SENTIMENTAL ANALYSIS OF COVID 19 TWEETS

**INDEX**

**1. INTRODUCTION**

**1.1 Overview**

**1.2 Purpose**

**2. LITERATURE SURVEY**

**2.1 Existing problem**

**2.2 Proposed solution**

**3. THEORITICAL ANALYSIS**

**3.1 Block diagram**

**3.2 Hardware / Software designing**

**4. EXPERIMENTAL INVESTIGATIONS**

**5. FLOWCHART**

**6. RESULT**

**7. ADVANTAGES & DISADVANTAGES**

**8. APPLICATIONS**

**9. CONCLUSION**

**10. FUTURE SCOPE**

**11. BIBILOGRAPHY**

**APPENDIX**

**A. Source code**

1. INTRODUCTION

1.1 Overview

The Web is a great place where everything can get connected and will be able to access the things. With the invention of cloud the Web Architecture has reached greater extent.

In this project, the topic we have taken is Sentimental analysis of Covid 19 Tweets, using Google Colab, Anvil App and IBM Watson Studio.

This, project was approached using CNN concept in machine learning and using Python as programming language the project was made. The training and testing data sets was imported and the model was trained and the trained model was saved and called next time when ever needed.

Then, using Anvil App a server connection was made between the Notebook and User Interface. Thus, the whole connection was made and a working project was completed.

1.2 Purpose

The main purpose of this project is to understand the feelings of people based on the current Covid situation, extensions of lock down and other stuff related to feelings. This Project can take a tweet or statement as the input and process it and will return the sentiment accordingly.

It also displays the status of live tweets after processing it and shows as a graphic part, the user can see it on how the value changes accordingly with time. Finally there is an option to view the live tweets from across the globe that are related to keywords of Covid 19.

2. LITERATURE SURVEY

2.1 Existing Problem

There is always a problem with government decisions regarding any thing, and this pandemic has made it to reach a peak. People feeling towards a decision is getting difficult to analyze and predict it. There is no application that can tell how people feel how they react for a particular decision.

There is no software to predict the feelings based on tweets taking directly from the website and show the feelings percentage in the form of UI. There are limited places that can take tweets directly from twitter and show the tweets.

2.2 Proposed solution

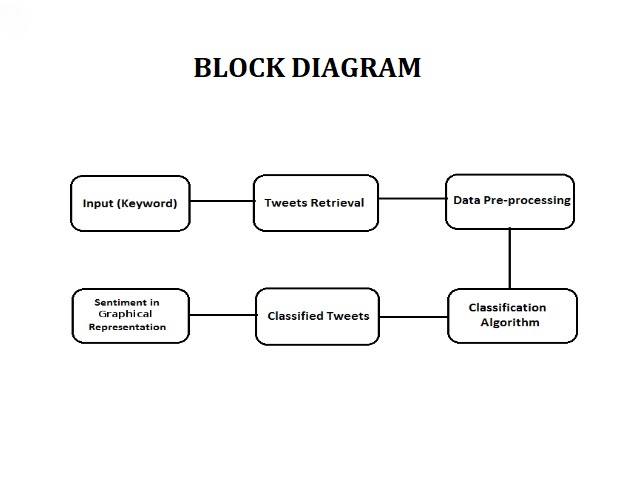
Using Machine Learning and the concept of CNN a model was created that can take a statement and process it accordingly, and distribute the sentiment of the feeling accordingly as Positive, Negative and Neutral.

Now to show the feelings of people across the globe, live tweets are taken and processed and the sentiments of positive, negative and neutral are kept and are processed into a bar graph form. The graphical part is displayed in the UI for the user to see and every time the button is pressed it refreshes and takes the current tweets that have come in the past time and changes the graph as per it.

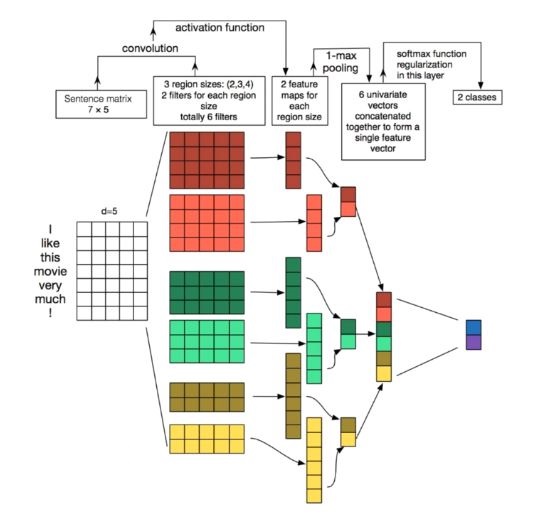
Finally a part to show the live tweets is also kept where a maximum of 300 Tweets can be displayed and the respective output will be displayed on the screen which the user can see or even if needed can copy paste it in the above text box and see the sentiment for himself.

3. THEORITICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware / Software designing

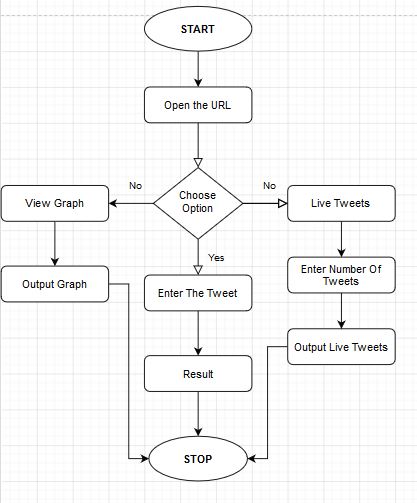


4. EXPERIMENTAL INVESTIGATIONS

While doing the following project, we came to learn about many things, and came across many investigation processes and concepts. In this Sentimental Analyzer we learnt the following things:

* IBM Cloud
* Node- Red
* Watson Studio
* Google Colab
* Anvil APP
* Slack Bot
* ZOHO Writer
* IBM Cloud Services
* Machine Learning
* Internet Of Things

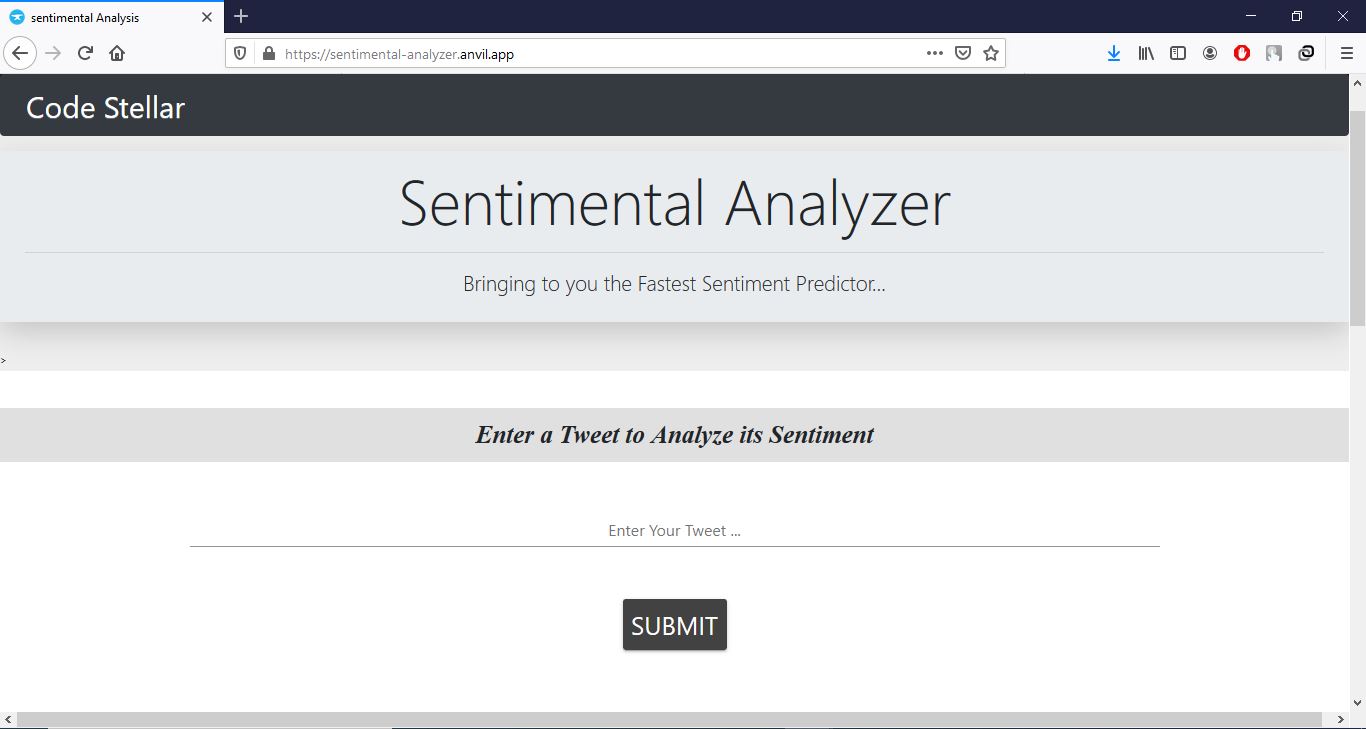
5. FLOWCHART



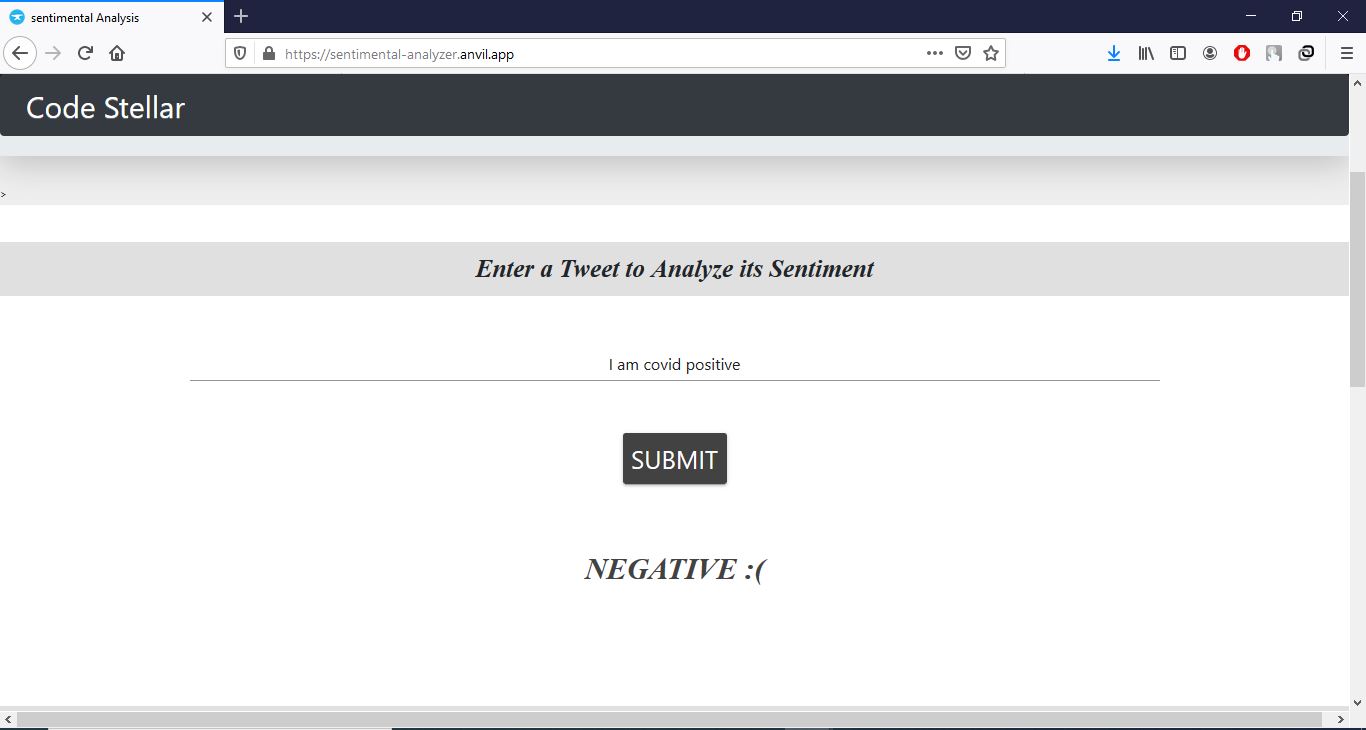
6. RESULT

USER INTERFACE

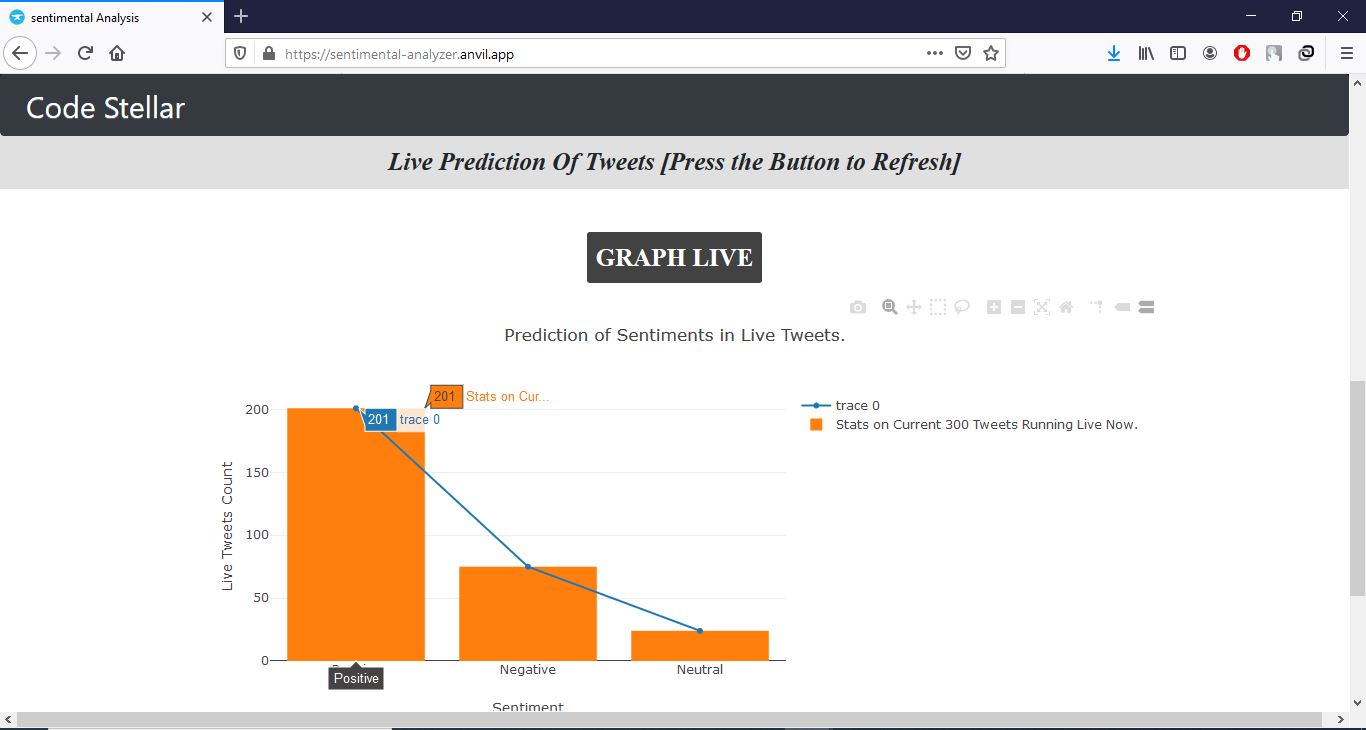
1. URL:**- https://sentimental-analyzer.anvil.app/**



