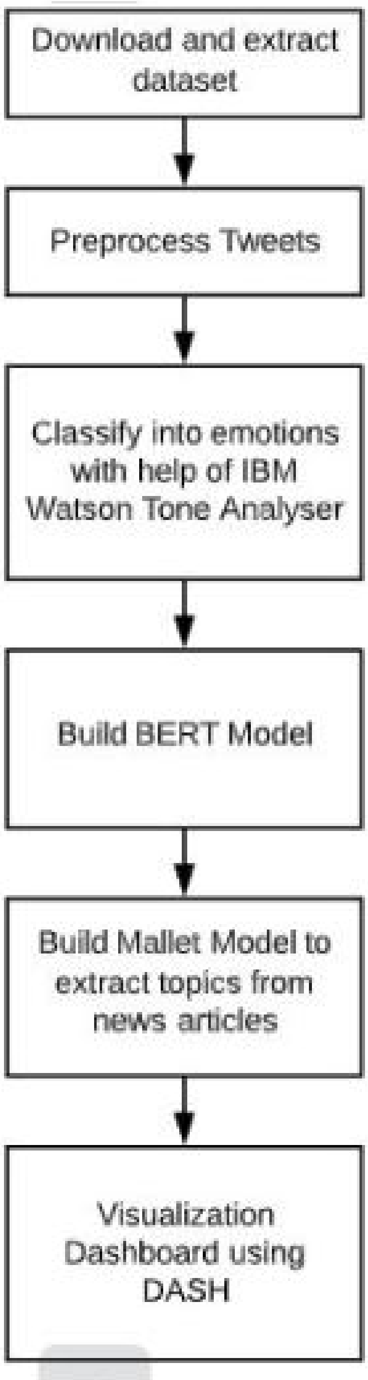
EXPERIMENTAL INVESTIGATION

For creation of database, we initially extracted tweets with the help of Tweepy. We could obtain tweets from Tweepy for a very 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 lockdown will have, we used IEEE database of corona virus Tweet IDs from March 21 to May30, the period of lockdown in India. With data from over 3 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 subjectivty 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

