SUMMER INTERNSHIP REPORT

ON

Location Based Twitter Sentimental Analysis Extraction for COVID-19

Submitted in partial fulfilment

of the requirements for the award

of degree of

BACHELOR OF TECHNOLOGY

GEO INFORMATICS ENGINEERING

BY

Ritik Tomar

SAP ID 500068081

School of Engineering

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES
Dehradun-248007

June-July, 2021

Candidate's Declaration

I hereby certify that the project work entitled, **"Location Based Twitter Sentimental Analysis Extraction for COVID-19"** in the partial fulfilment of the requirements for the award of the degree of ***Bachelor of Technology*** in ***Geo Informatics Engineering*** and submitted to the Department of Petroleum Engineering and Earth Sciences at School of Engineering, University of Petroleum and Energy Studies, Dehradun, is an authentic record of our work carried out during a period from June to July, 2021 under the supervision of **Mr. Suman Das Gupta** (Principal Consultant (GIS COE) at Infosys) and mentorship of **Mr. Anupam Kumar (**Principal Consultant at Infosys).

The matter presented in this project has not been submitted by us for the award of any other degree of this or any other university.

Ritik Tomar

SAP ID 500068081

This is to certify that above statements made by the candidate is correct to the best of my knowledge.

Date: 23 July 2021

Mr. Suman Das Gupta	**Mr. Anupam Kumar**
Principal Consultant (GIS COE)	**Principal Consultant**
Infosys	**Infosys**

ACKNOWLEDGEMENT

I have taken efforts in this project work entitled, **“Location Based Twitter Sentimental Analysis Extraction for COVID-19”.** However, it would not have been possible without the kind of support and help of many individuals. I would like to extend my sincere thanks to all of them.

I am highly indebted to Mr. Anupam Kumar, (Principal Consultant at Infosys) for their guidance and constant supervision as well as for providing necessary information regarding the project and also for their support in completing the project. I would like to express my gratitude towards my parents for their kind co-operation and encouragement which helped me in completion of this project.

I would like to extend a formal note of thanks to Mr. Suman Das Gupta, Principal Consultant (GIS COE) at Infosys.

Ritik Tomar

500068081

R620218014

Table of Contents

<u>Abstract</u>

In this age of rapidly advancing technology, social media has evolved into a strong forum for people to express their worries and opinions. Twitter is one example of such a platform. In recent years, Twitter has become a popular microblogging tool. Sentiment Analysis is highly valuable in social media monitoring in this case since it helps us to get a broad sense of public opinion on certain issues. The applications of sentiment analysis have grown in recent years as enterprises and governments throughout the world have begun to utilise the capacity to extract insights from social data. There has been a clear inference that variations in social media opinion connect with fluctuations in a country's finances as well as public perception.

Following the current COVID-19 outbreak, there has been a significant shift in public opinion in India toward government policies and actions. Studying public opinion on the epidemic and government initiatives is critical because it serves as a sanity check on the success of the measures that have been implemented. This research also sheds light on the business strategies that must be embraced in this post-COVID-19 era, in which people's attitudes have shifted dramatically. The issue statement ‘Sentiment Analysis of COVID-19 Tweets' is crucial in this context.

Keywords: COVID-19, Sentiment, Twitter, Tweets, Sentiment Analysis.

Problem Statement

The coronavirus epidemic is now affecting the whole planet. However, a new problem has emerged in tandem with the virus, one that is just as dangerous as the infection itself. That is, there is a big flow of information about the virus in the form of tweets, blogs, and news, and there is too little analysis of this massive volume of data pouring in and out of the system at all times. This information might be false, mixed news, or simply someone's perspective on the epidemic. The propagation of such incomplete, wrong information would produce mass panic and terror among the people, hence the urgent need is to address and better comprehend the pandemic's communication problem. Understanding the right sentiment and opinions of the people towards the pandemic plays a crucial role in policymaking and creating appropriate business models that are of necessity to the people. The government, companies, and many organisations form their decisions to cater to the needs of the people, and thus understanding the right sentiment and opinions of the people towards the pandemic plays a crucial role in policymaking and creating appropriate business models that are of necessity to the people.

Proposed Solution

The exponential rise in people's use of social media to express their opinions, perspectives, and as a source of news rather than traditional news in the last decade has placed a premium on the use of Deep Learning and Artificial Intelligence methods in gauging information from these sites to analyse and extract sentiments that serve as valuable sources of insights for corporate companies. The study of a Population Sentiment Tweets that displays the sentiment trend among the Indian public towards the epidemic is thus the major goal of this study. Twitter is a microblogging service that has grown in popularity in recent years as a venue for governments, organisations, and individuals to make official announcements, express emotions, and voice opinions on current, ongoing events. As a result, the data set for this research is comprised of tweets, which are brief messages made on this site. The use of Deep Learning algorithms to evaluate these tweets and determine their attitudes.

A sentiment extraction model that analyses this data on a big scale and gives useful information about public sentiment patterns. Recent language models such as BERT, XLNET, T5, Roberta, and Electra have demonstrated excellent contextual language recognition. As a result, a transfer learning strategy based on these models was employed. A caveat that came up in the context of COVID-19 is that a few terms, such as the term "positive," are used to express a negative mood when they are used in a regular context. Instances like this were investigated, and it was discovered after some investigation that these terms were not as commonly used out of context as had been imagined. Another fascinating difficulty that was tackled was the process of extracting public perceptions of the Indian government's and commercial corporations' COVID-19 initiatives. Language models capable of simultaneous sentiment categorization and extraction were created to help overcome these obstacles during sentiment analysis. Furthermore, the issue description necessitates a user-friendly graphical user interface that delivers valuable insights into public opinion on government and commercial actions during COVID. Thus, the sentiment extraction methodology and post prediction analysis have made it easier to extract the climate of opinion around the COVID-19 pandemic in this setting.

Softwares and Tools Used

1. **Google Colab**: Google Research's Colaboratory, or "Colab" for short, is a product. Colab is a web-based Python editor that allows anybody to create and run arbitrary Python code. It's notably useful for machine learning, data analysis, and teaching. Colab is a hosted Jupyter notebook service that doesn't require any setup and offers free access to computational resources, including GPUs.

2. **Python Programming Language**: Python is a dynamically semantic, interpreted, object-oriented high-level programming language. Its high-level built-in data structures, along with dynamic typing and dynamic binding, making it ideal for Rapid Application Development and as a scripting or glue language for connecting existing components.

3. **ArcGIS Online**: ArcGIS Online is a cloud-based mapping and analysis solution. Use it to make maps, analyze data, and to share and collaborate. Get access to workflow-specific apps, maps and data from around the Globe and tools for being mobile in the field. Your data and maps are stored in a secure and private infrastructure and can be configured to meet your mapping and IT requirements.
   In this project we have put ArcGIS in use for the creation of Web map. This web map consists of mainly four layers namely Leak Survey Area, Breadcrumb, Gas Leak and Pipe (Gas Main). All the layers are extracted on ArcGIS platform and simultaneously converted into hosted layers. This map with hosted layer is then published under the property of full edit so that only the owner can make required edits in the map.

Methodology

Initially, the IEEE Coronavirus (COVID-19) Tweets Data set was downloaded from their website. Upon inspection, it was found that many tweets did not have geo-location tags, and also many were in different languages apart from English. Due to this challenge in obtaining proper data, a new data set named Geo-Tagged Coronavirus (COVID-19) Tweets Data set was obtained from the same website. These tweets were then hydrated using a few python commands. Then, tweets in "English" and tweets from "India" were randomly chosen and a new dataset was created. Further, other data set containing COVID-19 related tweets from India were obtained from Kaggle. This data set was then cleaned and normalized to make it useful for further analysis.
Link to Notebook.

This cleaned data were then subjected to further analysis by extracting bigrams, trigrams, and plotting frequency bar graphs, Word Clouds, Relationship Nexus, Boxplots, etc. This was done using Python. Some Interactive Plots were also plotted. This complete process is termed as Exploratory Data Analysis.
Link to Notebook.

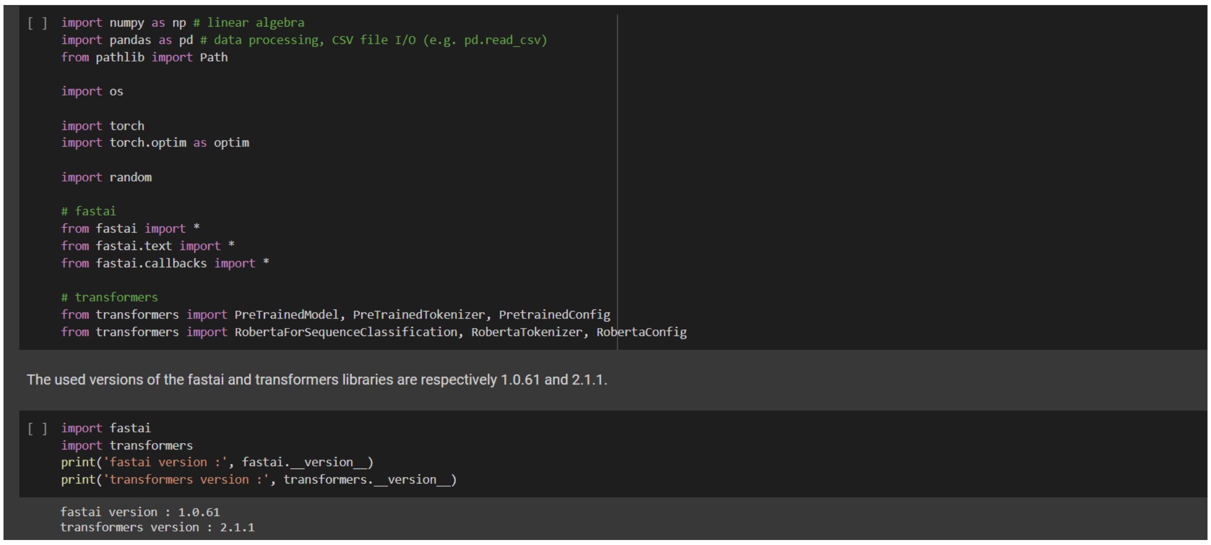
After Exploratory Data Analysis is completed, the tweets are then Tokenized and are made in a format suitable for the Language Model. In this Step, two models were used:

Roberta Model Training: Transfer learning methods were implemented to carry out sentiment analysis. Sentiment Analysis of Tweets was carried out by integrating and using both the Huggingface Transformer Library and FastAI. Further Slanted Triangular Learning Rates, Discriminate Learning Rate and even Gradual Unfreezing were used, as a result of which, state-of-the-art results were obtained rapidly without even tuning the parameters. The tokenized data was then passed through the RoBERTa model to perform Sentiment Analysis. This yielded a model with an accuracy of 91% over the data set. The Tweepy API was used to scrap tweets in real-time which were then passed through the model to obtain the sentiments.

1. **Transfer Learning**: Transfer learning (TL) is a machine learning (ML) research issue that focuses on storing and transferring information obtained while addressing one problem to a different but related problem. For example, the skills learned when learning to recognise automobiles may be applied to recognising trucks. Although formal links between the two areas are minimal, this area of research has some ties to the lengthy history of psychological literature on learning transfer. Reusing or transferring knowledge from previously learned tasks for the learning of new tasks has the potential to greatly increase the sampling efficiency of a reinforcement learning agent from a practical viewpoint.

2. **Transformers**: Hugging Face's major library is Transformers. It allows you to create, train, and fine-tune transformers using easy and highly abstracted capabilities. It includes almost 10,000 pre-trained models, which can be accessible on the Hub.

3. **PyTorch**: PyTorch is a Facebook-developed and-maintained open-source Python library for deep learning.



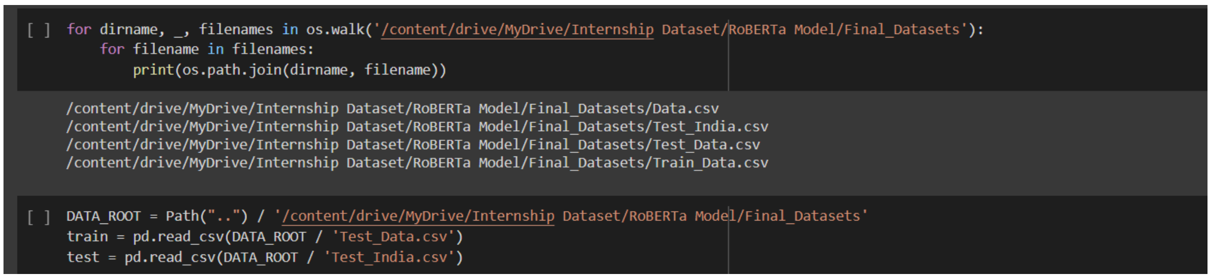
```
[ ] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
    from pathlib import Path

    import os

    import torch
    import torch.optim as optim

    import random

    # fastai
    from fastai import *
    from fastai.text import *
    from fastai.callbacks import *

    # transformers
    from transformers import PreTrainedModel, PreTrainedTokenizer, PretrainedConfig
    from transformers import RobertaForSequenceClassification, RobertaTokenizer, RobertaConfig
```

The used versions of the fastai and transformers libraries are respectively 1.0.61 and 2.1.1.

```
[ ] import fastai
    import transformers
    print('fastai version :', fastai.__version__)
    print('transformers version :', transformers.__version__)

    fastai version : 1.0.61
    transformers version : 2.1.1
```

Fig- Import Modules

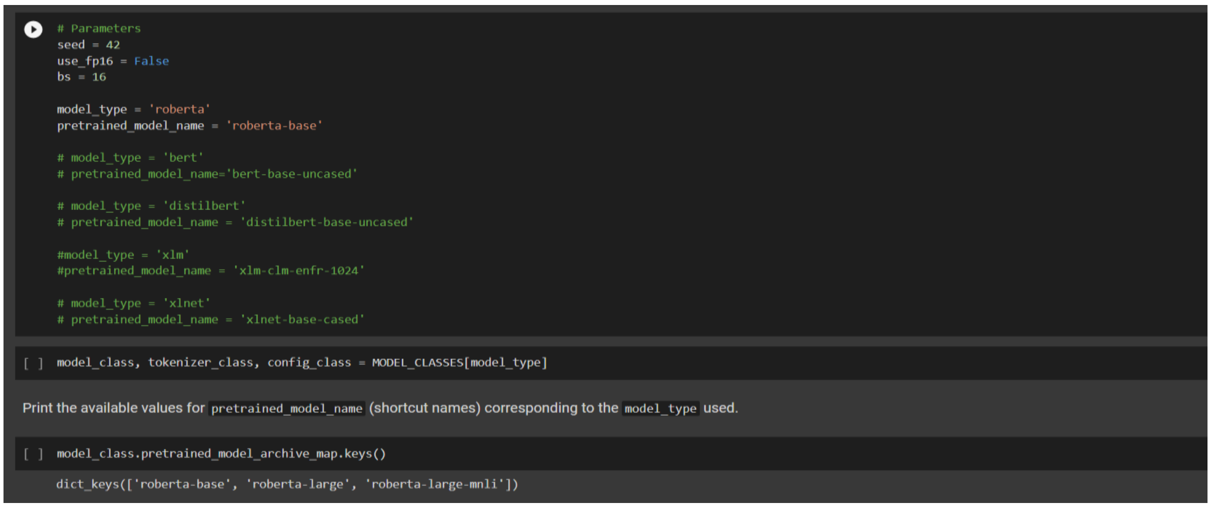
- For each tweet, the model has to predict a label for the sentiment. We evaluate the outputs of the model on classification accuracy. The sentiment labels are:
  0→ Negative
  1→ Neutral
  2→ Positive



```
[ ] for dirname, _, filenames in os.walk('/content/drive/MyDrive/Internship Dataset/RoBERTa Model/Final_Datasets'):
        for filename in filenames:
            print(os.path.join(dirname, filename))

    /content/drive/MyDrive/Internship Dataset/RoBERTa Model/Final_Datasets/Data.csv
    /content/drive/MyDrive/Internship Dataset/RoBERTa Model/Final_Datasets/Test_India.csv
    /content/drive/MyDrive/Internship Dataset/RoBERTa Model/Final_Datasets/Test_Data.csv
    /content/drive/MyDrive/Internship Dataset/RoBERTa Model/Final_Datasets/Train_Data.csv

[ ] DATA_ROOT = Path("..") / '/content/drive/MyDrive/Internship Dataset/RoBERTa Model/Final_Datasets'
    train = pd.read_csv(DATA_ROOT / 'Test_Data.csv')
    test = pd.read_csv(DATA_ROOT / 'Test_India.csv')
```

Fig- Specifying Train Test Data

- In transformers, each model architecture is associated with 3 main types of classes:
  I. A *model class* to load/store a particular pre-train model.
  II. A *tokenizer class* to pre-process the data and make it compatible with a particular model.
  III. A *configuration class* to load/store the configuration of a particular model.