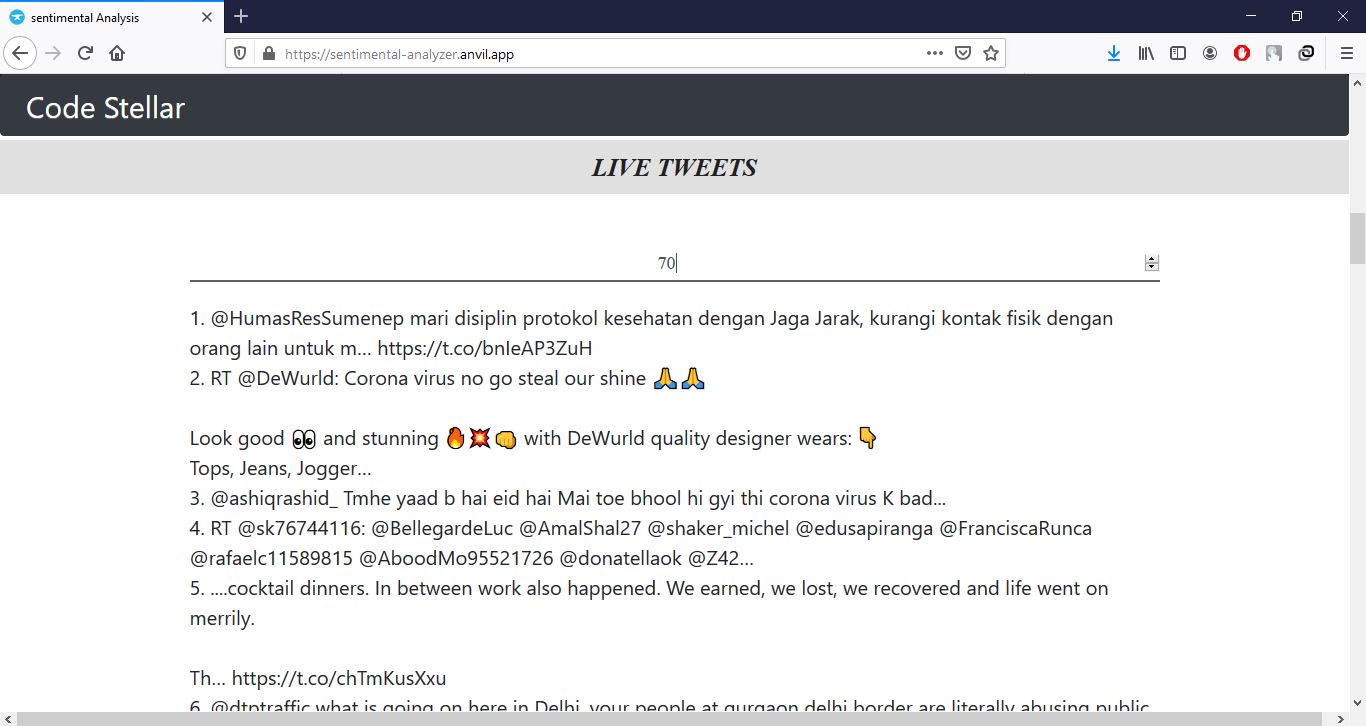
2. Enter a Tweet



3. View Sentiment Prediction Graph.

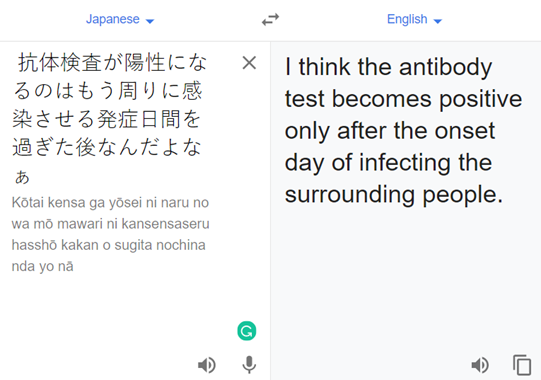


4. View Live Tweets.



5. Multi Language Acceptor.





*7. ADVANTAGES & DISADVANTAGES*

Advantages:

1. It Predicts the result Correctly.

2. It can be used to find the sentiment across the globe for an issue with the help of words.

3. It shows the sentiment in form of graph which is easy for access for everybody.

4. Instant results.

5. Multiple Language Supporter.

6. Fast in Processing.

Disadvantages:

1. It takes lot of time for training the model.

2. Anvil App needs to be paid for complete access.

3. Small errors occur due to large data processing.

4. Can run only when the trained model is running.

5. Entered Input should not have spelling errors.

8. APPLICATIONS

The Sentimental Analyzer Web App can be used in many places. It can be used to find the sentiment of any sentence in an instant and can help to understand the feelings of people.

If it is connected with Twitter then for every tweet it can display the sentiment next to itself and it will help a lot.

Government after making any decisions can take tweets related to that and learn from it, what to be done.

9. CONCLUSION

This project showed the working and usage of Notebook and UI. It showed how with the help of a tweet we can understand the feeling of peoples across the world. It also showed which feeling is the maximum currently across the globe.

In short, it process all tweets and gives the user what the user wants or wishes to see.

10. FUTURE SCOPE

This project will further be improved to get trained well by adding more datas, and will be made to do prediction more accurately. This will be made as a web application and a mobile application that can be accessed by any layman in any manner. For further applications it will be able to display more than 300 tweets and will store the old Tweets in a database for accessing.

11. BIBILOGRAPHY