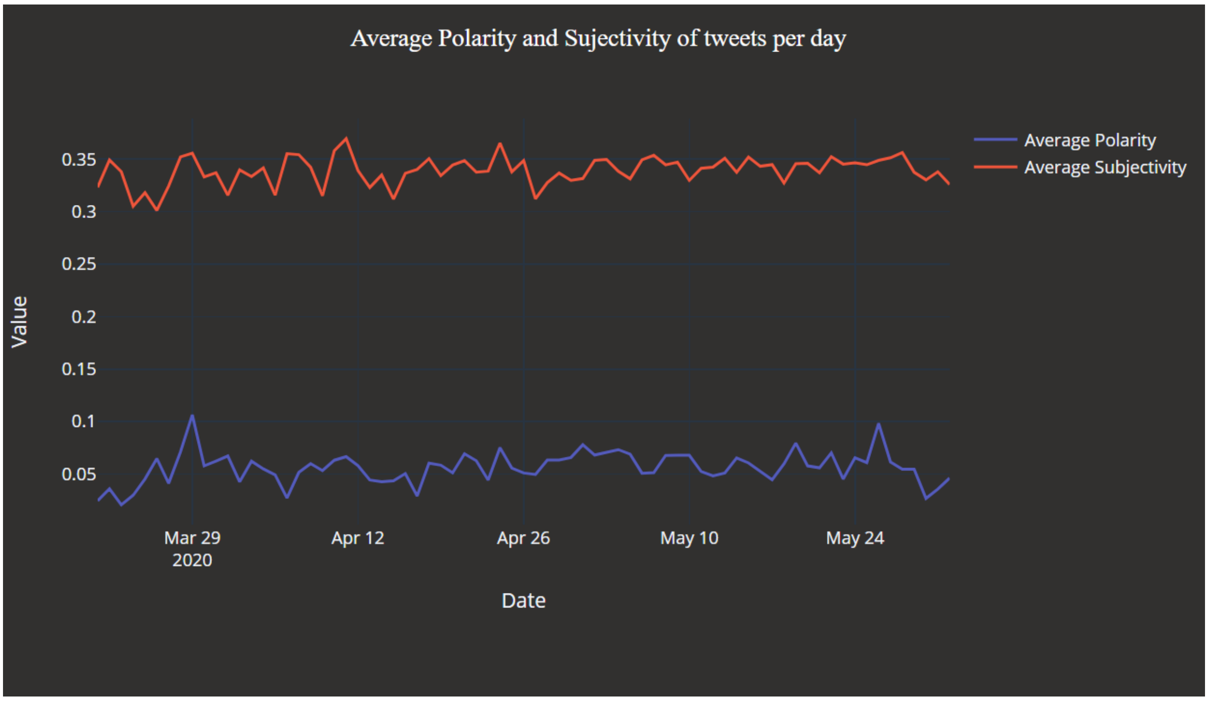


Download and extract dataset
Preprocess Tweets
Classify into emotions with help of IBM Watson Tone Analyser
Build BERT Model
Build Mallet Model to extract topics from news articles
Visualization Dashboard using DASH

RESULTS

5.1 Sentiment Analysis of #COVID-19 Tweets

5.1.1 Average polarity and subjectivity of the Tweets



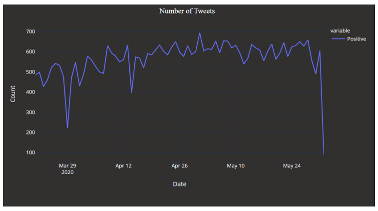
Metric Name	Description	Equation
Average polarity /Average Subjectivity	Calculates the average polarity or sentiment value of tweets per day	Mean of polartiy/subjectivity of all the tweets in a single day

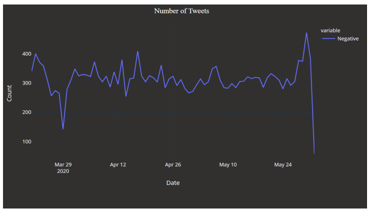
Average polarity recorded = 0.056

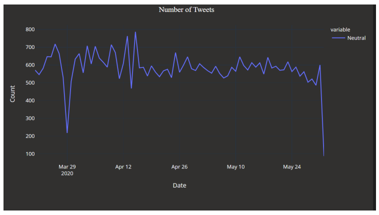
Average subjectivity recorded = 0.339

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 being 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 being 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

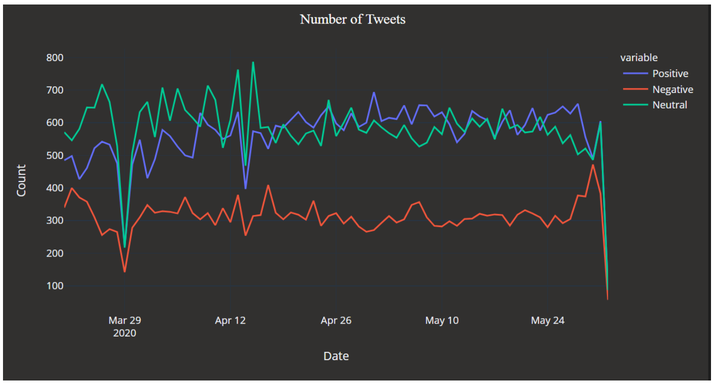


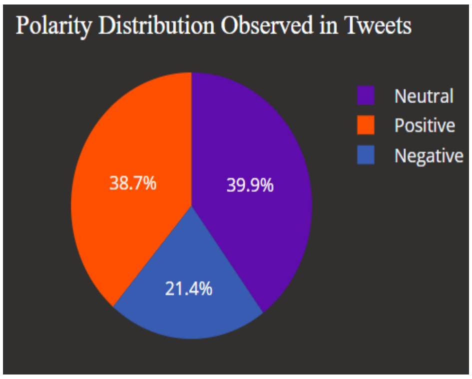




Positive Sentiment   Negative Sentiment   Neutral Sentiment

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 no 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.