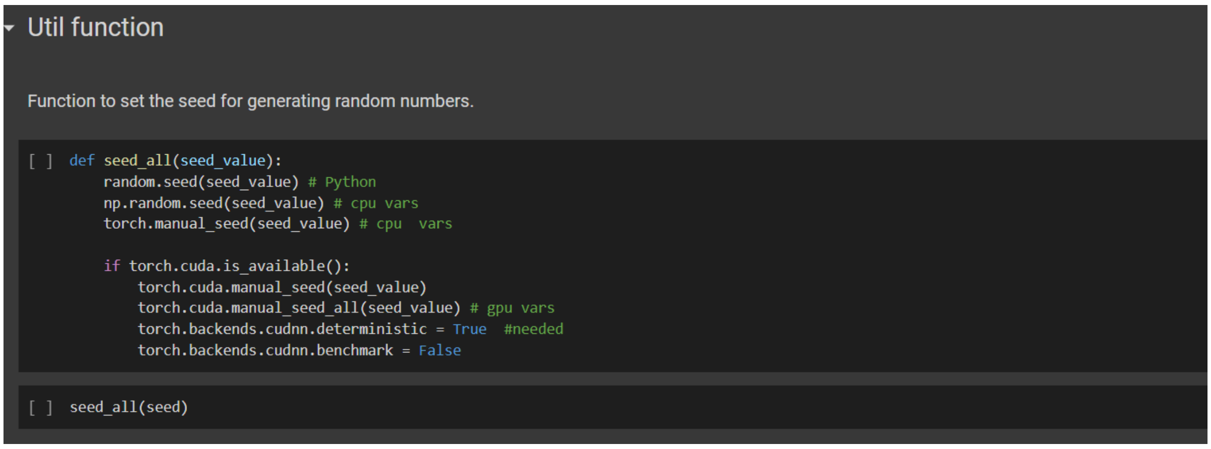
```
# Parameters
seed = 42
use_fp16 = False
bs = 16

model_type = 'roberta'
pretrained_model_name = 'roberta-base'

# model_type = 'bert'
# pretrained_model_name='bert-base-uncased'

# model_type = 'distilbert'
# pretrained_model_name = 'distilbert-base-uncased'

#model_type = 'xlm'
#pretrained_model_name = 'xlm-clm-enfr-1024'

# model_type = 'xlnet'
# pretrained_model_name = 'xlnet-base-cased'
```

```
[ ] model_class, tokenizer_class, config_class = MODEL_CLASSES[model_type]
```

Print the available values for `pretrained_model_name` (shortcut names) corresponding to the `model_type` used.

```
[ ] model_class.pretrained_model_archive_map.keys()

    dict_keys(['roberta-base', 'roberta-large', 'roberta-large-mnli'])
```

Fig- Extracting Information about pretrained model



Util function

Function to set the seed for generating random numbers.

```
[ ] def seed_all(seed_value):
        random.seed(seed_value) # Python
        np.random.seed(seed_value) # cpu vars
        torch.manual_seed(seed_value) # cpu  vars

        if torch.cuda.is_available():
            torch.cuda.manual_seed(seed_value)
            torch.cuda.manual_seed_all(seed_value) # gpu vars
            torch.backends.cudnn.deterministic = True  #needed
            torch.backends.cudnn.benchmark = False

[ ] seed_all(seed)
```

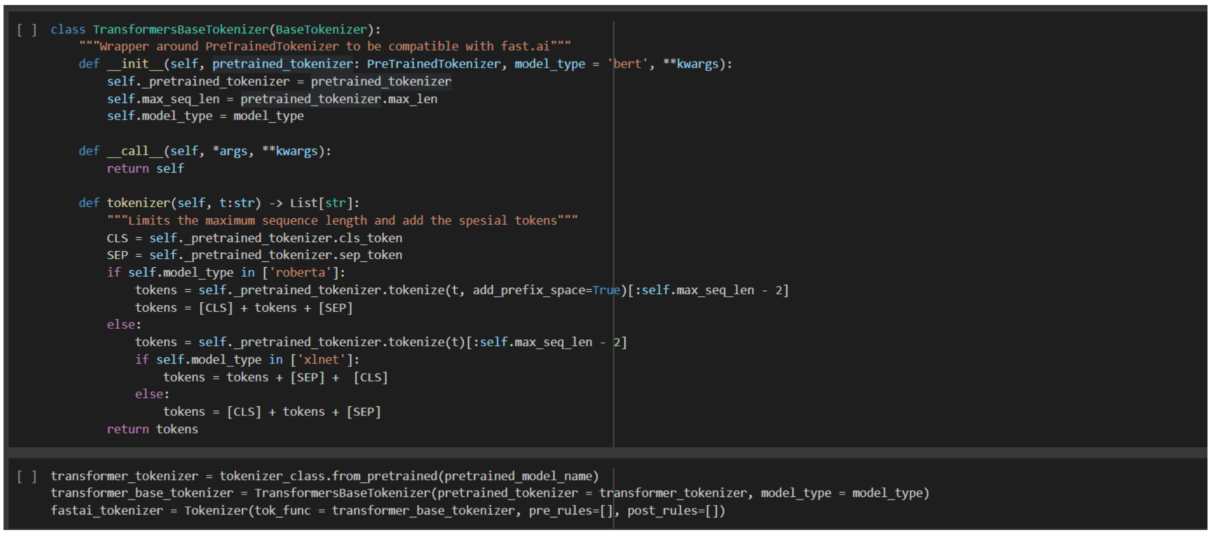
Fig- Defining Util Function

Data Pre Processing:

We must structure the model input sequence in a specified way to meet pre-training. To do so, we must appropriately tokenize and numericalize the sentences first. The problem is that each fine-tuned pre-trained model requires the exact same pre-process - tokenization and numericalization - as the pre-train component. Fortunately, the transformers tokenizer class offers the appropriate pre-processing tools for each pre-trained model.

Data pre-processing is done automatically during the construction of the DataBunch in the fastai package. As you will see in the DataBunch implementation, the tokenizer and numericalizer are passed in the processor argument under the following format :

Processor = [TokenizeProcessor(tokenizer=tokenizer,...), NumericalizeProcessor(vocab=vocab,...)]

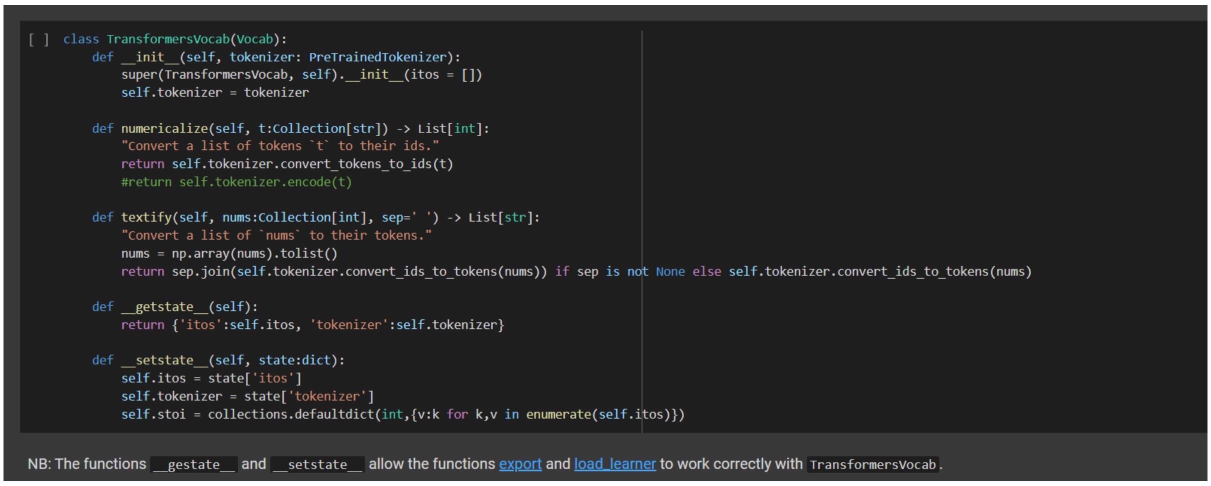


```
[ ] class TransformersBaseTokenizer(BaseTokenizer):
        """Wrapper around PreTrainedTokenizer to be compatible with fast.ai"""
        def __init__(self, pretrained_tokenizer: PreTrainedTokenizer, model_type = 'bert', **kwargs):
            self._pretrained_tokenizer = pretrained_tokenizer
            self.max_seq_len = pretrained_tokenizer.max_len
            self.model_type = model_type

        def __call__(self, *args, **kwargs):
            return self

        def tokenizer(self, t:str) -> List[str]:
            """Limits the maximum sequence length and add the spesial tokens"""
            CLS = self._pretrained_tokenizer.cls_token
            SEP = self._pretrained_tokenizer.sep_token
            if self.model_type in ['roberta']:
                tokens = self._pretrained_tokenizer.tokenize(t, add_prefix_space=True)[:self.max_seq_len - 2]
                tokens = [CLS] + tokens + [SEP]
            else:
                tokens = self._pretrained_tokenizer.tokenize(t)[:self.max_seq_len - 2]
                if self.model_type in ['xlnet']:
                    tokens = tokens + [SEP] +  [CLS]
                else:
                    tokens = [CLS] + tokens + [SEP]
            return tokens
```

```
[ ] transformer_tokenizer = tokenizer_class.from_pretrained(pretrained_model_name)
    transformer_base_tokenizer = TransformersBaseTokenizer(pretrained_tokenizer = transformer_tokenizer, model_type = model_type)
    fastai_tokenizer = Tokenizer(tok_func = transformer_base_tokenizer, pre_rules=[], post_rules=[])
```

Fig- Defining function for Custom Tokenizer

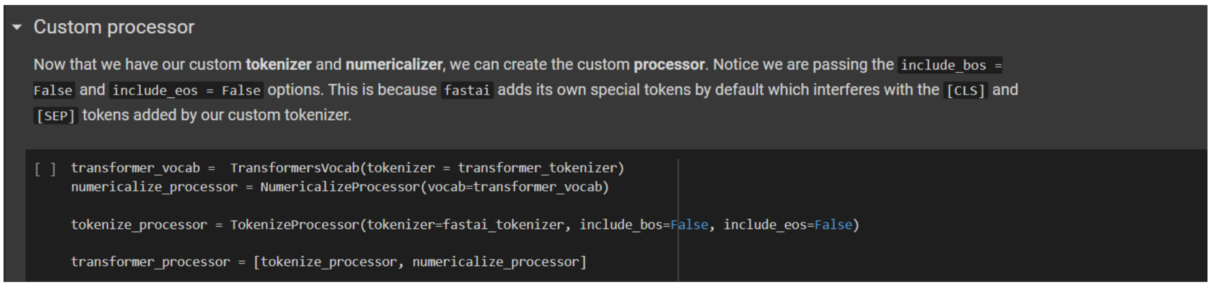
- Three factors were taken into consideration throughout this implementation:
    I. We must limit the sequence length to the model input size because we are not utilising RNN.
    II. The majority of the models necessitate the use of special tokens at the start and conclusion of the sequences.
    III. Some models, such as RoBERTa, need a space at the beginning of the input string. The encoding methods for such models should be called with add prefix space set to True.



```
[ ] class TransformersVocab(Vocab):
        def __init__(self, tokenizer: PreTrainedTokenizer):
            super(TransformersVocab, self).__init__(itos = [])
            self.tokenizer = tokenizer

        def numericalize(self, t:Collection[str]) -> List[int]:
            "Convert a list of tokens `t` to their ids."
            return self.tokenizer.convert_tokens_to_ids(t)
            #return self.tokenizer.encode(t)

        def textify(self, nums:Collection[int], sep=' ') -> List[str]:
            "Convert a list of `nums` to their tokens."
            nums = np.array(nums).tolist()
            return sep.join(self.tokenizer.convert_ids_to_tokens(nums)) if sep is not None else self.tokenizer.convert_ids_to_tokens(nums)

        def __getstate__(self):
            return {'itos':self.itos, 'tokenizer':self.tokenizer}

        def __setstate__(self, state:dict):
            self.itos = state['itos']
            self.tokenizer = state['tokenizer']
            self.stoi = collections.defaultdict(int,{v:k for k,v in enumerate(self.itos)})
```

NB: The functions `__gestate__` and `__setstate__` allow the functions export and load_learner to work correctly with `TransformersVocab`.

Fig- Defining function for Custom Numericalizer



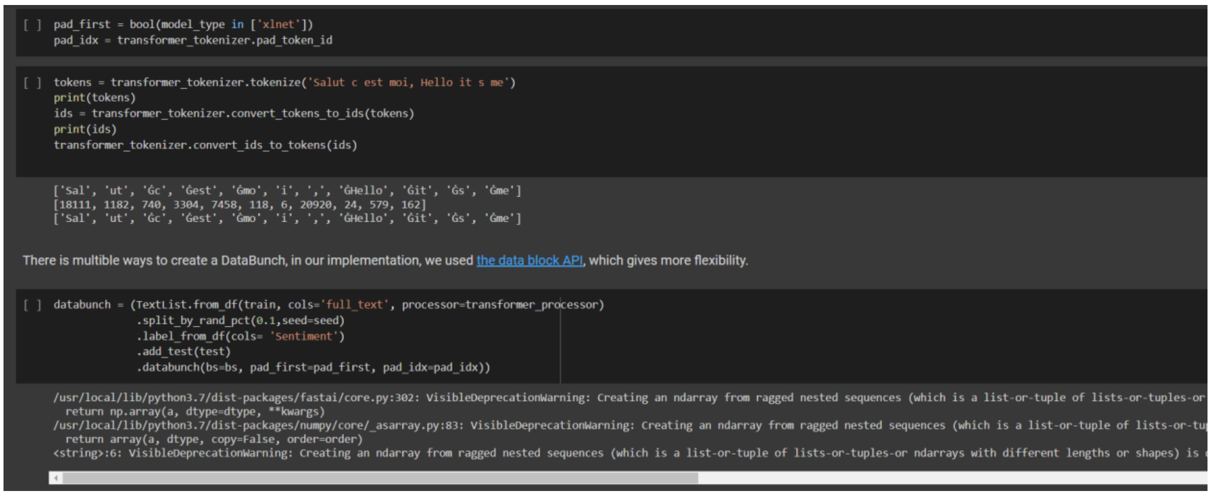
Custom processor

Now that we have our custom **tokenizer** and **numericalizer**, we can create the custom **processor**. Notice we are passing the `include_bos = False` and `include_eos = False` options. This is because `fastai` adds its own special tokens by default which interferes with the `[CLS]` and `[SEP]` tokens added by our custom tokenizer.

```
[ ] transformer_vocab =  TransformersVocab(tokenizer = transformer_tokenizer)
    numericalize_processor = NumericalizeProcessor(vocab=transformer_vocab)

    tokenize_processor = TokenizeProcessor(tokenizer=fastai_tokenizer, include_bos=False, include_eos=False)

    transformer_processor = [tokenize_processor, numericalize_processor]
```

Fig- Custom Processor

- For the DataBunch creation, you have to pay attention to set the processor argument to our new custom processor transformer_processor and manage correctly the padding. As mentioned in the HuggingFace documentation RoBERTa model is with absolute position embeddings, so it's usually advised to pad the inputs on the right rather than the left.



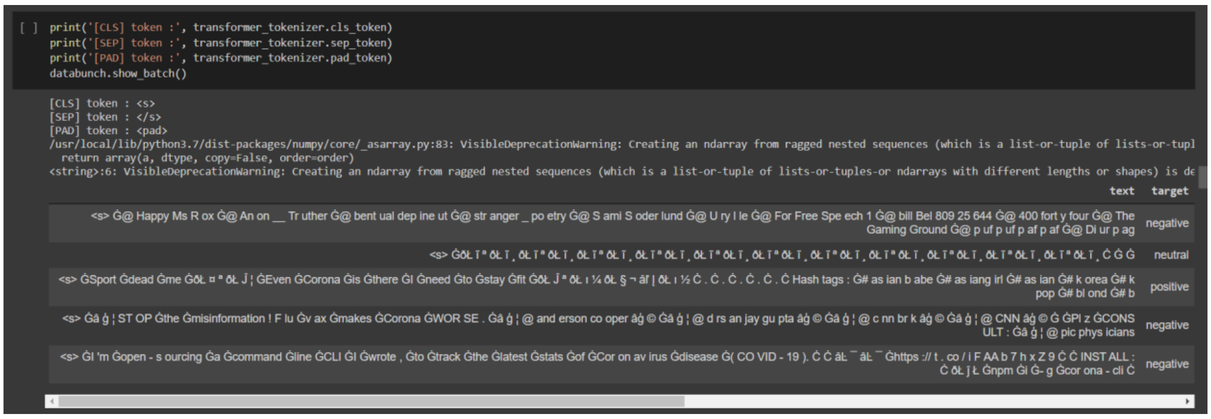
```
[ ] pad_first = bool(model_type in ['xlnet'])
    pad_idx = transformer_tokenizer.pad_token_id
```

```
[ ] tokens = transformer_tokenizer.tokenize('Salut c est moi, Hello it s me')
    print(tokens)
    ids = transformer_tokenizer.convert_tokens_to_ids(tokens)
    print(ids)
    transformer_tokenizer.convert_ids_to_tokens(ids)

    ['Sal', 'ut', 'Ġc', 'Ġest', 'Ġmo', 'i', ',', 'ĠHello', 'Ġit', 'Ġs', 'Ġme']
    [18111, 1182, 740, 3304, 7458, 118, 6, 20920, 24, 579, 162]
    ['Sal', 'ut', 'Ġc', 'Ġest', 'Ġmo', 'i', ',', 'ĠHello', 'Ġit', 'Ġs', 'Ġme']
```

There is multible ways to create a DataBunch, in our implementation, we used the data block API, which gives more flexibility.

```
[ ] databunch = (TextList.from_df(train, cols='full_text', processor=transformer_processor)
                 .split_by_rand_pct(0.1,seed=seed)
                 .label_from_df(cols= 'Sentiment')
                 .add_test(test)
                 .databunch(bs=bs, pad_first=pad_first, pad_idx=pad_idx))

    /usr/local/lib/python3.7/dist-packages/fastai/core.py:302: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or
      return np.array(a, dtype=dtype, **kwargs)
    /usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tu
      return array(a, dtype, copy=False, order=order)
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is
```