***Name 1: Maaz Hussain***

***College : RMK Engineering College***

***Name 2: Prajeshwar***

***College : RMK Engineering College***

***Topic : Sentimental Analysis of Covid 19 Tweets.***

***Refrences :***

<https://www.kaggle.com/smid80/coronavirus-covid19-tweets>  
<https://ieee-dataport.org/open-access/corona-virus-covid-19-tweets-dataset>  
 <http://help.sentiment140.com/for-students>

[https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-w ords-works/](https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/)

<https://medium.com/@canerkilinc/padding-for-nlp-7dd8598c916a>

[https://subscription.packtpub.com/book/big\_data\_and\_business\_int elligenc e/9781787128422/3/ch03lvl1sec24/deep-convolutional-neural-network-d cnn](https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781787128422/3/ch03lvl1sec24/deep-convolutional-neural-network-dcnn)

https://anvil.works/

***12. APPENDIX***

Link To Sentimental Analyser Anvil App:

**https://sentimental-analyzer.anvil.app/**

Link To Git Hub Code:

**https://github.com/SmartPracticeschool/SBSPS-Challenge-3446-Sentiment-Analysis-of-COVID-19-Tweets**

13. SOURCE CODE

PARTS OF THE CODE WITH EXPLANATION

1. IMPORTING DEPENDENCIES

1. import numpy as np
2. import math
3. import re
4. import time
5. import pandas as pd
6. from bs4 import BeautifulSoup
7. import types
8. from botocore.client import Config
9. import ibm\_boto3
10. try:
11. %tensorflow\_version 2.x
12. except:
13. pass
14. import tensorflow as tf
16. from tensorflow.keras import layers
17. import tensorflow\_datasets as tfds

The Following Modules are imported for the working of the sentimental analyzer.Each of the module imported has its own purpose in the working of the analyzer.

numpy is used for formatting arrays.

math is used to perform mathematical calculations.

re is used in Cleaning the datasets.

pandas is used to read csv files.

bs4 is used to clean the datasets from , unwanted characters.

Types,config and ibm\_boto3 is used to get the input of datasets from the IBM cloud.

tensorflow,layers is used to train the model

tensorflow-datasets is used to tokenize & train the datasets.