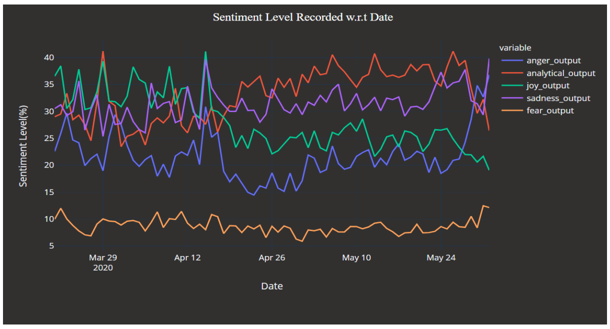


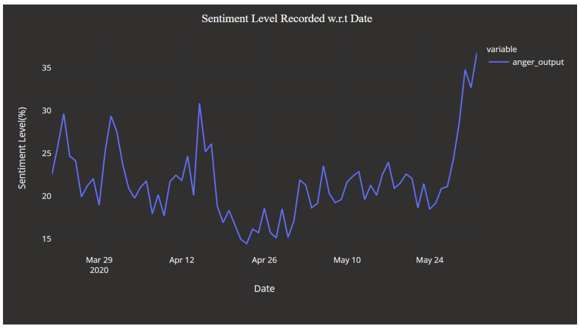
For positive,Negative,Neutral Sentiments a) Number of tweets b)Polarity Observed in Tweets

Here we see, the majority of tweets (39.9%) have a negative sentiment. This supports the current havoc of the mindstate 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. But it is eased with the duration of time.

5.2 Five Emotions Analysis of #COVID 19 Tweets

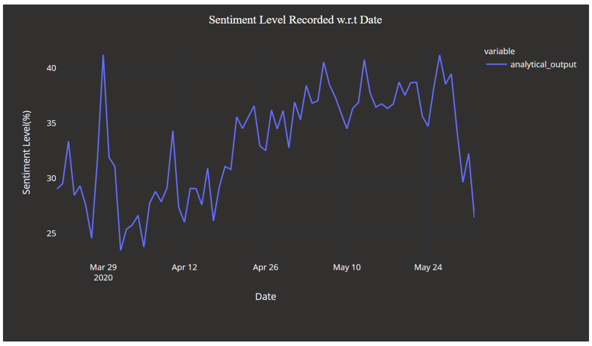
5.2.1 Sentiment level of all the emotions of the tweets