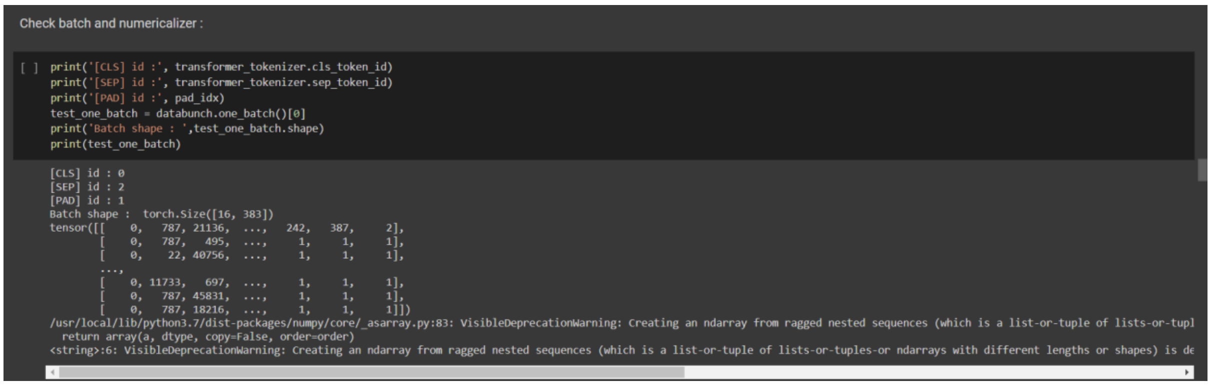
Fig- Setting up the Data Bunch



```
[ ] print('[CLS] token :', transformer_tokenizer.cls_token)
    print('[SEP] token :', transformer_tokenizer.sep_token)
    print('[PAD] token :', transformer_tokenizer.pad_token)
    databunch.show_batch()

    [CLS] token : <s>
    [SEP] token : </s>
    [PAD] token : <pad>
    /usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
      return array(a, dtype, copy=False, order=order)
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de
```

text	target
<s> Ġ@ Happy Ms R ox Ġ@ An on __ Tr uther Ġ@ bent ual dep ine ut Ġ@ str anger _ po etry Ġ@ S ami S oder lund Ġ@ U ry l le Ġ@ For Free Spe ech 1 Ġ@ bill Bel 809 25 644 Ġ@ 400 fort y four Ġ@ The Gaming Ground Ġ@ p uf p uf p af p af Ġ@ Di ur p ag	negative
<s> ĠðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ ðŁ Ĩ ª ðŁ Ĩ ¸ Ċ Ġ Ġ	neutral
<s> ĠSport Ġdead Ġme ĠðŁ ¤ ª ðŁ Ĵ ¦ ĠEven ĠCorona Ġis Ġthere ĠI Ġneed Ġto Ġstay Ġfit ĠðŁ Ĵ ª ðŁ ı ¼ ðŁ § ¬ âĩ | ðŁ ı ½ Ċ . Ċ . Ċ . Ċ . Ċ . Ċ Hash tags : Ġ# as ian b abe Ġ# as iang irl Ġ# as ian Ġ# k orea Ġ# k pop Ġ# bl ond Ġ# b	positive
<s> Ġâ ģ ¦ ST OP Ġthe Ġmisinformation ! F lu Ġv ax Ġmakes ĠCorona ĠWOR SE . Ġâ ģ ¦ @ and erson co oper âģ © Ġâ ģ ¦ @ d rs an jay gu pta âģ © Ġâ ģ ¦ @ c nn br k âģ © Ġâ ģ ¦ @ CNN âģ © Ġ ĠPl z ĠCONS ULT : Ġâ ģ ¦ @ pic phys icians	negative
<s> ĠI 'm Ġopen - s ourcing Ġa Ġcommand Ġline ĠCLI ĠI Ġwrote , Ġto Ġtrack Ġthe Ġlatest Ġstats Ġof ĠCor on av irus Ġdisease Ġ(CO VID - 19). Ċ Ċ âŁ ¯ âŁ ¯ Ġhttps :// t . co / i F AA b 7 h x Z 9 Ċ Ċ INST ALL : Ċ ðŁ ĵ Ł Ġnpm Ġi Ġ- g Ġcor ona - cli Ċ	negative

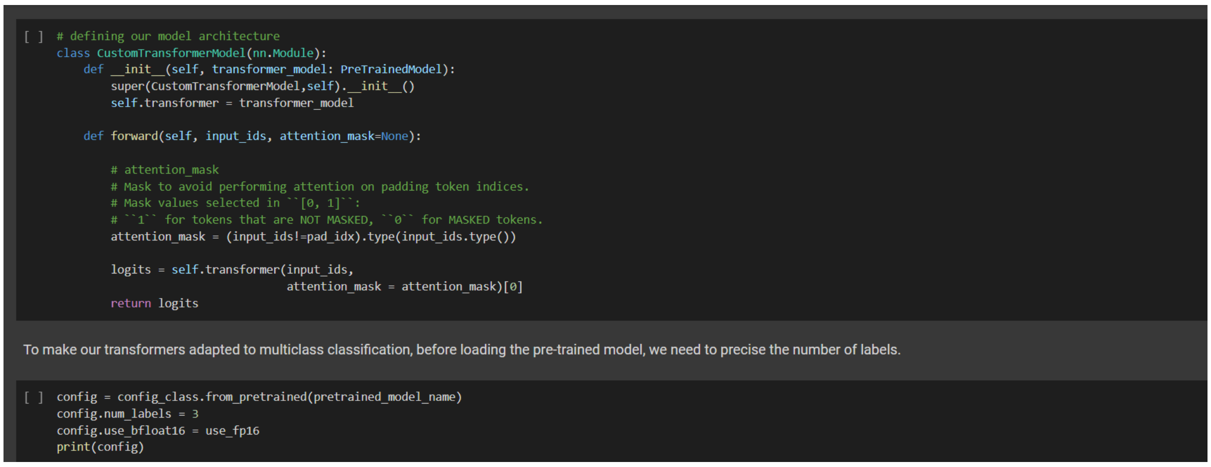
Fig- Check Batch and Tokenizer



Check batch and numericalizer :

```
[ ] print('[CLS] id :', transformer_tokenizer.cls_token_id)
    print('[SEP] id :', transformer_tokenizer.sep_token_id)
    print('[PAD] id :', pad_idx)
    test_one_batch = databunch.one_batch()[0]
    print('Batch shape : ',test_one_batch.shape)
    print(test_one_batch)

    [CLS] id : 0
    [SEP] id : 2
    [PAD] id : 1
    Batch shape :  torch.Size([16, 383])
    tensor([[    0,   787, 21136,  ...,   242,   387,     2],
            [    0,   787,   495,  ...,     1,     1,     1],
            [    0,    22, 40756,  ...,     1,     1,     1],
            ...,
            [    0, 11733,   697,  ...,     1,     1,     1],
            [    0,   787, 45831,  ...,     1,     1,     1],
            [    0,   787, 18216,  ...,     1,     1,     1]])
    /usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
      return array(a, dtype, copy=False, order=order)
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de
```

Fig- Check Batch and Numericalizer



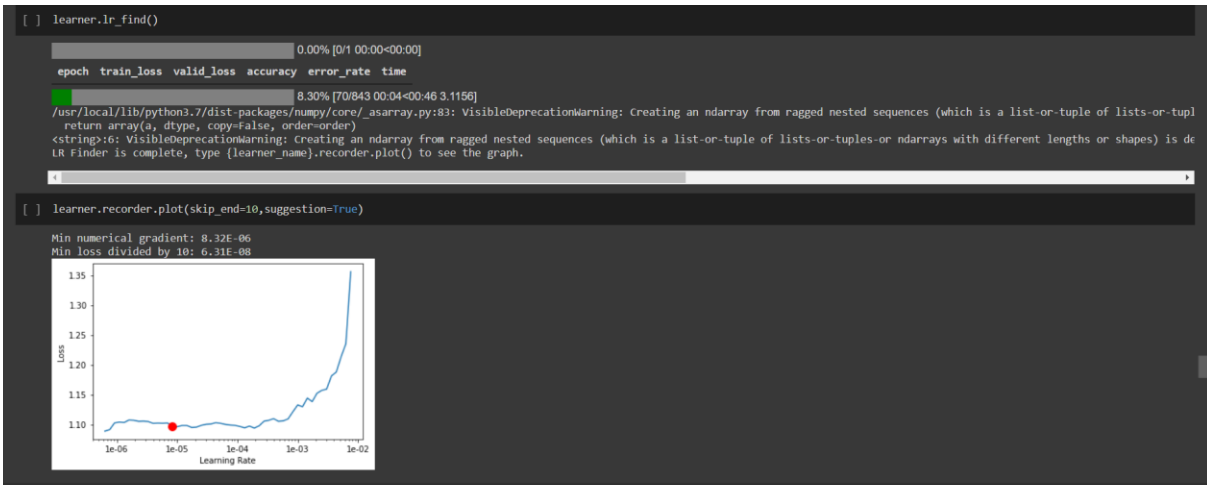
```
[ ] # defining our model architecture
    class CustomTransformerModel(nn.Module):
        def __init__(self, transformer_model: PreTrainedModel):
            super(CustomTransformerModel,self).__init__()
            self.transformer = transformer_model

        def forward(self, input_ids, attention_mask=None):

            # attention_mask
            # Mask to avoid performing attention on padding token indices.
            # Mask values selected in ``[0, 1]``:
            # ``1`` for tokens that are NOT MASKED, ``0`` for MASKED tokens.
            attention_mask = (input_ids!=pad_idx).type(input_ids.type())

            logits = self.transformer(input_ids,
                                      attention_mask = attention_mask)[0]
            return logits
```

To make our transformers adapted to multiclass classification, before loading the pre-trained model, we need to precise the number of labels.

```
[ ] config = config_class.from_pretrained(pretrained_model_name)
    config.num_labels = 3
    config.use_bfloat16 = use_fp16
    print(config)
```

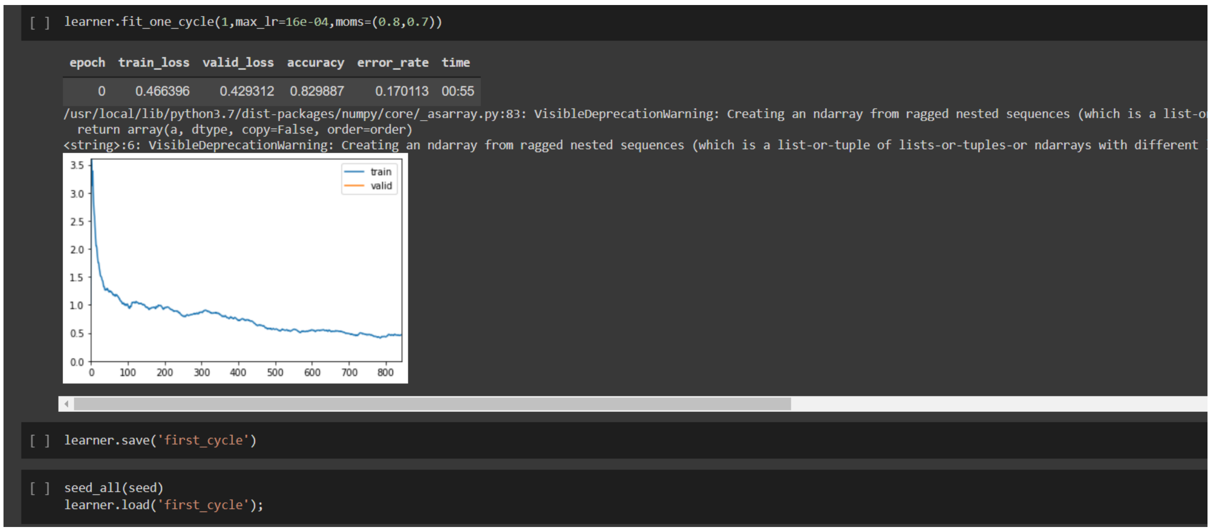
Fig- Defining the model architecture

- Now we have finally used all the fastai build-in features to train our model. Like the ULMFiT method, we have used Slanted Triangular Learning Rates, Discriminate Learning Rate and gradually unfreeze the model.
- For Slanted Triangular Learning Rates we have to use the function one_cycle. To use our one_cycle we will need an optimum learning rate. We can find this learning rate by using a learning rate finder which can be called by using lr_find.



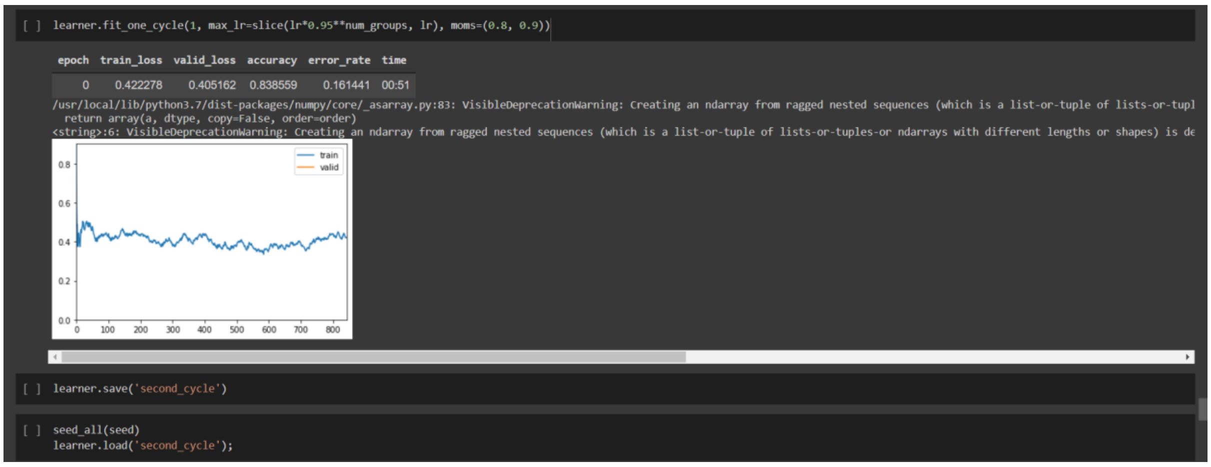
```
[ ] learner.lr_find()

0.00% [0/1 00:00<00:00]
epoch  train_loss  valid_loss  accuracy  error_rate  time

8.30% [70/843 00:04<00:46 3.1156]
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
  return array(a, dtype, copy=False, order=order)
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de
LR Finder is complete, type {learner_name}.recorder.plot() to see the graph.

[ ] learner.recorder.plot(skip_end=10,suggestion=True)

Min numerical gradient: 8.32E-06
Min loss divided by 10: 6.31E-08
```

Fig- Finding the learning rate



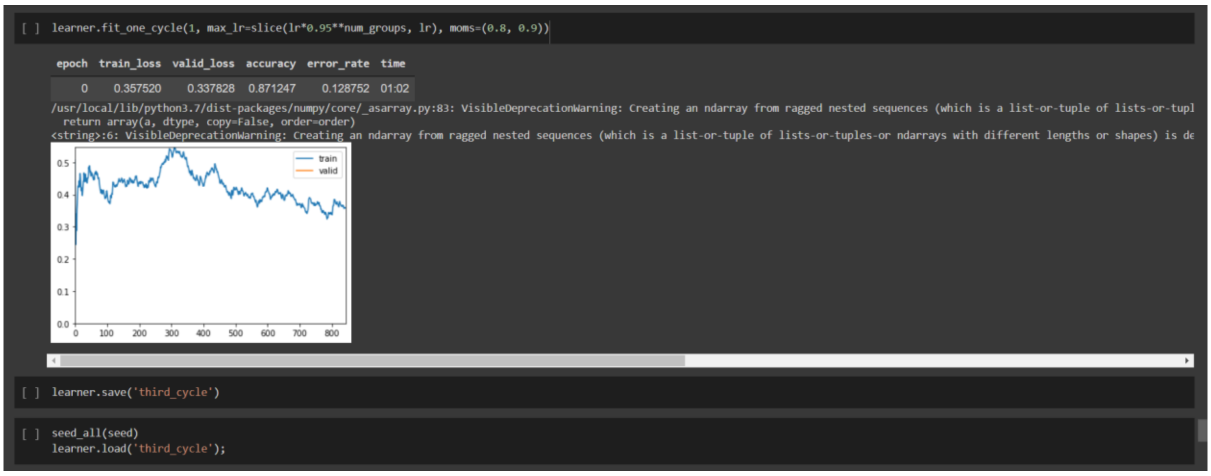
```
[ ] learner.fit_one_cycle(1,max_lr=16e-04,moms=(0.8,0.7))

epoch  train_loss  valid_loss  accuracy  error_rate  time
    0    0.466396    0.429312  0.829887    0.170113  00:55
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-o
  return array(a, dtype, copy=False, order=order)
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different

[ ] learner.save('first_cycle')

[ ] seed_all(seed)
    learner.load('first_cycle');
```