2. IMPORTING DATASETS

1. cols=["sentiment","text"]
2. def \_\_iter\_\_(self): return 0
3. client\_470a2373ed9f45e5bfa9ab0a46f242bd = ibm\_boto3.client(service\_name='s3', ibm\_api\_key\_id='raGZamx8niNdqewOA3CblpHzFKoJEiZDQav40LRsU1he',
4. ibm\_auth\_endpoint="https://iam.cloud.ibm.com/oidc/token",
5. config=Config(signature\_version='oauth'), endpoint\_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')
6. body = client\_470a2373ed9f45e5bfa9ab0a46f242bd.get\_object(Bucket='sentimentalanalysisofcovidtweets-donotdelete-pr-ymk9ojnpvn3l4y',Key='test.csv')['Body']
7. # add missing \_\_iter\_\_ method, so pandas accepts body as file-like object
8. if not hasattr(body, "\_\_iter\_\_"): body.\_\_iter\_\_ = types.MethodType( \_\_iter\_\_, body )
9. test\_data = pd.read\_csv(body,
10. header=None,
11. names=cols,
12. encoding="latin1"
13. )
14. test\_data.head()
15. body = client\_470a2373ed9f45e5bfa9ab0a46f242bd.get\_object(Bucket='sentimentalanalysisofcovidtweets-donotdelete-pr-ymk9ojnpvn3l4y',Key='train.csv')['Body']
16. # add missing \_\_iter\_\_ method, so pandas accepts body as file-like object
17. if not hasattr(body, "\_\_iter\_\_"): body.\_\_iter\_\_ = types.MethodType( \_\_iter\_\_, body )
18. train\_data = pd.read\_csv(body,
19. header=None,
20. names=cols,
21. encoding="latin1"
22. )
23. train\_data.head()
24. data=train\_data

The Above code is used to extract two datasets Train.csv and Test.csv from the IBM cloud. The api key and all the other authentication end points is given, so that the datasets are imported into the code, the two imported data sets are stored inside the train\_data and Test\_data variables.

The data sets contains two columns namely sentiment and text, where sentiment column will contain values from 0-1 in train.csv and 0-4 in test.csv, which will be processed later. The text column contain tweets from various users and responding sentiment is also present, these tweets come in text field, which will be cleaned in the next step to get clear datas.

3. CLEANING

1. def clean\_tweet(tweet):
2. tweet = BeautifulSoup(tweet, "lxml").get\_text()
3. tweet = re.sub(r"@[A-Za-z0-9]+",' ', tweet)
4. tweet = re.sub(r"https?://[A-Za-z0-9./]+",' ', tweet)
5. tweet = re.sub(r"[^a-zA-Z.!?']",' ', tweet)
6. tweet = re.sub(r" +"," ", tweet)
7. return tweet
8. data\_clean=[clean\_tweet(tweet) for tweet in data.text]
9. data\_labels=data.sentiment.values
10. data\_labels[data\_labels==4]=1
11. set(data\_labels)

This Code takes the datasets tweets as input and process it to remove all unwanted characters , and unwanted digits and make it pure words.

re.sub is used to substitute the datas containing https to a space and similarly @ symbols and have only characters.

Then after cleaning is done we take the sentiment valus and make it to 0 and 1 as we need details between them only.

The final set(data\_labels) will print 0 and 1 as output.

data\_clean contains the clean tweets for machine to learn.

4. TOKENIZATION

1. tokenizer= tfds.features.text.SubwordTextEncoder.build\_from\_corpus(
2. data\_clean,target\_vocab\_size=2\*\*16
3. )
4. data\_inputs=[tokenizer.encode(sentence) for sentence in data\_clean]

The tfds module is used to encode the tweets into seperate words and store in data inputs to be processed later. The whole lot of inputs is seperated into different words and are stored in data\_inputs.

5. PADDING

1. MAX\_LEN=max([len(sentence) for sentence in data\_inputs])
2. data\_inputs=
3. tf.keras.preprocessing.sequence.pad\_sequences(
4. data\_inputs, value=0, padding="post", maxlen=MAX\_LEN)

To make the sentences of equal lengths we need to do padding. Here MAX\_LEN holds the maximum length of the data inputs and then we use keras in tensorflow to do padding to all other words and append them with 0 and padding is post which implies it is appended at the end.

6. SPLITTING INTO TRAINING / TESTING SETS.

1. test\_idx = np.random.randint(0, 800000, 8000)
2. text\_idx = np.concatenate((test\_idx,test\_idx+800000))
3. test\_inputs = data\_inputs[test\_idx]
4. test\_labels = data\_labels[test\_idx]
5. train\_inputs = np.delete(data\_inputs,test\_idx,axis=0)
6. train\_labels = np.delete(data\_labels,test\_idx)

This part of the code makes two values of train\_inputs which is used to train the model by comparing it with test inputs where the model gets trained well in Model Building.

np.random.randint takes random index value for test index and it is deleted from train inputs and train labels to make the model work effectively.