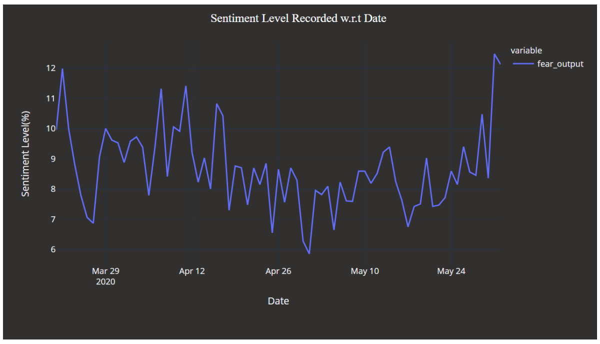




All emotions Sentiment level

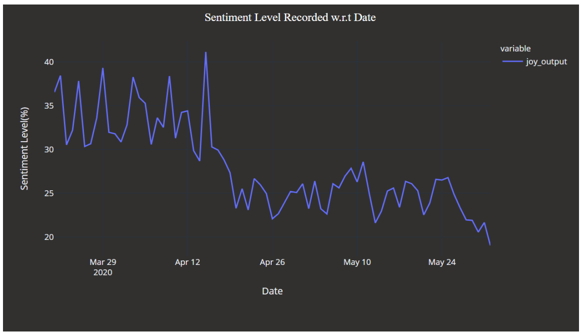
'Anger' Sentiment level

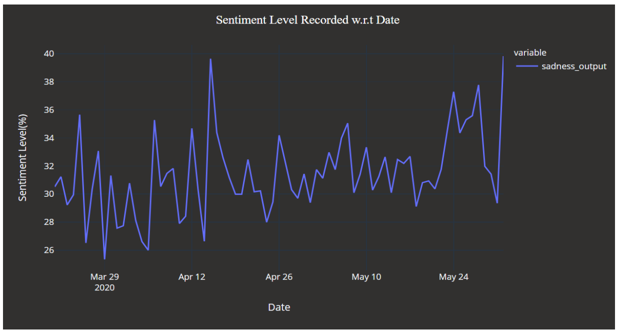




'Analytical' Sentiment level

'Fear' Sentiment level





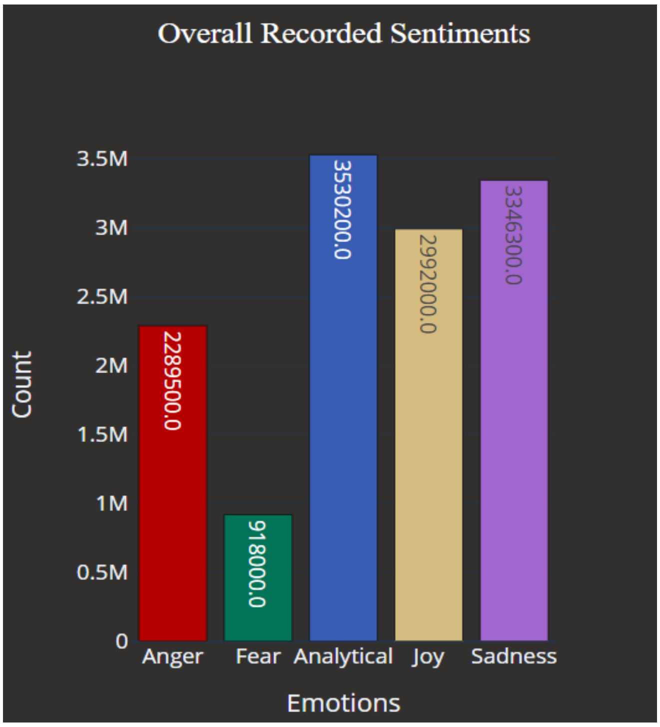
'Joy' Sentiment level

'Sadness' Sentiment level

Metric Name	Description	Equation

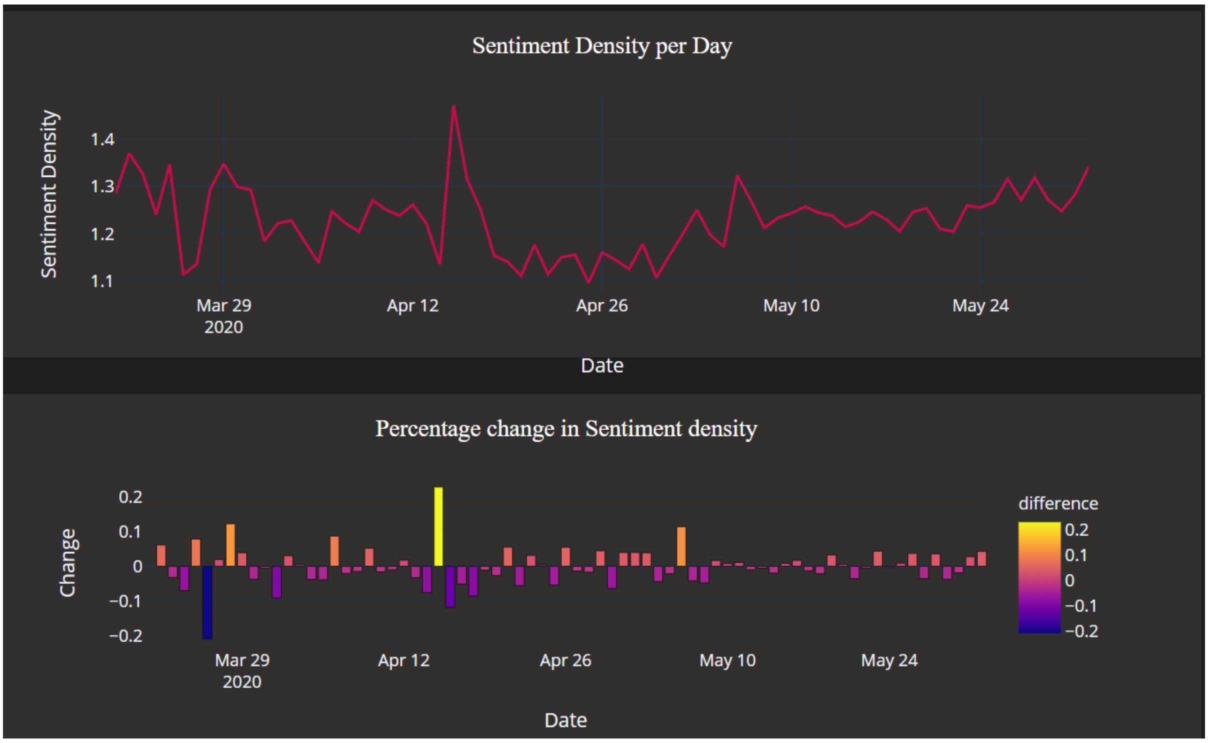
Sentiment level (%)	A proxy for of the intensity of a certain sentiment	count (tweets with certain sentiment) / count (total tweets)