Fig- Plotting one cycle at max_lr=16e-04



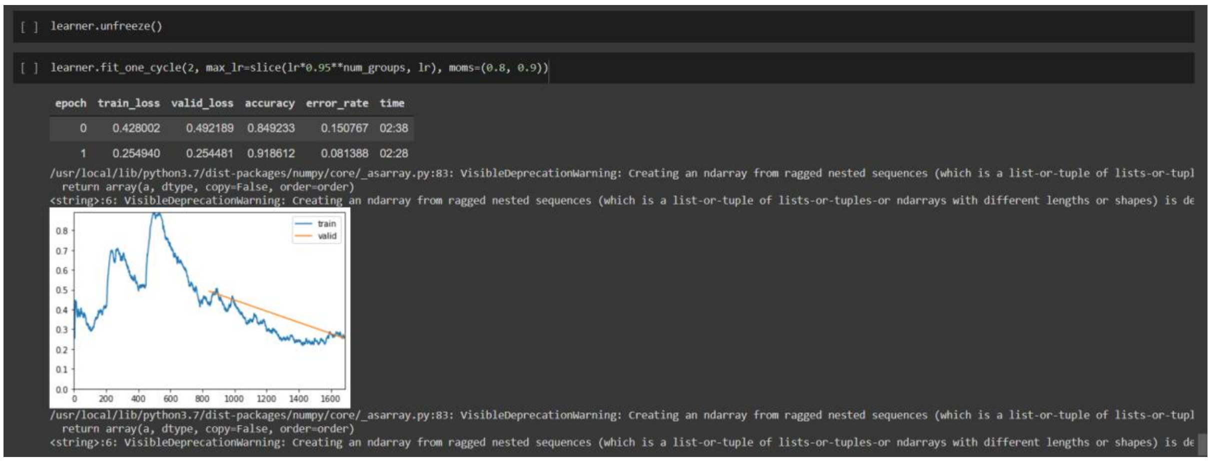
```
[ ] learner.fit_one_cycle(1, max_lr=slice(lr*0.95**num_groups, lr), moms=(0.8, 0.9))

epoch  train_loss  valid_loss  accuracy  error_rate  time
    0    0.422278    0.405162  0.838559    0.161441  00:51
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
  return array(a, dtype, copy=False, order=order)
<string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de

[ ] learner.save('second_cycle')

[ ] seed_all(seed)
    learner.load('second_cycle');
```

Fig- Plotting one cycle by unfreezing second group of layers



```
[ ] learner.fit_one_cycle(1, max_lr=slice(lr*0.95**num_groups, lr), moms=(0.8, 0.9))

    epoch  train_loss  valid_loss  accuracy  error_rate  time
        0    0.357520    0.337828  0.871247    0.128752  01:02
    /usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
      return array(a, dtype, copy=False, order=order)
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de

[ ] learner.save('third_cycle')

[ ] seed_all(seed)
    learner.load('third_cycle');
```

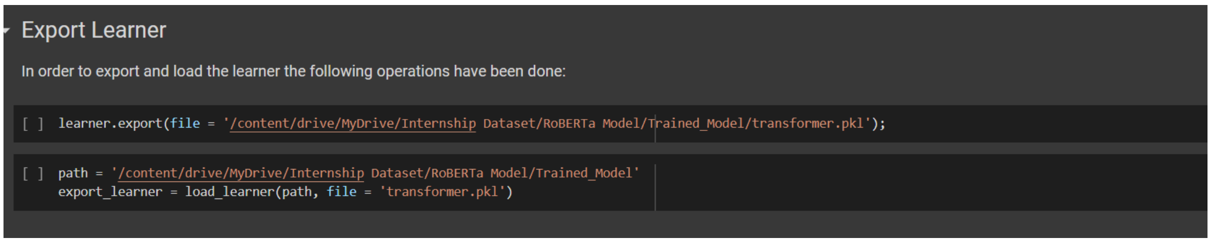
Fig- Plotting one cycle by unfreezing third group of layers



```
[ ] learner.unfreeze()

[ ] learner.fit_one_cycle(2, max_lr=slice(lr*0.95**num_groups, lr), moms=(0.8, 0.9))

    epoch  train_loss  valid_loss  accuracy  error_rate  time
        0    0.428002    0.492189  0.849233    0.150767  02:38
        1    0.254940    0.254481  0.918612    0.081388  02:28
    /usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
      return array(a, dtype, copy=False, order=order)
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de
    /usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tupl
      return array(a, dtype, copy=False, order=order)
    <string>:6: VisibleDeprecationWarning: Creating an ndarray from ragged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different lengths or shapes) is de
```

Fig- Plotting one cycle by unfreezing all the group of layers

- In order to export the learner, following operations have been performed.



Export Learner

In order to export and load the learner the following operations have been done:

```
[ ] learner.export(file = '/content/drive/MyDrive/Internship Dataset/RoBERTa Model/Trained_Model/transformer.pkl');

[ ] path = '/content/drive/MyDrive/Internship Dataset/RoBERTa Model/Trained_Model'
    export_learner = load_learner(path, file = 'transformer.pkl')
```

Fig- Export Learner

- In the above notebook we had performed Sentiment Analysis of Twitter Tweets by integrating and using both the Huggingface Transformer Library and FastAI. We had used the Slanted Triangular Learning Rates, Discriminate Learning Rate and even Gradual Unfreezing ,as a result, without even tunning the parameters, you can obtain rapidly state-of-the-art results.

Link to Notebook.

RoBERTa-CNN Sentiment Extractor: After the completion of the sentiment analysis the data was further explored for the sentiment triggers in the tweets. HuggingFace transformers don't have a TFRobertaForQuestionAnswering, for this purpose, a TFRobertaModel was created to convert trained data into arrays that the Roberta model can interpret.

Here, we tokenize the data, create question answer targets, and how to build a custom question answer head for roBERTa in TensorFlow. Note that HuggingFace transformers don't have a TFRobertaForQuestionAnswering so we must make our own from TFRobertaModel. Roberta with CNN head is used for Twitter Sentiment Extraction.

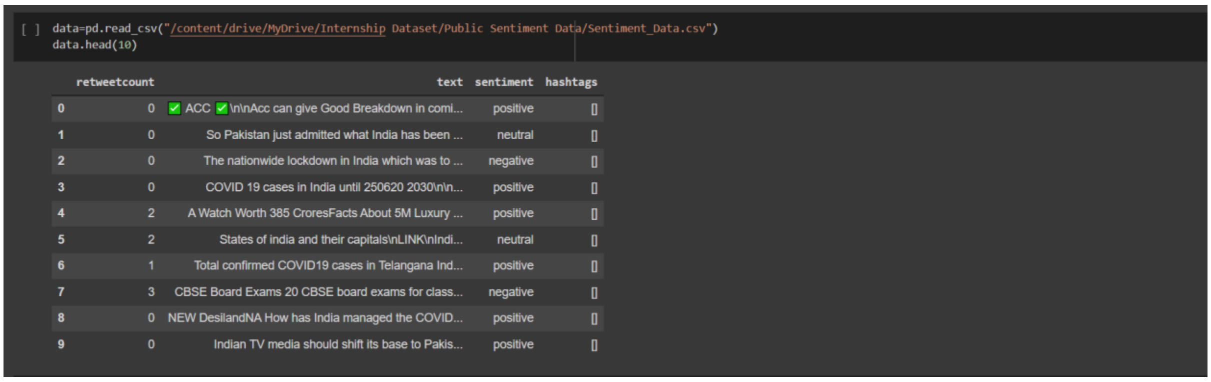
While training the Sentiment Extractor model, 5 stratified KFolds were used in such a way that, in each fold, the best model weights were saved and these weights were reloaded before carrying out testing and predictions. Roberta with CNN head was used for Twitter Sentiment Extraction.

Thus after passing the data through this model we obtained a new column of the extracted text for the sentiments which was also used to plot certain graphs.

Link to Notebook.

Public Sentimental Analysis:

- Public Sentiment Data were downloaded from Kaggle with information about hashtags.

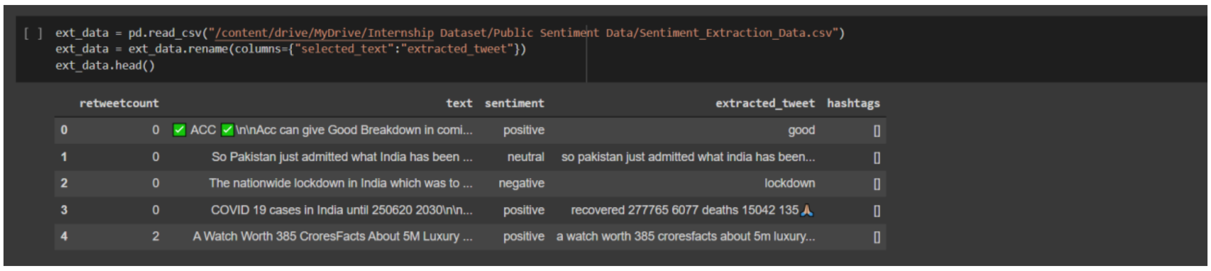


```
[ ] data=pd.read_csv("/content/drive/MyDrive/Internship Dataset/Public Sentiment Data/Sentiment_Data.csv")
    data.head(10)
```

	retweetcount	text	sentiment	hashtags
0	0	✅ ACC ✅\n\nAcc can give Good Breakdown in comi...	positive	[]
1	0	So Pakistan just admitted what India has been ...	neutral	[]
2	0	The nationwide lockdown in India which was to ...	negative	[]
3	0	COVID 19 cases in India until 250620 2030\n\n...	positive	[]
4	2	A Watch Worth 385 CroresFacts About 5M Luxury ...	positive	[]
5	2	States of india and their capitals\nLINK\nIndi...	neutral	[]
6	1	Total confirmed COVID19 cases in Telangana Ind...	positive	[]
7	3	CBSE Board Exams 20 CBSE board exams for class...	negative	[]
8	0	NEW DesilandNA How has India managed the COVID...	positive	[]
9	0	Indian TV media should shift its base to Pakis...	positive	[]

Fig- Sentiment Data

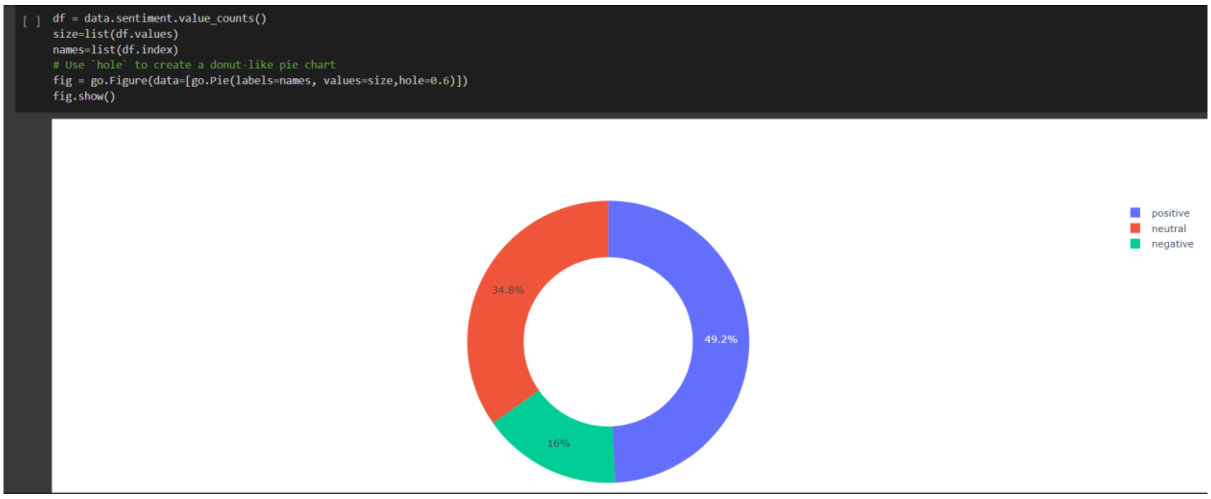
- Extraction data were also downloaded from Kaggle having information about extracted texts.



```
[ ] ext_data = pd.read_csv("/content/drive/MyDrive/Internship Dataset/Public Sentiment Data/Sentiment_Extraction_Data.csv")
    ext_data = ext_data.rename(columns={"selected_text":"extracted_tweet"})
    ext_data.head()
```

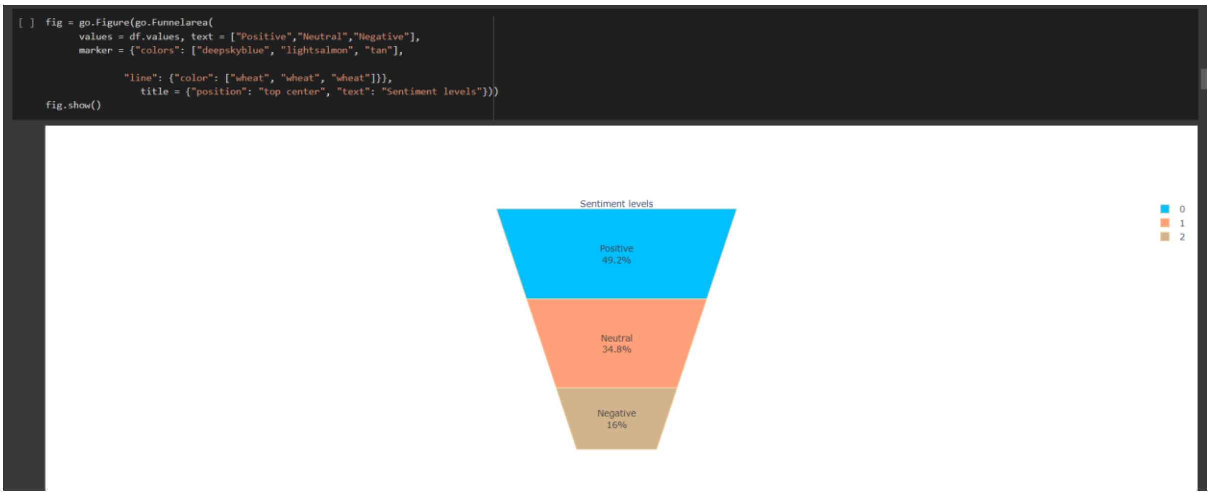
	retweetcount	text	sentiment	extracted_tweet	hashtags
0	0	✅ ACC ✅\n\nAcc can give Good Breakdown in comi...	positive	good	[]
1	0	So Pakistan just admitted what India has been ...	neutral	so pakistan just admitted what india has been...	[]
2	0	The nationwide lockdown in India which was to ...	negative	lockdown	[]
3	0	COVID 19 cases in India until 250620 2030\n\n...	positive	recovered 277765 6077 deaths 15042 135🙏	[]
4	2	A Watch Worth 385 CroresFacts About 5M Luxury ...	positive	a watch worth 385 croresfacts about 5m luxury...	[]

Fig- Sentiment Extraction Data



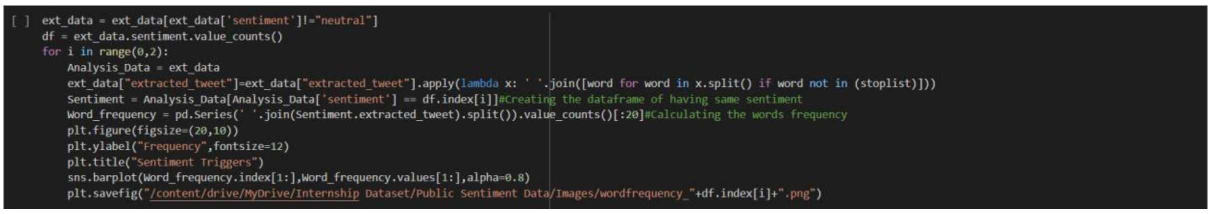
```
[ ] df = data.sentiment.value_counts()
    size=list(df.values)
    names=list(df.index)
    # Use `hole` to create a donut-like pie chart
    fig = go.Figure(data=[go.Pie(labels=names, values=size,hole=0.6)])
    fig.show()
```

Fig- Sentiment Value Counts



```
[ ] fig = go.Figure(go.Funnelarea(
          values = df.values, text = ["Positive","Neutral","Negative"],
          marker = {"colors": ["deepskyblue", "lightsalmon", "tan"],

                    "line": {"color": ["wheat", "wheat", "wheat"]}},
                       title = {"position": "top center", "text": "Sentiment levels"}))
    fig.show()
```

Fig- Funnel Figure for Sentiment Counts



```
[ ] ext_data = ext_data[ext_data['sentiment']!="neutral"]
    df = ext_data.sentiment.value_counts()
    for i in range(0,2):
        Analysis_Data = ext_data
        ext_data["extracted_tweet"]=ext_data["extracted_tweet"].apply(lambda x: ' '.join([word for word in x.split() if word not in (stoplist)]))
        Sentiment = Analysis_Data[Analysis_Data['sentiment'] == df.index[i]]#Creating the dataframe of having same sentiment
        Word_frequency = pd.Series(' '.join(Sentiment.extracted_tweet).split()).value_counts()[:20]#Calculating the words frequency
        plt.figure(figsize=(20,10))
        plt.ylabel("Frequency",fontsize=12)
        plt.title("Sentiment Triggers")
        sns.barplot(Word_frequency.index[1:],Word_frequency.values[1:],alpha=0.8)
        plt.savefig("/content/drive/MyDrive/Internship Dataset/Public Sentiment Data/Images/wordfrequency_"+df.index[i]+".png")
```