7. DCNN KERAS MODEL

1. class DCNN(tf.keras.Model):
2. def \_\_init\_\_(self,
3. vocab\_size,
4. emb\_dim=128,
5. nb\_filters=50,
6. FFN\_units=512,
7. nb\_classes=2,
8. dropout\_rate=0.1,
9. training=False,
10. name="dcnn"):
11. super(DCNN, self).\_\_init\_\_(name=name)
12. +
13. self.embedding=layers.Embedding(vocab\_size,emb\_dim)
14. self.bigram= layers.Conv1D(filters=nb\_filters,
15. kernel\_size=2,
16. padding="valid",
17. activation="relu")
18. self.pool\_1 = layers.GlobalMaxPool1D()
19. self.trigram= layers.Conv1D(filters=nb\_filters,
20. kernel\_size=3,
21. padding="valid",
22. activation="relu")
23. self.pool\_2 = layers.GlobalMaxPool1D()
24. self.fourgram= layers.Conv1D(filters=nb\_filters,
25. kernel\_size=4,
26. padding="valid",
27. activation="relu")
28. self.pool\_3 = layers.GlobalMaxPool1D()
29. self.dense\_1= layers.Dense(units=FFN\_units,activation="relu")
30. self.dropout= layers.Dropout(rate=dropout\_rate)
31. if nb\_classes == 2:
32. self.last\_dense= layers.Dense(units=1,
33. activation="sigmoid")
34. else:
35. self.last\_dense= layers.Dense(units=nb\_classes,
36. activation="softmax")
37. def call(self, inputs,training):
38. x= self.embedding(inputs)
39. x\_1=self.bigram(x)
40. x\_1=self.pool\_1(x\_1)
41. x\_2=self.trigram(x)
42. x\_2=self.pool\_1(x\_2)
43. x\_3=self.fourgram(x)
44. x\_3=self.pool\_1(x\_3)
45. merged = tf.concat([x\_1,x\_2,x\_3], axis=-1) # (batch\_size, 3 \* nb\_filters)
46. merged = self.dense\_1(merged)
47. merged = self.dropout(merged, training)
48. output = self.last\_dense(merged)
49. return output

This function is the main part of the project, here is where the model gets trained well to make itself into fully functioning one.

Vocab size, number of classes, FFN units, number of filters and embedded dimension are set to default values. These values are passed to the macine and Three layers are made namely bigram, trigram, and fourgram where kernel size varies from 2,3 and 4.

Dropout\_rate variable contains the drop out rate, Dense variable contains the FFN units.

Finally, A function call () is used to call all the methods one by one and the model is thus trained one by one, after that the trained layers are stored under a variable named merged, by using tensorflow. Then

the merged layer is trained under dense\_1 method where FFN units are added to it. Then the model is again trained with dropout rate, finally it is trained under the last\_dense for one time or number of classes, depending on NB\_classes.

Thus, the trained model is returns the output to the called function.

8. CONFIGURATION

1. VOCAB\_SIZE = tokenizer.vocab\_size
2. EMB\_DIM = 200
3. NB\_FILTERS = 100
4. FFN\_UNITS = 256
5. NB\_CLASSES = len(set(train\_labels))
6. DROPOUT\_RATE = 0.1
7. BATCH\_SIZE = 32
8. NB\_EPOCHS = 10

This part of the code contains default values for all training sets, By default the vocabulary size is set to tokenizer.vocabsize which holds 2^16, similarly the embedded dimension is given 200 , 100 filters are kept and 256 FFN units are kept. dropout rate shows the speed of the model to train itself. The number of Epochs shows how many times should the model be trained the more the epochs value the better the efficiency.

9. TRAINING

1. Dcnn = DCNN(vocab\_size=VOCAB\_SIZE,
2. emb\_dim=EMB\_DIM,
3. nb\_filters=NB\_FILTERS,
4. FFN\_units=FFN\_UNITS,
5. nb\_classes=NB\_CLASSES,
6. dropout\_rate=DROPOUT\_RATE
7. )
8. if NB\_CLASSES==2:
9. Dcnn.compile(loss="binary\_crossentropy",
10. optimizer="adam",
11. metrics=["accuracy"])
12. else:
13. Dcnn.compile(loss="sparse\_categorical\_crossentropy",
14. optimizer="adam",
15. metrics=["sparse\_categorical\_accuracy"])
16. Dcnn.fit(train\_inputs,
17. train\_labels,
18. batch\_size=BATCH\_SIZE,
19. epochs=NB\_EPOCHS
20. )

Finally using all the default set values the model is all set to be trained. DCNN method is called by passing all the train inputs and train labels batch size and epochs to know how many times it should train the model, thus it passes the control to DCNN function and it returns the output from that function.