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

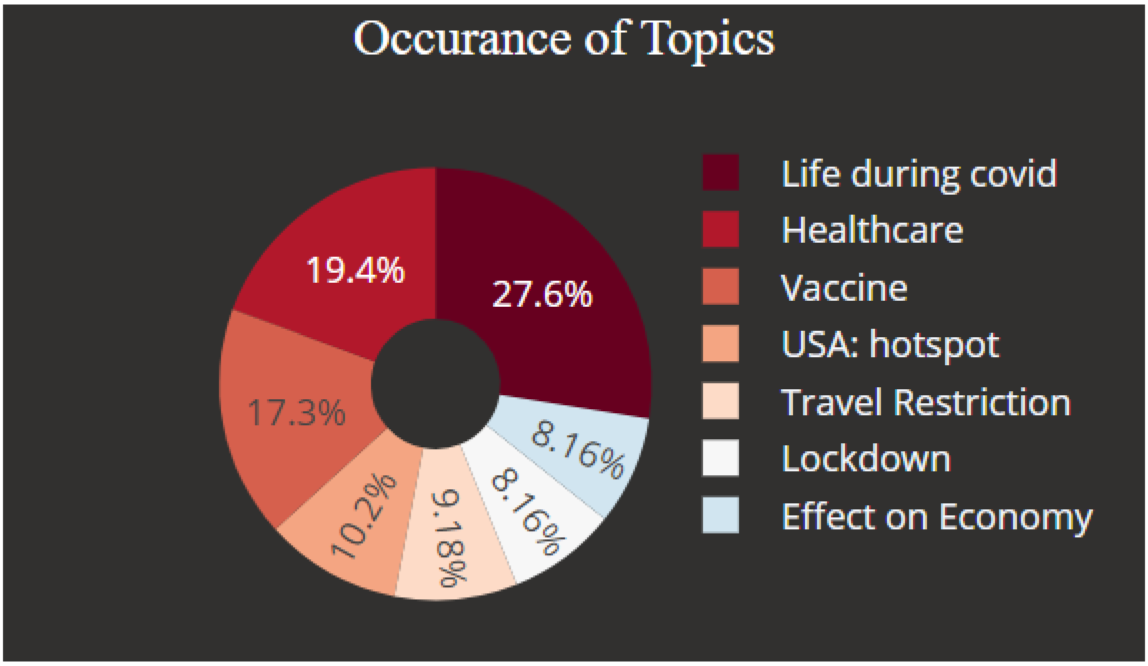


Metric Name	Description	Equation
Sentiment Density	The average count of sentiment that a tweet posses	COUNT (all the sentiments) / COUNT (all the tweets)
% Change in Sentiment Density	The percentage change in daily Sentiment density	[Sentiment Density on day(n) / Sentiment density on day (n-1)]-1