Fig- Code for Plotting Word Frequency

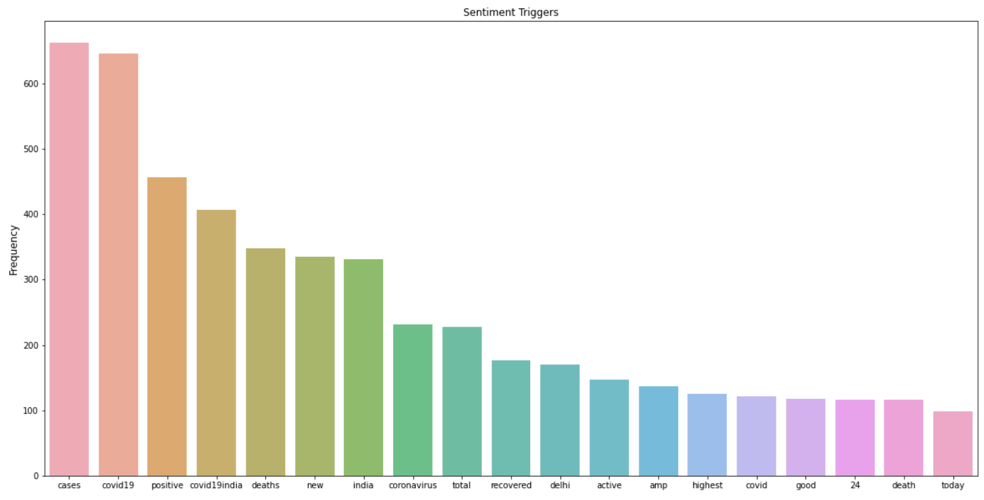


Fig- Word Frequency Positive

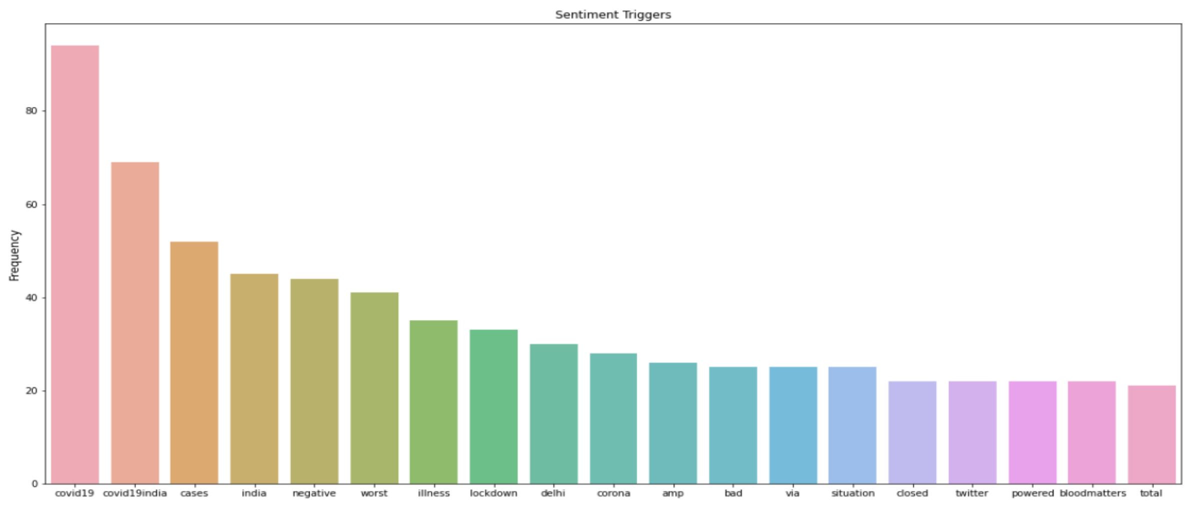


Fig- Word Frequency Negative

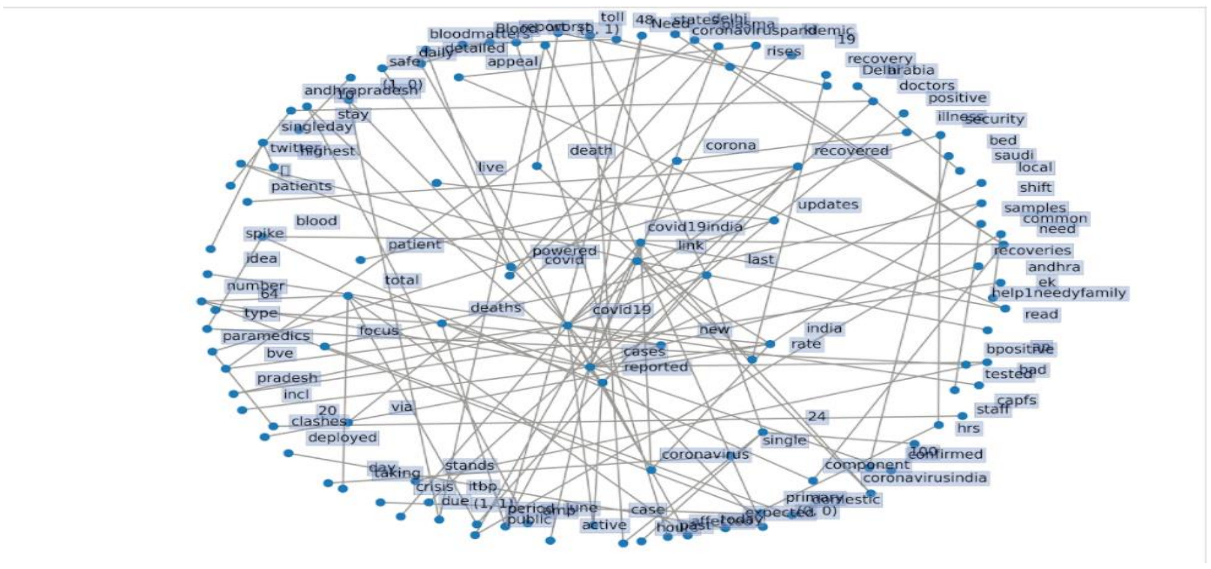


Fig- Positive Sentiment Network Plot

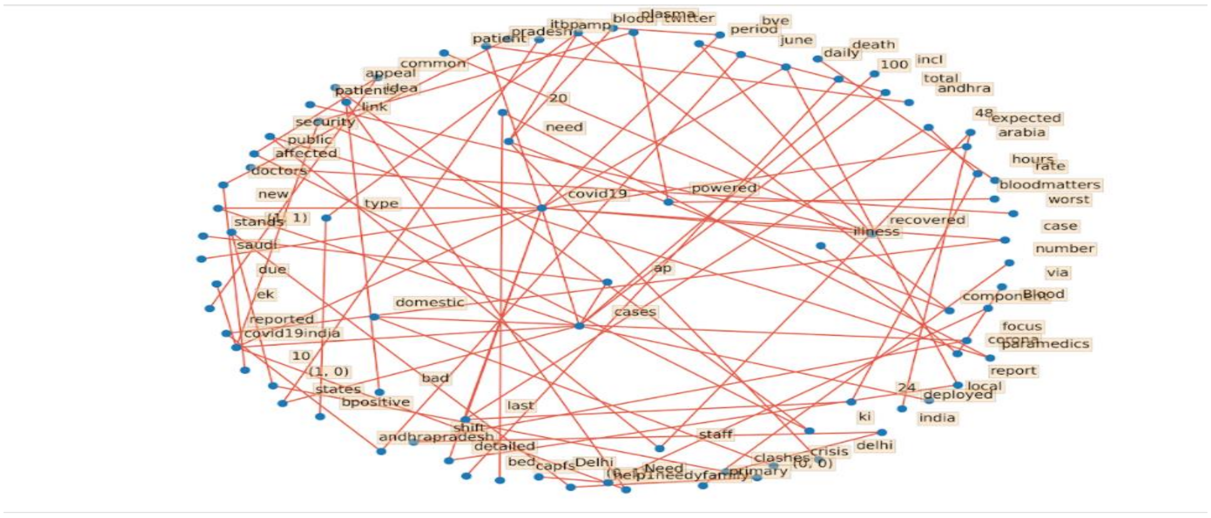
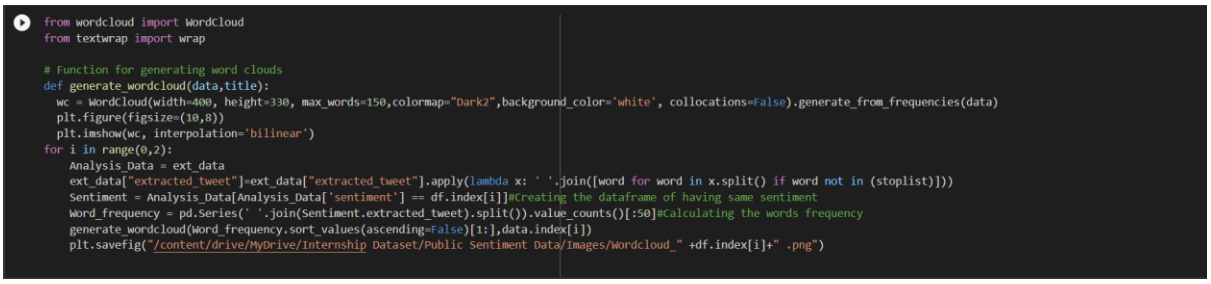


Fig- Negative Sentiment Network Plot



```python
from wordcloud import WordCloud
from textwrap import wrap

# Function for generating word clouds
def generate_wordcloud(data,title):
  wc = WordCloud(width=400, height=330, max_words=150,colormap="Dark2",background_color='white', collocations=False).generate_from_frequencies(data)
  plt.figure(figsize=(10,8))
  plt.imshow(wc, interpolation='bilinear')
for i in range(0,2):
    Analysis_Data = ext_data
    ext_data["extracted_tweet"]=ext_data["extracted_tweet"].apply(lambda x: ' '.join([word for word in x.split() if word not in (stoplist)]))
    Sentiment = Analysis_Data[Analysis_Data['sentiment'] == df.index[i]]#Creating the dataframe of having same sentiment
    Word_frequency = pd.Series(' '.join(Sentiment.extracted_tweet).split()).value_counts()[:50]#Calculating the words frequency
    generate_wordcloud(Word_frequency.sort_values(ascending=False)[1:],data.index[i])
    plt.savefig("/content/drive/MyDrive/Internship Dataset/Public Sentiment Data/Images/Wordcloud_" +df.index[i]+" .png")
```

Fig- Code for Plotting Wordcloud

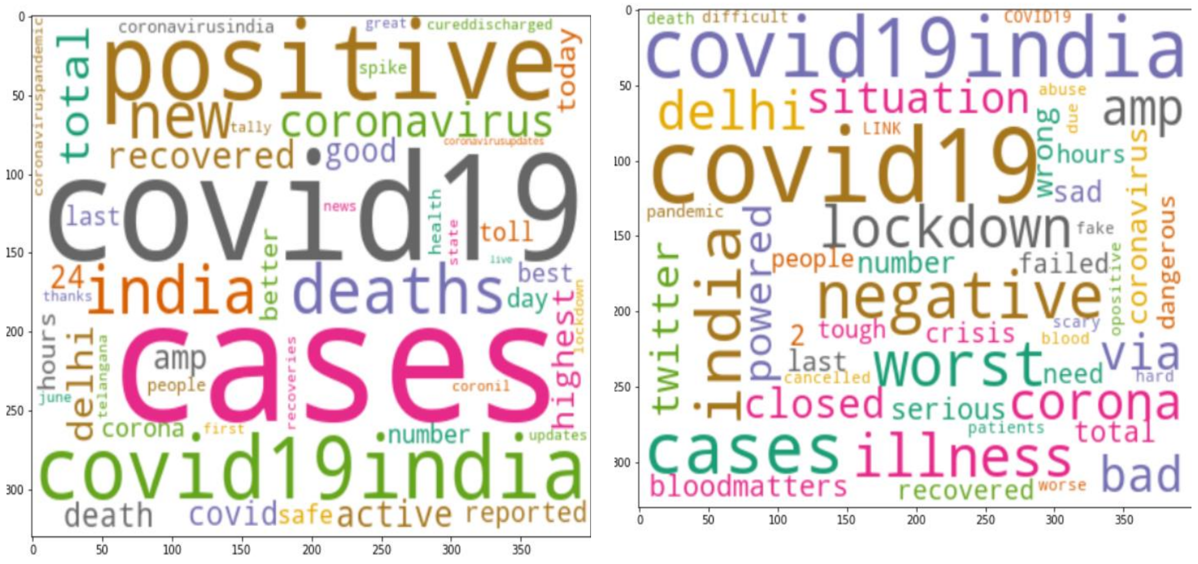
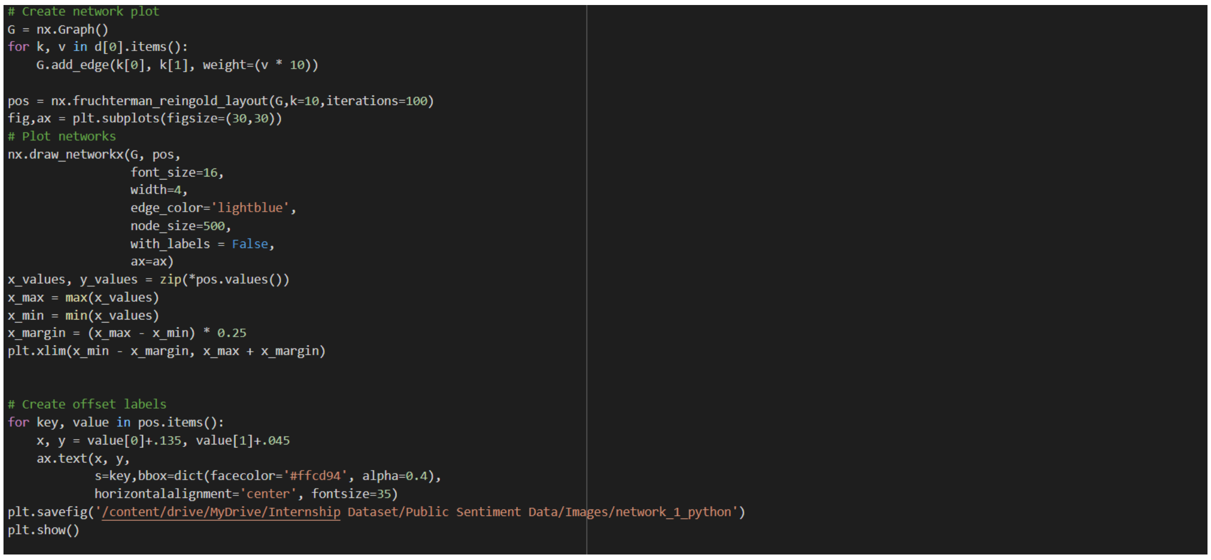


Fig- Wordclouds (Positive and Negative)



```python
# Create network plot
G = nx.Graph()
for k, v in d[0].items():
    G.add_edge(k[0], k[1], weight=(v * 10))

pos = nx.fruchterman_reingold_layout(G,k=10,iterations=100)
fig,ax = plt.subplots(figsize=(30,30))
# Plot networks
nx.draw_networkx(G, pos,
                 font_size=16,
                 width=4,
                 edge_color='lightblue',
                 node_size=500,
                 with_labels = False,
                 ax=ax)
x_values, y_values = zip(*pos.values())
x_max = max(x_values)
x_min = min(x_values)
x_margin = (x_max - x_min) * 0.25
plt.xlim(x_min - x_margin, x_max + x_margin)


# Create offset labels
for key, value in pos.items():
    x, y = value[0]+.135, value[1]+.045
    ax.text(x, y,
            s=key,bbox=dict(facecolor='#ffcd94', alpha=0.4),
            horizontalalignment='center', fontsize=35)
plt.savefig('/content/drive/MyDrive/Internship Dataset/Public Sentiment Data/Images/network_1_python')
plt.show()
```

Fig- Code for Plotting Network Plot for Extracted Tweets

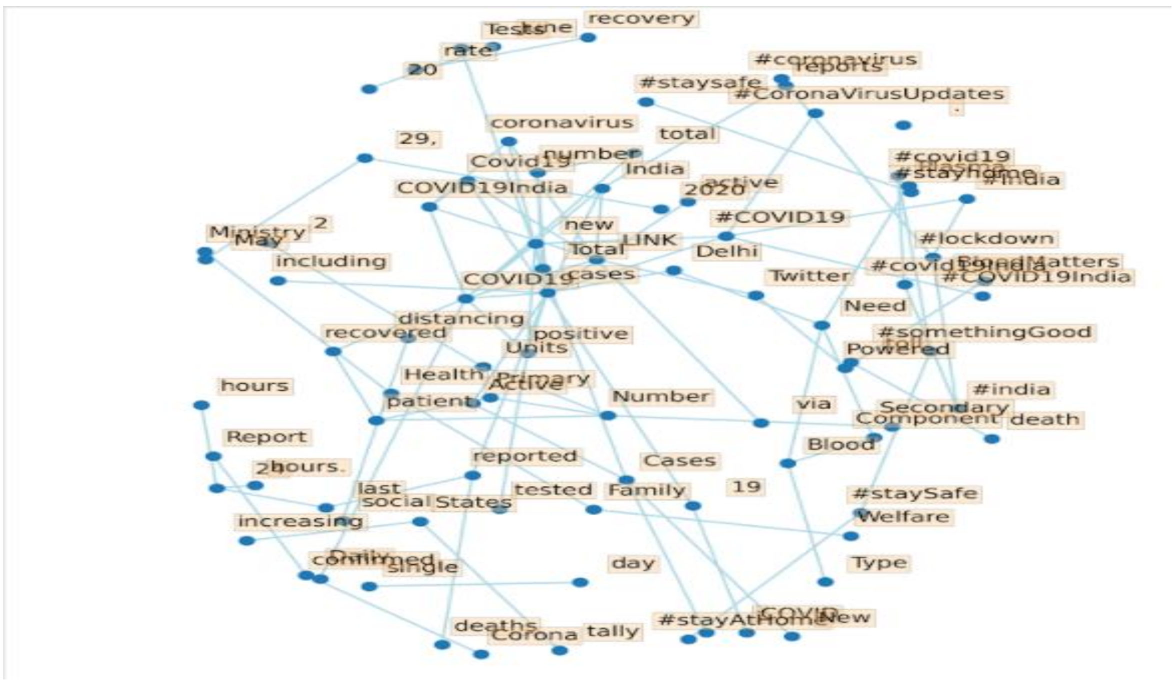
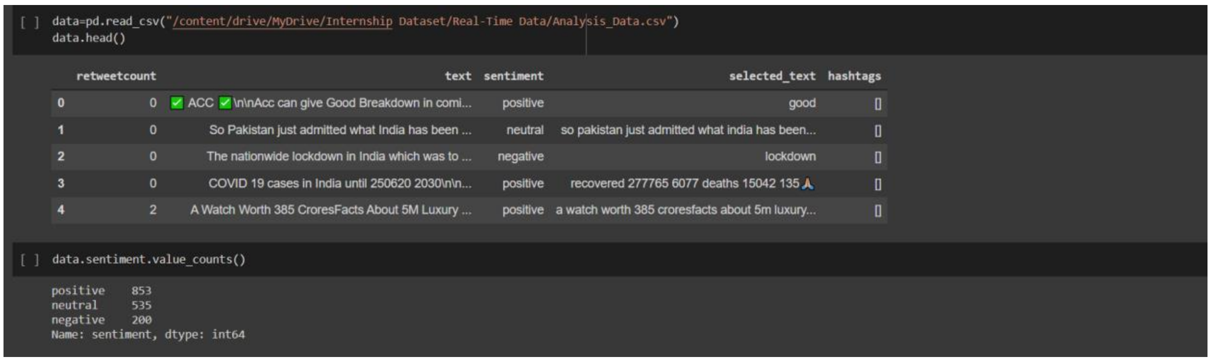


Fig- Network Plot for Extracted Tweets

Link to Notebook.