10. RESTORING

1. checkpoint\_path="./ckpt/"
2. ckpt= tf.train.Checkpoint(Dcnn=Dcnn)
3. ckpt\_manager = tf.train.CheckpointManager(ckpt,checkpoint\_path, max\_to\_keep=1)
4. if ckpt\_manager.latest\_checkpoint:
5. ckpt.restore(ckpt\_manager.latest\_checkpoint)
6. print("Latest Checkpoint Restored")

This part of the the code is used to check if the trained model is already available in the watson studio, If the trained model is found then we use ckpt.restore to get back the trained data. This will help us to reduce the burden of often training the model again and again and spend hours for it.

11. USER INTERFACE.

1. <!DOCTYPE html>
2. <html>
3. <head>
4. <title>Signin</title>
5. <meta charset="utf-8">
6. <meta name="viewport" content="width=device-width, initial-scale=1">
7. <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8. <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>
9. <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10. </head>
11. <style>
12. body{
13. overflow-x: hidden;
14. }
15. .main-content{
16. width: 50%;
17. height: 40%;
18. margin: 10px auto;
19. background-color: #fff;
20. border: 2px solid #e6e6e6;
21. padding: 40px 50px;
22. }
23. .header{
24. border: 0px solid #000;
25. margin-bottom: 5px;
26. }
27. .well{
28. background-color: #000000;
29. }
30. #live\_stream\_btn{
31. width: 60%;
32. background-color: #051094;
33. border-radius: 30px;
34. }
35. #old\_tweets\_btn{
36. width: 60%;
37. background-color: #051094;
38. border-radius: 30px;
39. }
40. #user\_input\_btn{
41. width: 60%;
42. background-color: #051094;
43. border-radius: 30px;
44. }
45. .overlap-text{
46. position: relative;
47. }
48. .overlap-text a{
49. position: absolute;
50. top: 8px;
51. right: 10px;
52. font-size: 14px;
53. text-decoration: none;
54. font-family: 'Overpass Mono', monospace;
55. letter-spacing: -1px;
56. }
57. .sidebar {
58. height: 100%;
59. width: 0;
60. position: fixed;
61. z-index: 1;
62. top: 0;
63. left: 0;
64. background-color: #111;
65. overflow-x: hidden;
66. transition: 0.5s;
67. padding-top: 60px;
68. }
69. .sidebar a {
70. padding: 8px 8px 8px 32px;
71. text-decoration: none;
72. font-size: 25px;
73. color: #818181;
74. display: block;
75. transition: 0.3s;
76. }
77. .sidebar a:hover {
78. color: #f1f1f1;
79. }
80. .sidebar .closebtn {
81. position: absolute;
82. top: 0;
83. right: 25px;
84. font-size: 36px;
85. margin-left: 50px;
86. }
87. .openbtn {
88. font-size: 20px;
89. cursor: pointer;
90. background-color: #111;
91. color: white;
92. padding: 10px 15px;
93. border: none;
94. }
95. .openbtn:hover {
96. background-color: #444;
97. }
98. #main {
99. transition: margin-left .5s;
100. padding: 16px;
101. }
102. @media screen and (max-height: 450px) {
103. .sidebar {padding-top: 15px;}
104. .sidebar a {font-size: 18px;}
105. }
106. </style>
107. <body>
108. <div id="mySidebar" class="sidebar">
109. <a href="javascript:void(0)" class="closebtn" onclick="closeNav()">×</a>
110. <a href="https://node-red-maaz.mybluemix.net/livestream.html">Live Stream Tweets</a>
111. <a href="https://node-red-maaz.mybluemix.net/oldtweet.html">Old Tweets</a>
112. <a href="https://node-red-maaz.mybluemix.net/userinput.html">User Input</a>
113. <a href="#">About Us</a>
114. </div>
115. <div class="row">
116. <div class="col-sm-12">
117. <div class="well">
118. <div class="main">
120. </div></div>
121. </div>
122. </div>
123. <div class="row">
124. <div class="col-sm-12">
125. <div class="well">
126. <center><h1 style="color: white;">Sentimental Analyzer.</h1></center>
127. </div>
128. </div>
129. </div>
130. <h2><b>User Input</b></h2>
131. <br>
132. <center><button id="user\_input\_btn" type="button" onclick="getInputValue();">Get Results :</button></center>
133. </body></html>

The above HTML Code is used to make a User Interface, to link an easy access for the users to use the Sentimental Analyzer. This code is Pasted in Anvil App for connecting UI with Notebook, and thus show the desired output.

***THANK YOU***