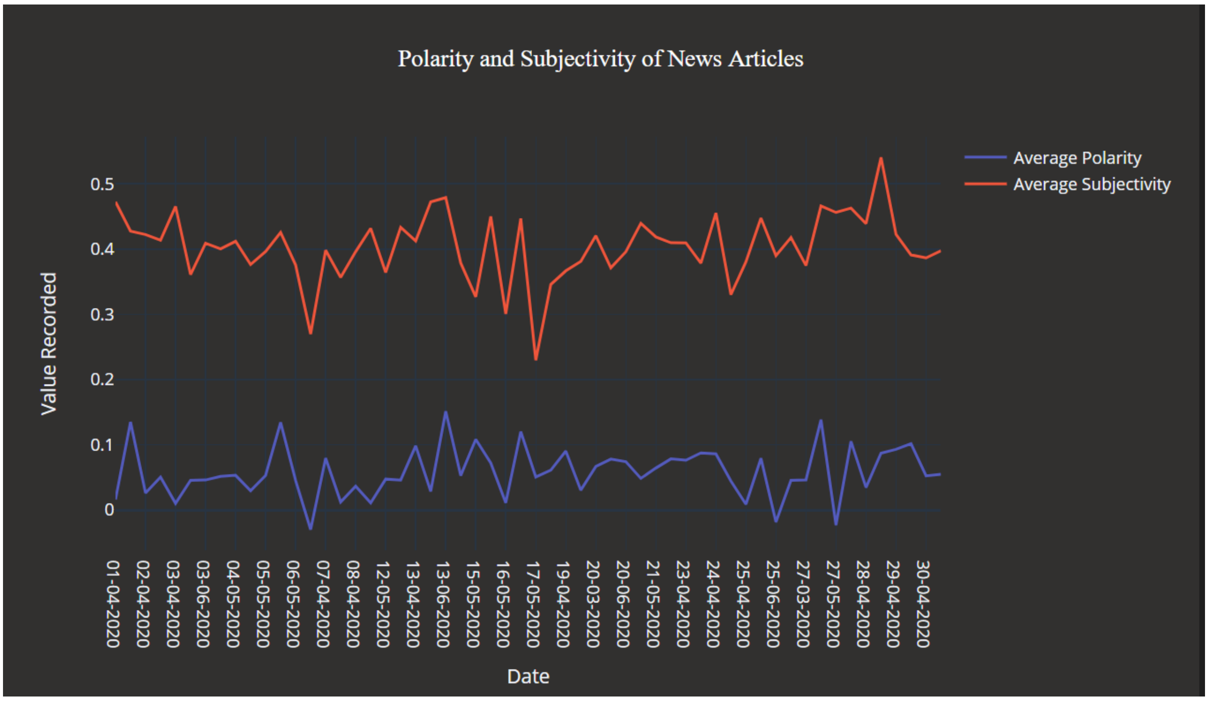
Keeping in mind that there could be more than one sentiment present in a tweet, we computed the Sentiment Density to show that on average how many different sentiments a tweet had on a single day. This figure will give us a direct impression of how much the tweets were “packed with” different emotions 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 level same 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 modeling, 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 trendings of these topics over time are as follows:



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 lockdown3.0, the polarity is maintained at a positive side.

APPLICATIONS

1. The project can be used to analyze 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 role 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 appications 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...
for sent in sentences:
    encoded_sent = tokenizer.encode(
                        sent,  # Sentence to encode.
```

```python
                         add_special_tokens = True,
                               )
  # Add the encoded sentence to the list.
    input_ids.append(encoded_sent)

"""##Padding and Trunking"""
from keras.preprocessing.sequence import pad_sequences
MAX_LEN = 250
# Pad our input tokens with value 0.
input_ids = pad_sequences(input_ids, maxlen=MAX_LEN, dtype="long",
                          value=0, truncating="post", padding="post")
attention_masks = []
# For each sentence...
for sent in input_ids:
    # Create the attention mask.
    att_mask = [int(token_id > 0) for token_id in sent]
    # Store the attention mask for this sentence.
    attention_masks.append(att_mask)

"""##Train/Validation Split"""
from sklearn.model_selection import train_test_split
# Use 90% for training and 10% for validation.
train_inputs,validation_inputs,train_labels,validation_labels=train_test_split(input_ids, labels, random_state=1999, test_size=0.1)
# Do the same for the masks.
train_masks, validation_masks, _, _ = train_test_split(attention_masks, labels,
                              random_state=1999, test_size=0.1)
train_inputs = torch.tensor(train_inputs)
validation_inputs = torch.tensor(validation_inputs)
train_labels = torch.tensor(train_labels)
validation_labels = torch.tensor(validation_labels)
train_masks = torch.tensor(train_masks)
validation_masks = torch.tensor(validation_masks)
from torch.utils.data import TensorDataset, DataLoader, RandomSampler, SequentialSampler
batch_size = 16
# Create the DataLoader for our training set.
train_data = TensorDataset(train_inputs, train_masks, train_labels)
train_sampler = RandomSampler(train_data)
train_dataloader = DataLoader(train_data, sampler=train_sampler, batch_size=batch_size)
```

```
# Create the DataLoader for our validation set.
validation_data = TensorDataset(validation_inputs, validation_masks, validation_labels)
validation_sampler = SequentialSampler(validation_data)
validation_dataloader = DataLoader(validation_data, sampler=validation_sampler, batch_size=batch_size)
from transformers import BertForSequenceClassification, AdamW, BertConfig
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels = 2,
    output_attentions = False,
    output_hidden_states = False, # Whether the model returns all hidden-states.
model.cuda()