Real Time Data Analysis:

- For the Real Time Data Analysis we would be using the Analysis Data extracted from the selected texts while training our RoBERTa Model.



```
[ ] data=pd.read_csv("/content/drive/MyDrive/Internship Dataset/Real-Time Data/Analysis_Data.csv")
    data.head()
```

	retweetcount	text	sentiment	selected_text	hashtags
0	0	✅ ACC ✅ \n\nAcc can give Good Breakdown in comi...	positive	good	[]
1	0	So Pakistan just admitted what India has been ...	neutral	so pakistan just admitted what india has been...	[]
2	0	The nationwide lockdown in India which was to ...	negative	lockdown	[]
3	0	COVID 19 cases in India until 250620 2030\n\n...	positive	recovered 277765 6077 deaths 15042 135 🙏	[]
4	2	A Watch Worth 385 CroresFacts About 5M Luxury ...	positive	a watch worth 385 croresfacts about 5m luxury...	[]

```
[ ] data.sentiment.value_counts()

    positive    853
    neutral     535
    negative    200
    Name: sentiment, dtype: int64
```

Fig- Analysis Data

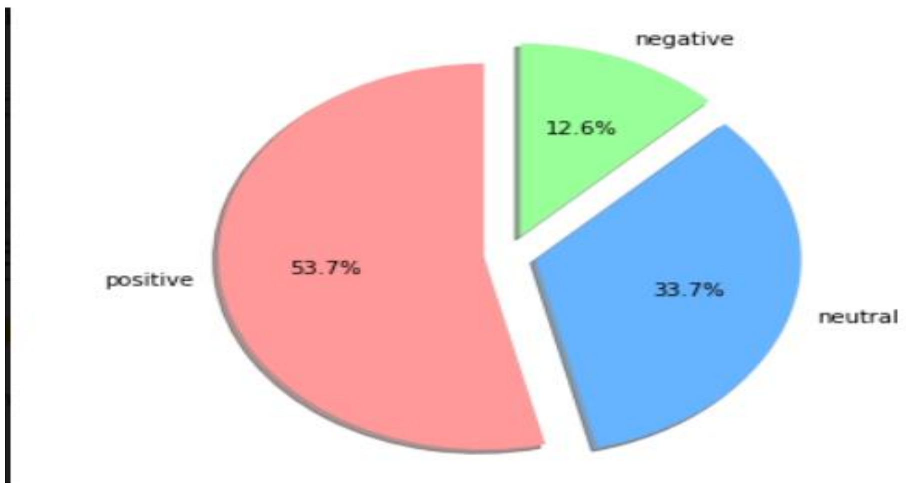


Fig- Pie Chart Depicting Sentiments

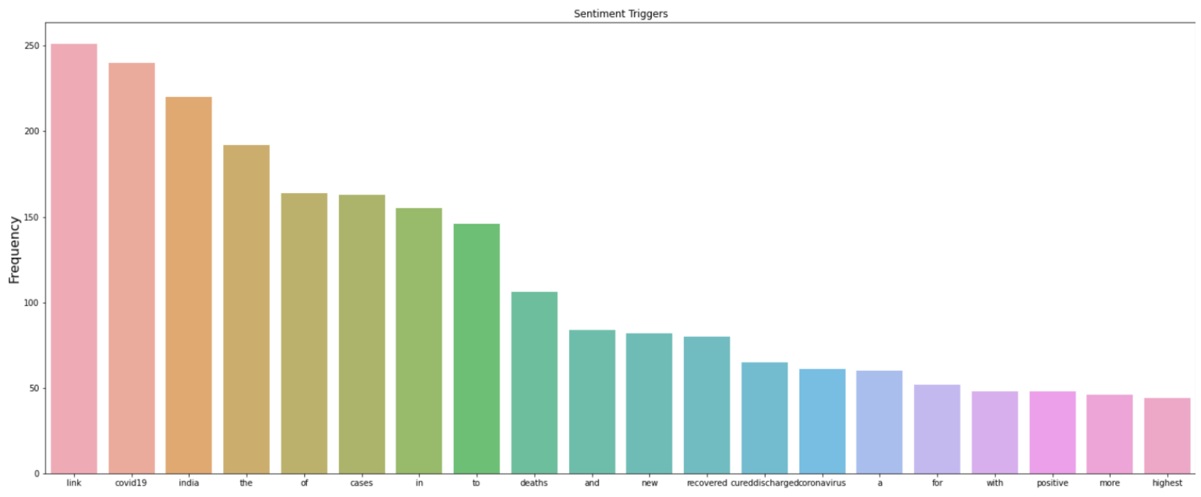


Fig- Word Frequency Positive

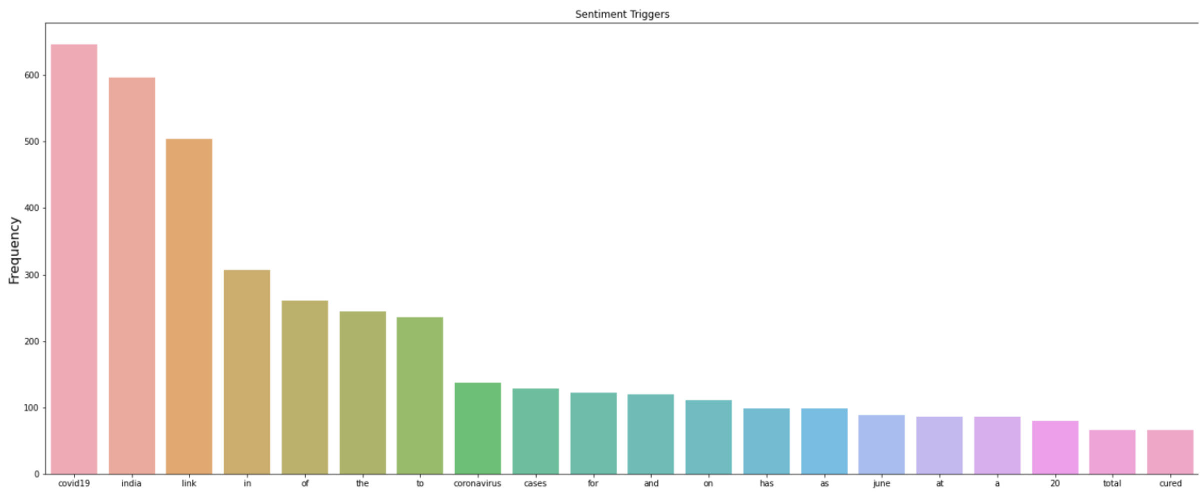


Fig- Word Frequency Neutral

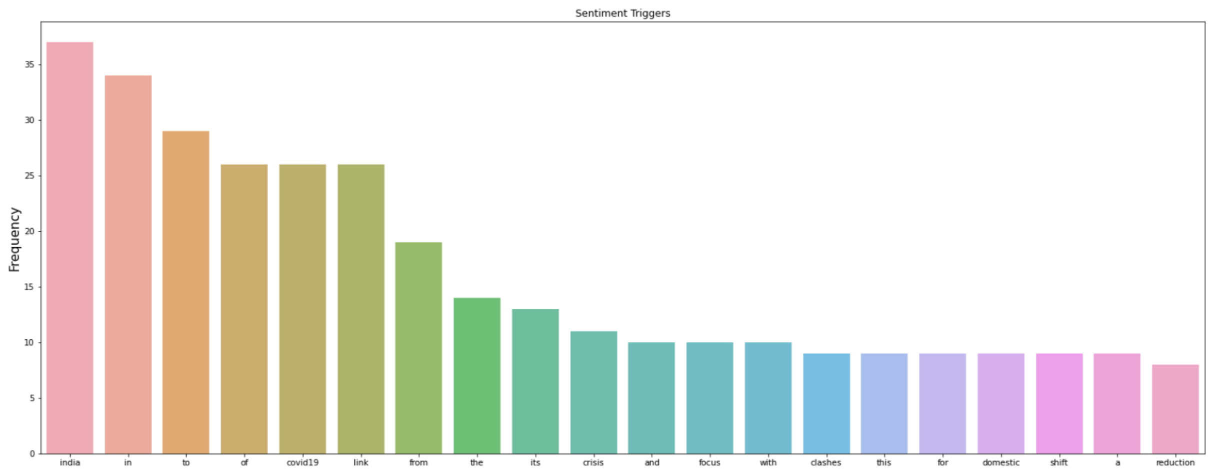


Fig- Word Frequency Negative

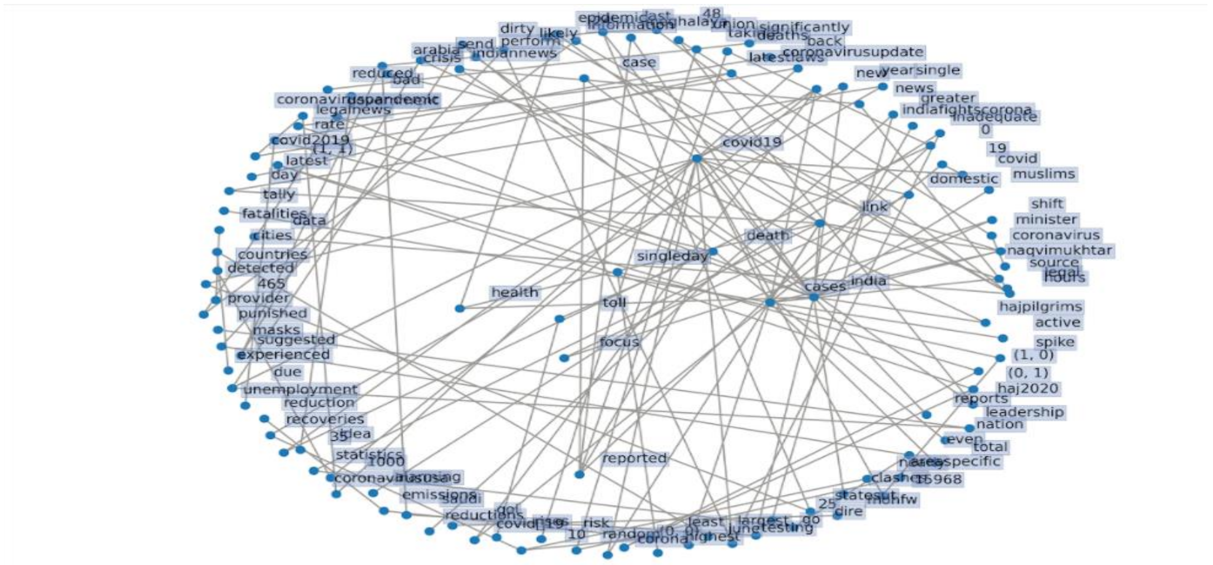


Fig- Positive Sentiment Network Plot

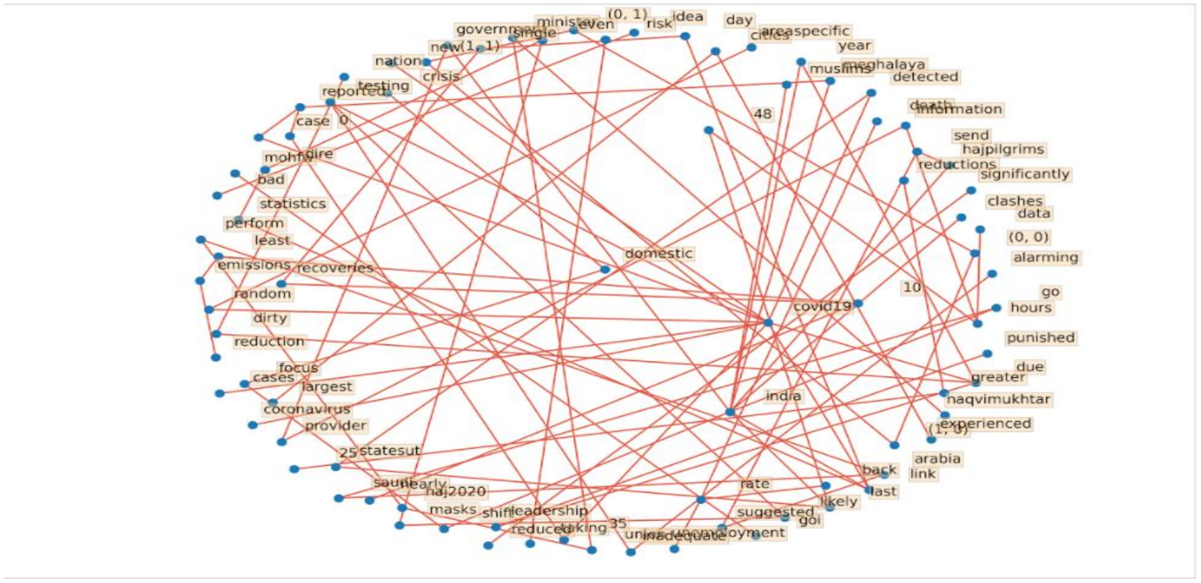


Fig- Negative Sentiment Network Plot

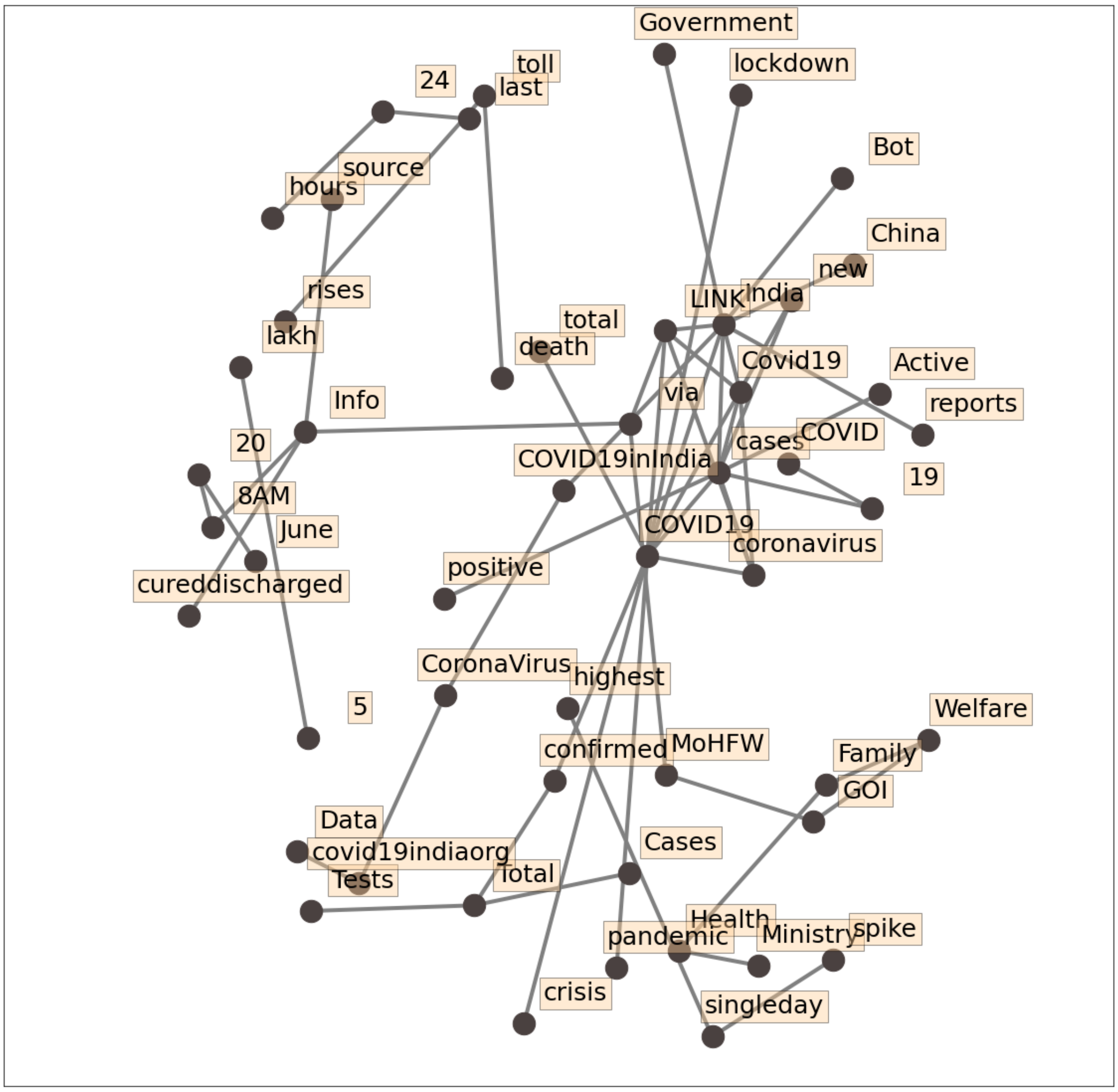
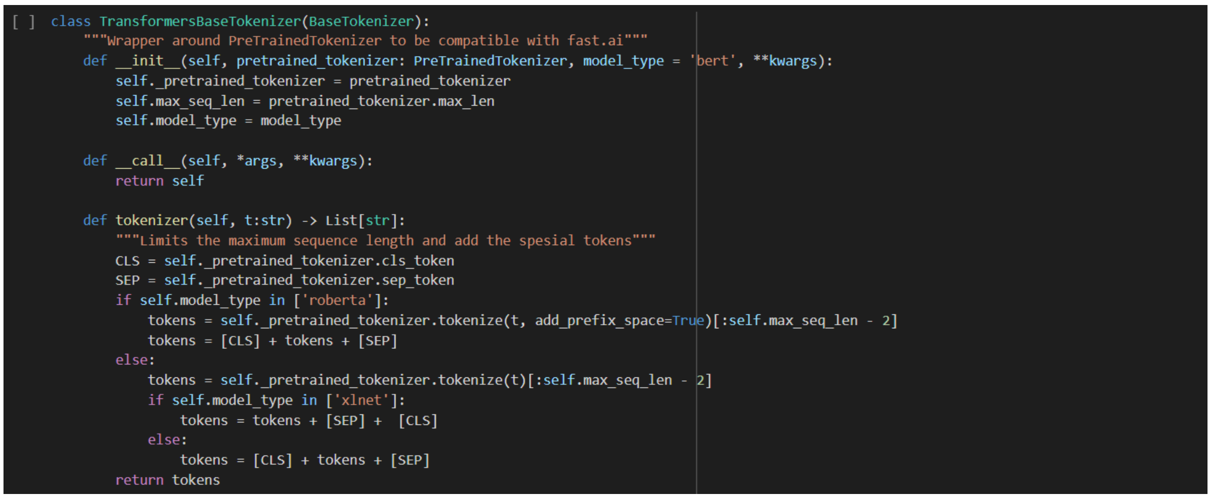


Fig- Network Plot for Selected Text

Link to Notebook.

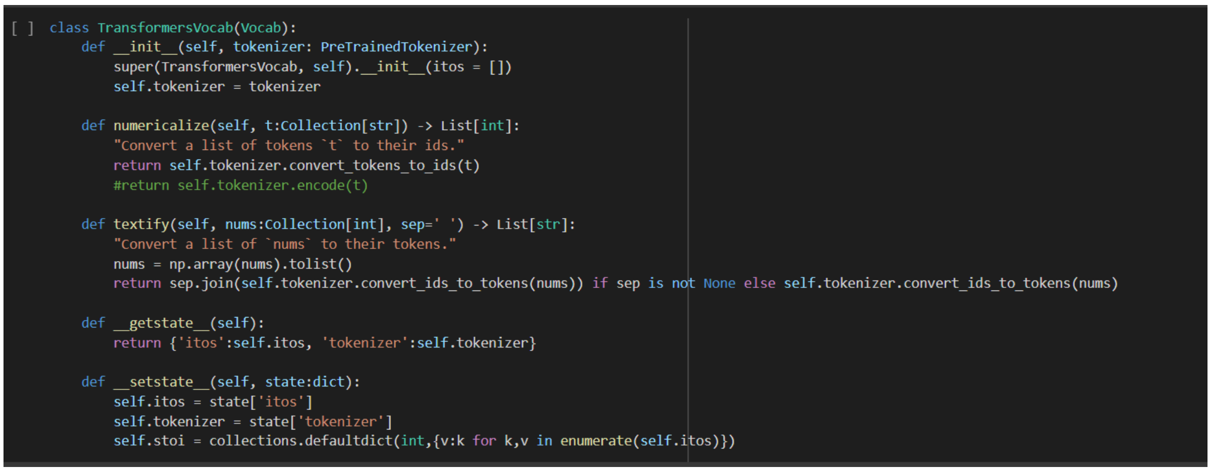
Sentiment Analysis:

- In order to perform sentiment analysis we first need to define our custom tokenizer and custom numericalizer.



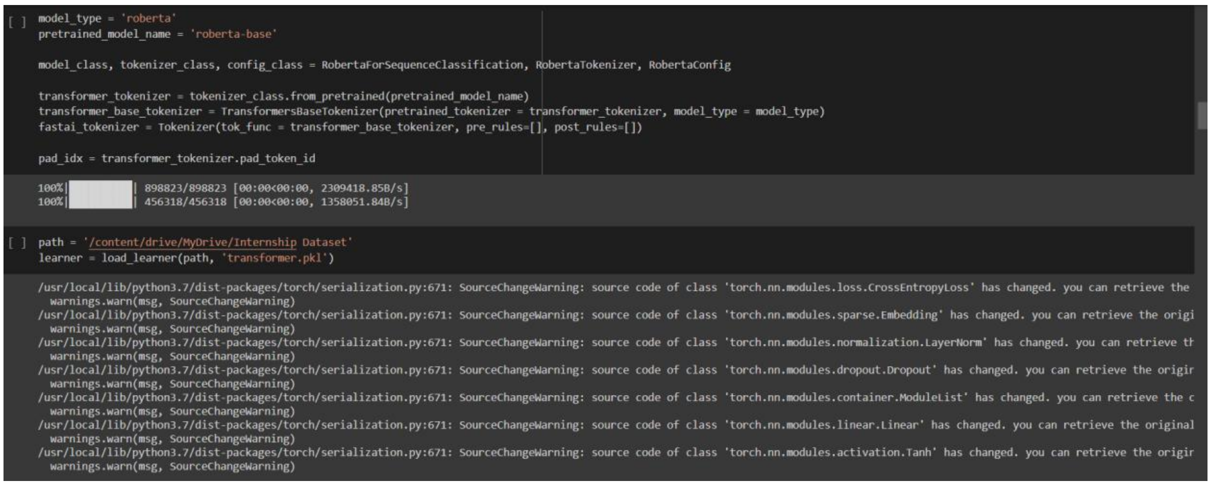
```
[ ] class TransformersBaseTokenizer(BaseTokenizer):
        """Wrapper around PreTrainedTokenizer to be compatible with fast.ai"""
        def __init__(self, pretrained_tokenizer: PreTrainedTokenizer, model_type = 'bert', **kwargs):
            self._pretrained_tokenizer = pretrained_tokenizer
            self.max_seq_len = pretrained_tokenizer.max_len
            self.model_type = model_type

        def __call__(self, *args, **kwargs):
            return self

        def tokenizer(self, t:str) -> List[str]:
            """Limits the maximum sequence length and add the spesial tokens"""
            CLS = self._pretrained_tokenizer.cls_token
            SEP = self._pretrained_tokenizer.sep_token
            if self.model_type in ['roberta']:
                tokens = self._pretrained_tokenizer.tokenize(t, add_prefix_space=True)[:self.max_seq_len - 2]
                tokens = [CLS] + tokens + [SEP]
            else:
                tokens = self._pretrained_tokenizer.tokenize(t)[:self.max_seq_len - 2]
                if self.model_type in ['xlnet']:
                    tokens = tokens + [SEP] +  [CLS]
                else:
                    tokens = [CLS] + tokens + [SEP]
            return tokens
```

Fig- Custom Tokenizer



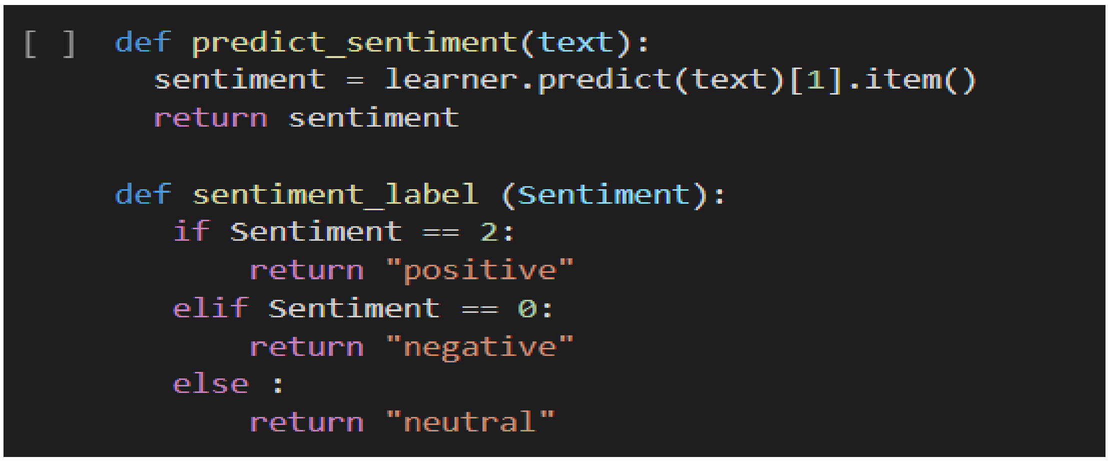
```
[ ] class TransformersVocab(Vocab):
        def __init__(self, tokenizer: PreTrainedTokenizer):
            super(TransformersVocab, self).__init__(itos = [])
            self.tokenizer = tokenizer

        def numericalize(self, t:Collection[str]) -> List[int]:
            "Convert a list of tokens `t` to their ids."
            return self.tokenizer.convert_tokens_to_ids(t)
            #return self.tokenizer.encode(t)

        def textify(self, nums:Collection[int], sep=' ') -> List[str]:
            "Convert a list of `nums` to their tokens."
            nums = np.array(nums).tolist()
            return sep.join(self.tokenizer.convert_ids_to_tokens(nums)) if sep is not None else self.tokenizer.convert_ids_to_tokens(nums)

        def __getstate__(self):
            return {'itos':self.itos, 'tokenizer':self.tokenizer}

        def __setstate__(self, state:dict):
            self.itos = state['itos']
            self.tokenizer = state['tokenizer']
            self.stoi = collections.defaultdict(int,{v:k for k,v in enumerate(self.itos)})
```

Fig- Custom Numericalizer