"""##Set Optimizer and Learning Rate"""
optimizer = AdamW(model.parameters(),
                  lr = 5e-5, # args.learning_rate
                  eps = 1e-8 # args.adam_epsilon
                )
from transformers import get_linear_schedule_with_warmup
epochs = 4
total_steps = len(train_dataloader) * epochs
# Create the learning rate scheduler.
scheduler = get_linear_schedule_with_warmup(optimizer,
                       num_warmup_steps = 0, # Default value in run_glue.py
                       num_training_steps = total_steps)
import numpy as np
def flat_accuracy(preds, labels):
    pred_flat = np.argmax(preds, axis=1).flatten()
    labels_flat = labels.flatten()
    return np.sum(pred_flat == labels_flat) / len(labels_flat)
import time
import datetime
def format_time(elapsed):
    '''
    Takes a time in seconds and returns a string hh:mm:ss
    '''
    # Round to the nearest second.
    elapsed_rounded = int(round((elapsed)))
    # Format as hh:mm:ss
    return str(datetime.timedelta(seconds=elapsed_rounded))
```

```python

import random
seed_val = 42
random.seed(seed_val)
np.random.seed(seed_val)
torch.manual_seed(seed_val)
torch.cuda.manual_seed_all(seed_val)
loss_values = []

# For each epoch...
for epoch_i in range(0, epochs):

    print("")
    print('======== Epoch {:} / {:} ========'.format(epoch_i + 1, epochs))
    print('Training...')
    # Measure how long the training epoch takes.
    t0 = time.time()
    # Reset the total loss for this epoch.
    total_loss = 0
    model.train()
    # For each batch of training data...
    for step, batch in enumerate(train_dataloader):

        # Progress update every 40 batches.
        if step % 40 == 0 and not step == 0:
            # Calculate elapsed time in minutes.
            elapsed = format_time(time.time() - t0)

            # Report progress.
            print('  Batch {:>5,}  of  {:>5,}.    Elapsed: {:}.'.format(step, len(train_dataloader), elapsed))
        b_input_ids = batch[0].to(device)
        b_input_mask = batch[1].to(device)
        b_labels = batch[2].to(device)
        model.zero_grad()
        outputs = model(b_input_ids,
                    token_type_ids=None,
                    attention_mask=b_input_mask,
                    labels=b_labels)

```

```
        loss = outputs[0]
        total_loss += loss.item()
        loss.backward()
torch.nn.utils.clip_grad_norm_(model.parameters(), 1.0)
        optimizer.step()
        scheduler.step()
    avg_train_loss = total_loss / len(train_dataloader)
    loss_values.append(avg_train_loss)

    print("")
    print("  Average training loss: {0:.3f}".format(avg_train_loss))
    print("  Training epcoh took: {:}".format(format_time(time.time() - t0)))
    print("")
    print("Running Validation...")

    t0 = time.time()
    # during evaluation.
    model.eval()

    # Tracking variables
    eval_loss, eval_accuracy = 0, 0
    nb_eval_steps, nb_eval_examples = 0, 0

    # Evaluate data for one epoch
    for batch in validation_dataloader:
        batch = tuple(t.to(device) for t in batch)

        # Unpack the inputs from our dataloader
        b_input_ids, b_input_mask, b_labels = batch
        with torch.no_grad():
            outputs = model(b_input_ids,
                            token_type_ids=None,
                            attention_mask=b_input_mask)
        logits = outputs[0]
        # Move logits and labels to CPU
        logits = logits.detach().cpu().numpy()
        label_ids = b_labels.to('cpu').numpy()
        # Calculate the accuracy for this batch of test sentences.
        tmp_eval_accuracy = flat_accuracy(logits, label_ids)
        # Accumulate the total accuracy.
```

```
        eval_accuracy += tmp_eval_accuracy
        # Track the number of batches
        nb_eval_steps += 1
               print("  Accuracy: {0:.3f}".format(eval_accuracy/nb_eval_steps))
    print("  Validation took: {:}".format(format_time(time.time() - t0)))
print("Training complete!")
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(style='darkgrid')
sns.set(font_scale=1.5)
plt.rcParams["figure.figsize"] = (12,6)
plt.plot(loss_values, 'b-o')
plt.title("Training loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.show()
```

The complete code is available on [github.]()