```
[ ] model_type = 'roberta'
    pretrained_model_name = 'roberta-base'

    model_class, tokenizer_class, config_class = RobertaForSequenceClassification, RobertaTokenizer, RobertaConfig

    transformer_tokenizer = tokenizer_class.from_pretrained(pretrained_model_name)
    transformer_base_tokenizer = TransformersBaseTokenizer(pretrained_tokenizer = transformer_tokenizer, model_type = model_type)
    fastai_tokenizer = Tokenizer(tok_func = transformer_base_tokenizer, pre_rules=[], post_rules=[])

    pad_idx = transformer_tokenizer.pad_token_id

    100%|██████████| 898823/898823 [00:00<00:00, 2309418.85B/s]
    100%|██████████| 456318/456318 [00:00<00:00, 1358051.84B/s]

[ ] path = '/content/drive/MyDrive/Internship Dataset'
    learner = load_learner(path, 'transformer.pkl')

    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.loss.CrossEntropyLoss' has changed. you can retrieve the
      warnings.warn(msg, SourceChangeWarning)
    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.sparse.Embedding' has changed. you can retrieve the origi
      warnings.warn(msg, SourceChangeWarning)
    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.normalization.LayerNorm' has changed. you can retrieve th
      warnings.warn(msg, SourceChangeWarning)
    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.dropout.Dropout' has changed. you can retrieve the origir
      warnings.warn(msg, SourceChangeWarning)
    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.container.ModuleList' has changed. you can retrieve the c
      warnings.warn(msg, SourceChangeWarning)
    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.linear.Linear' has changed. you can retrieve the original
      warnings.warn(msg, SourceChangeWarning)
    /usr/local/lib/python3.7/dist-packages/torch/serialization.py:671: SourceChangeWarning: source code of class 'torch.nn.modules.activation.Tanh' has changed. you can retrieve the origir
      warnings.warn(msg, SourceChangeWarning)
```

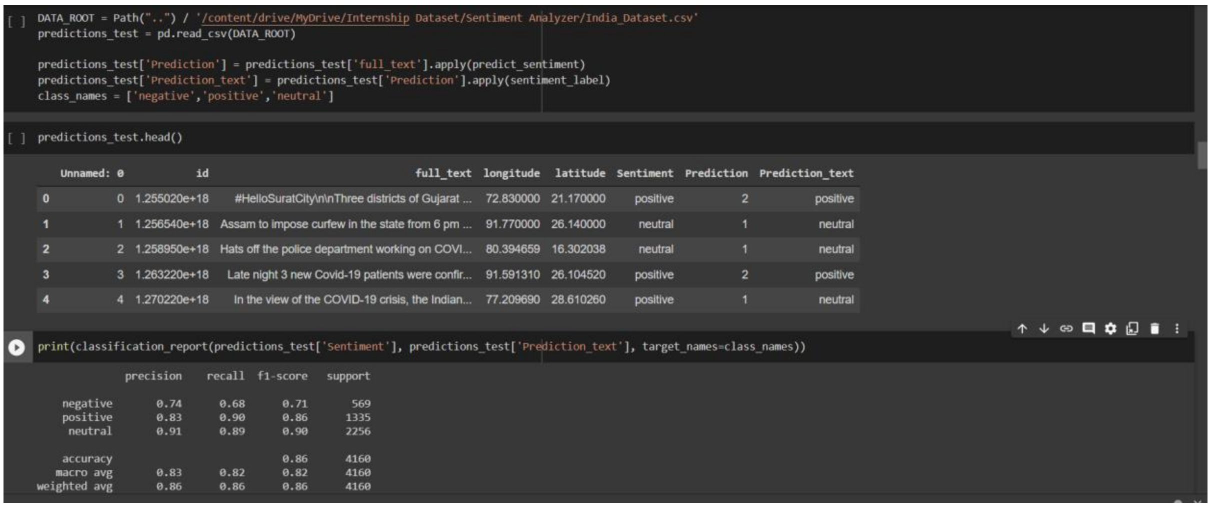
Fig- Declaring Path for our Trained Model



```
[ ]  def predict_sentiment(text):
       sentiment = learner.predict(text)[1].item()
       return sentiment

     def sentiment_label (Sentiment):
        if Sentiment == 2:
            return "positive"
        elif Sentiment == 0:
            return "negative"
        else :
            return "neutral"
```

Fig- Defining Prediction Function

- We will call this function in our India_Dataset.csv file to predict the sentiments and further calculating the accuracy.



```
[ ] DATA_ROOT = Path("..") / '/content/drive/MyDrive/Internship Dataset/Sentiment Analyzer/India_Dataset.csv'
    predictions_test = pd.read_csv(DATA_ROOT)

    predictions_test['Prediction'] = predictions_test['full_text'].apply(predict_sentiment)
    predictions_test['Prediction_text'] = predictions_test['Prediction'].apply(sentiment_label)
    class_names = ['negative','positive','neutral']
```

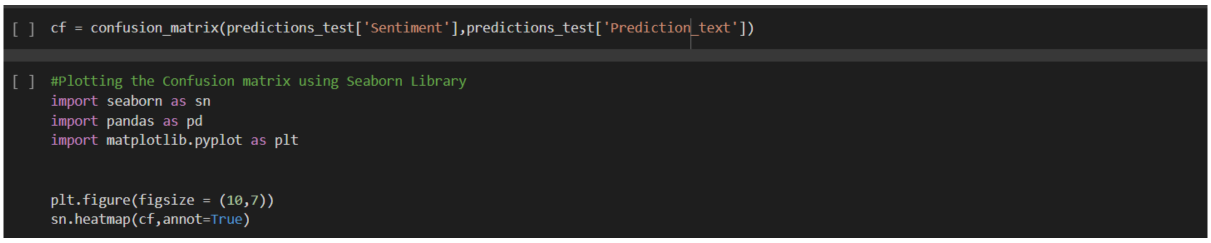
```
[ ] predictions_test.head()
```

	Unnamed: 0	id	full_text	longitude	latitude	Sentiment	Prediction	Prediction_text
0	0	1.255020e+18	#HelloSuratCity\n\nThree districts of Gujarat ...	72.830000	21.170000	positive	2	positive
1	1	1.256540e+18	Assam to impose curfew in the state from 6 pm ...	91.770000	26.140000	neutral	1	neutral
2	2	1.258950e+18	Hats off the police department working on COVI...	80.394659	16.302038	neutral	1	neutral
3	3	1.263220e+18	Late night 3 new Covid-19 patients were confir...	91.591310	26.104520	positive	2	positive
4	4	1.270220e+18	In the view of the COVID-19 crisis, the Indian...	77.209690	28.610260	positive	1	neutral

```
print(classification_report(predictions_test['Sentiment'], predictions_test['Prediction_text'], target_names=class_names))
```

```
              precision    recall  f1-score   support

    negative       0.74      0.68      0.71       569
    positive       0.83      0.90      0.86      1335
     neutral       0.91      0.89      0.90      2256

    accuracy                           0.86      4160
   macro avg       0.83      0.82      0.82      4160
weighted avg       0.86      0.86      0.86      4160
```

Fig- Calculating Predictions and Classification Report



```
[ ] cf = confusion_matrix(predictions_test['Sentiment'],predictions_test['Prediction_text'])
```

```
[ ] #Plotting the Confusion matrix using Seaborn Library
    import seaborn as sn
    import pandas as pd
    import matplotlib.pyplot as plt


    plt.figure(figsize = (10,7))
    sn.heatmap(cf,annot=True)
```

Fig- Code for plotting Confusion Matric

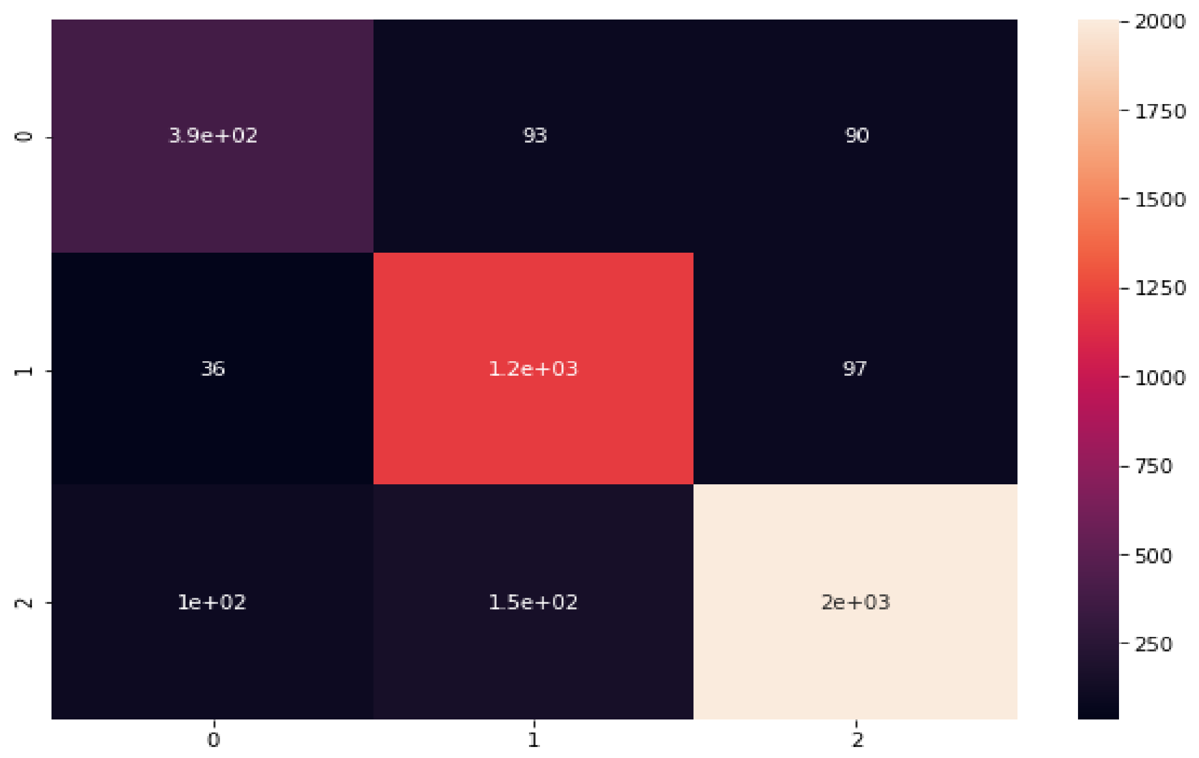
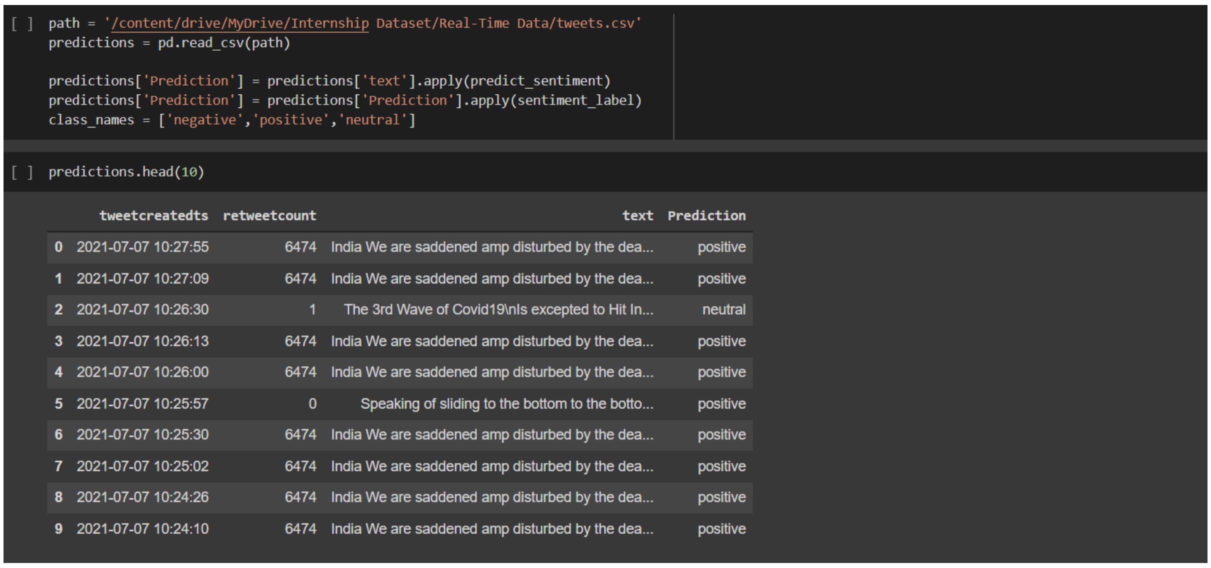


Fig- Confusion Matrix

- After sentiment analysis being completed on Indian Dataset, real time data was scrapped using tweepy and same model was used for prediction.



```
path = '/content/drive/MyDrive/Internship Dataset/Real-Time Data/tweets.csv'
predictions = pd.read_csv(path)

predictions['Prediction'] = predictions['text'].apply(predict_sentiment)
predictions['Prediction'] = predictions['Prediction'].apply(sentiment_label)
class_names = ['negative','positive','neutral']
```

```
predictions.head(10)
```

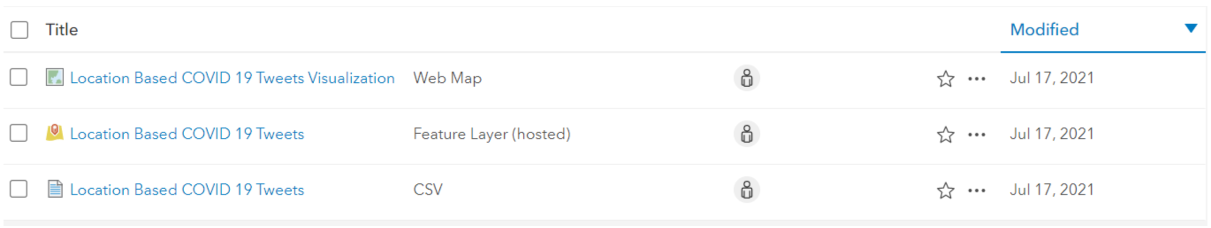
	tweetcreatedts	retweetcount	text	Prediction
0	2021-07-07 10:27:55	6474	India We are saddened amp disturbed by the dea...	positive
1	2021-07-07 10:27:09	6474	India We are saddened amp disturbed by the dea...	positive
2	2021-07-07 10:26:30	1	The 3rd Wave of Covid19\nIs excepted to Hit In...	neutral
3	2021-07-07 10:26:13	6474	India We are saddened amp disturbed by the dea...	positive
4	2021-07-07 10:26:00	6474	India We are saddened amp disturbed by the dea...	positive
5	2021-07-07 10:25:57	0	Speaking of sliding to the bottom to the botto...	positive
6	2021-07-07 10:25:30	6474	India We are saddened amp disturbed by the dea...	positive
7	2021-07-07 10:25:02	6474	India We are saddened amp disturbed by the dea...	positive
8	2021-07-07 10:24:26	6474	India We are saddened amp disturbed by the dea...	positive
9	2021-07-07 10:24:10	6474	India We are saddened amp disturbed by the dea...	positive

Fig- Sentiment Analysis on Real Time Data

Link to Notebook.

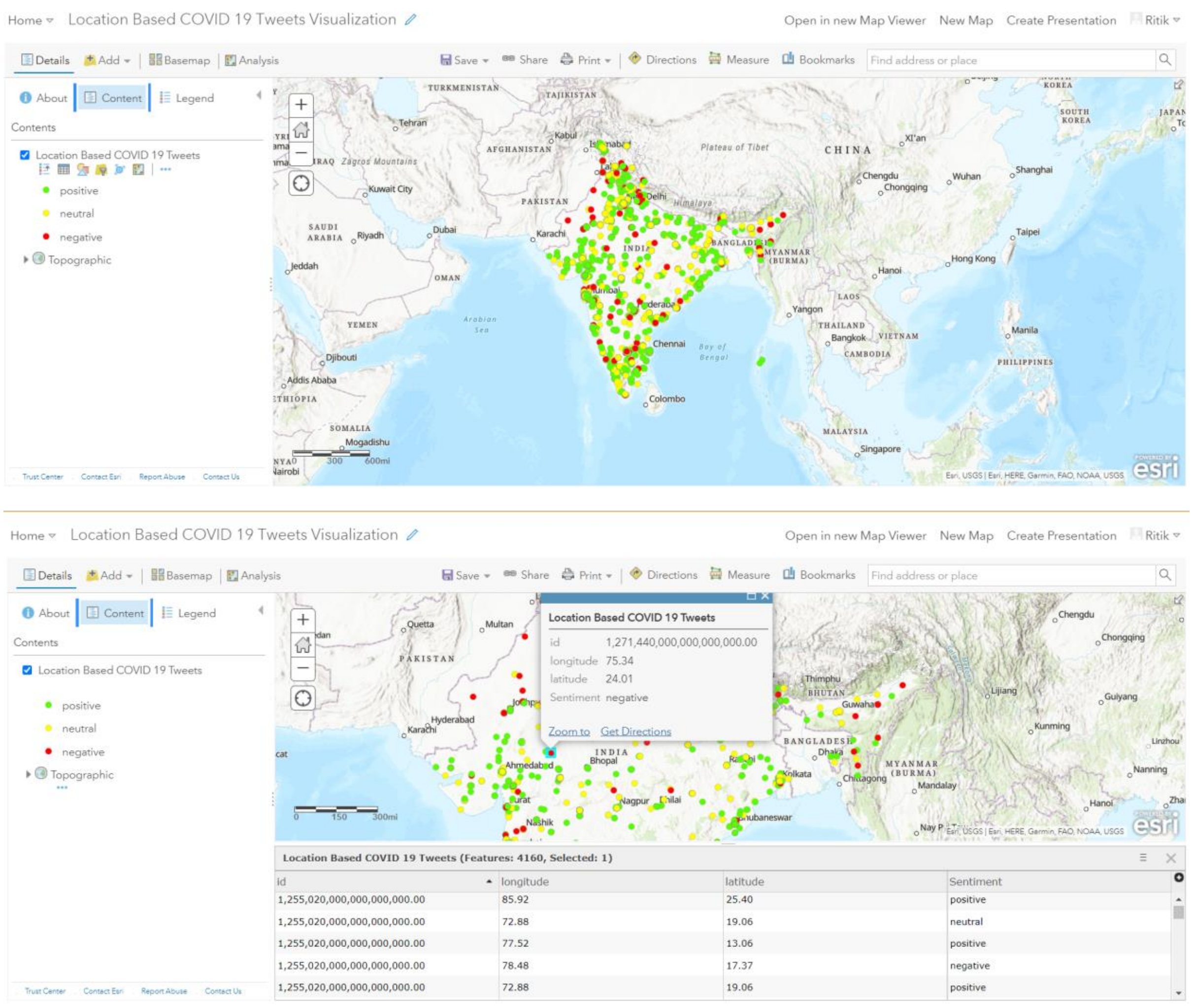
ArcGIS Online Visualization

After carrying out successful Sentimental Analysis on Indian Dataset, the CSV file was hosted as a feature layer on ArcGIS Online in order to visualize the tweets with the location as their coordinates information is already extracted while hydrating the tweets.



Title			Modified
Location Based COVID 19 Tweets Visualization	Web Map		Jul 17, 2021
Location Based COVID 19 Tweets	Feature Layer (hosted)		Jul 17, 2021
Location Based COVID 19 Tweets	CSV		Jul 17, 2021

Fig- Hosted Layers on ArcGIS Online



Location Based COVID 19 Tweets

id	1,271,440,000,000,000,000.00
longitude	75.34
latitude	24.01
Sentiment	negative

Zoom to Get Directions

Location Based COVID 19 Tweets (Features: 4160, Selected: 1)

id	longitude	latitude	Sentiment
1,255,020,000,000,000,000.00	85.92	25.40	positive
1,255,020,000,000,000,000.00	72.88	19.06	neutral
1,255,020,000,000,000,000.00	77.52	13.06	positive
1,255,020,000,000,000,000.00	78.48	17.37	negative
1,255,020,000,000,000,000.00	72.88	19.06	positive

Fig- Visualization of Location Based COVID 19 Tweets

Advantages and Disadvantages

Advantages:

The effect of COVID-19 pandemic is visible all over the world. National healthcare systems are facing the contagion with incredible strength, but concern regarding psychosocial and economic effects is critically growing. In a fast-moving crisis, as information swarms in from every direction, citizens look to their governments for information, guidance, and leadership. Sentimental Analysis is only the option in this current situation to understand the psychological condition/mental condition of the public. By Sentimental Analysis, the public opinion on COVID-19, regime policies, and actions can be understood. After Analysis, amendments can be made to the decisions taken by the regime policies, and the public can be fortified in such a way so as to enhance the sentiment towards a positive outlook. Not only this but also sentiment analysis will help NGOs and various organizations to come forward to help the people. Businesses can adapt their products and services to match the requirements of the people based on the real-time trending mood of the public, which will not only help businesses to grow but will also help the public meet their need of the hour. Also, this will enable the government to make business and people-friendly rules and laws to help in the betterment of the economy and the market in these untested times.

Disadvantages:

Natural Language Processing models, in general, face a problem in recognizing human aspects of a language like irony, sarcasm, negotiations, exaggerations, and jokes - the sorts of things humans wouldn’t face many problems in understanding. Machines sometimes fail in recognizing these aspects, which leads to skewed and incorrect results.

Applications

The application of Sentiment Analysis lies in taking the sentiments of the people into consideration and creating appropriate business strategies and government policies so as to meet their needs. For example, many food delivery companies like Swiggy and online retail stores like Amazon are now opting for “Contactless Delivery” as a preventive step towards controlling the spread of the virus. This step taken by such companies would not only increase their profits but also would provide the essential services to the people while taking their sentiments into consideration in this pandemic. Not only this but also based on the sentiments analyzed form the tweets, rehabilitation or positive suggestions can be made to users who have consistently expressed negative sentiments. This way, public health can also be monitored using sentiment analysis.

Conclusion

Sentiment analysis or opinion mining is a hot topic in deep learning. There is still a long way to go before sentiments can be accurately detected from texts, because of the complexity involved in the English language, and even more when other languages like Hindi are considered. Though the Roberta model developed as a part of this project has predicted and classified the sentiments of the test data set into positive, negative and neutral categories with an accuracy of 91%, by making necessary modifications and additions to the model, sentiment analysis can be done with greater accuracy by taking the language complexities into consideration.

Future Scope

Sentiment Analysis still has many aspects that can still be worked upon. This project can be further enhanced by training the model about the human aspects of a language and making it more accurate in cases where sarcasm, irony, and other aspects are used. Taking the actions of all the ministries into consideration while gauging the sentiments of the public can also make the analysis more detailed and sector-specific, which would help in the analysis of the area of development required in those sectors. The classifier can be further improved by trying to extract more features from the tweets, trying different kinds of features, tuning the Hyperparameters, and also by making it work on various Indian languages.

References

1. https://ieee-dataport.org/open-access/coronavirus-covid-19-tweets-dataset
2. https://www.kaggle.com/ragnisah/text-data-cleaning-tweets-analysis
3. https://cognitiveclass.ai/courses/data-visualization-with-python
4. https://github.com/DocNow/hydrator/blob/master/README.md
5. https://dphi.tech/data-science-bootcamp-day-15-exploratory-data-analysis/
6. https://github.com/huggingface/transformers
7. https://arxiv.org/abs/1801.06146
8. https://towardsdatascience.com/fastai-with-transformers-bert-roberta-xlnet-xlm-distilbert-4f41ee18ecb2
9. https://towardsdatascience.com/fasttext-sentiment-analysis-for-tweets-a-straightfor%20ward-guide-9a8c070449a2
10. https://link.medium.com/ZZvca1Mf27