CSCE 5290
Natural Language Processing
FINAL PROJECT

GithubLink:

https://github.com/nehareddy9399/NLP-Projectgroup11

1.Project Title and Members:

Project Title: Optical Character Recognition and Sentiment Analysis

Members: (Team 11)

Keerthana Pinikeshi (Student ID - 11448714)

Neha Kalluri (Student ID - 11445206)

Lahari Kunduru (Student ID - 11445208)

2.Goals and Objectives:

Motivation:

As part of this NLP project, we thought of building something that is very close to the real world and widely used in NLP applications. We found that Optical Character Recognition (OCR) is one of the most popular area of research in NLP applications. We as a team felt that Optical Character Recognition problem statement will be used by all kinds individuals in their day to day life. Hence we have worked on Optical Character Recognition model to recognize the text in the given input image and perform sentimental anlayis on the text we extracted as positive or negative.

Significance :

The major significance of Optical Character Recognition (OCR) is used in almost all kinds of industries and individuals. Before the advancements in the OCR, we used to look at the hard copies of documents and type the content manually. But now we can process any document or image to extract the text from the OCR. It saves huge time and manpower in almost every industry and every individuals life.

OCR technology is used very often in real world applications. Some of the applications of the OCR is Listed below.

1. Data entry of documents
2. Data extraction from KYC documents
3. Number plate recognition
4. Self Drving Cars
5. Hand Written text recognition.
6. Scanned documents to Searchable PDFs
7. CAPTCHA Recognition

Objectives:

The main objective of our project is to build a real world OCR model to recognize the text in the image. As we have researched that , to build a OCR model which recognizes the whole text in the image is not scale-able and required huge computation power along with powerful models and datasets. We have also researched that most of the OCR products performs word by word extraction and concatenates the words to get the whole text extraction. Hence we are going to build the OCR model for the extraction of the single word.

In order to build the OCR model, we have some objectives to follow. They are :

- **<u>Data Collection :</u>** We have taken the dataset that consists of number of gray scale word level images. It consists of around ninety thousand images with different English words. From that huge set of data we have taken only 150000 images as our dataset separately. Again from those images we considered 90% of them as our training data and rest of them as our test data. For Sentiment Analysis we took senetence level images and performed sentiment analysis for the sentences.

- **<u>Data Cleaning :</u>** It is very important clean the data after collection. Sometimes there could be wrong annotations in the dataset. More the time we spend on data cleaning , more the quality results we can get. In Data Cleaning step we check if our image shape is 128x32. By doing that we maintain uniformity for our dtaaset. We skip images which do not match this image shape. The next step will be Data Preprocessing.

- **<u>Data Preprocessing :</u>** In this phase, we will take the image and perform some preprocessing techniques like data augmentation, rotation, adding noise etc. From our dataset taken we consider 90% of the images as our training data and remaining as our test data. The word in the image will be same as the filename which we can use as labels. For that we are extracting the labels from the filename of the image as assigning labels for our train data.

- **<u>Model Building :</u>** Since the dataset is in the image format, it required to build a deep neural network model instead of machine learning models. We need to build a suitable model architecture. For that we are using Convolutional Layer which uses a filter that is applied on the original image to get a feature map which helps to identify the image more conveniently. In the BatchNormalization layer we normalize the output obtained from the previous layer and train the model which helps in aligning features together. By using Bi-directional LSTM, we consider both the previous and next word to understand the context in which the word is used. Dense layers used the output from all its previous layers and classifies the output. It depends on output of convolutional layer. From transformers package we use pipeline() method to classify the output generated i.e., the text extracted from images as either positive or negative.

- **<u>Evaluation of results :</u>** In section , we evaluate the results by checking the value loss and plotting the image and it's recognized text. We will also predict to which class does the text extracted will belong to.

Features :

In general, to train a deep learning model, we require GPU computing power. By using Google colabaratory we can get the free GPU for limited time per day. Colab’s computer power will be sufficient to train OCR model. Colab also provides the option to mount the google drive. Since we are going with huge dataset, we can upload the dataset to the google drive once and we can easily able to mount the drive and use it for the model training.

Since we are training the OCR model from scratch, We require a huge dataset. We would like to choose the dataset that contains at-least 100,000 image samples for training the model. It is also important to pick the dataset with good accurate annotations.

Some of the important modules are required for this project are:

1.Tensorflow

2.Keras

3.Numpy

4.Opencv

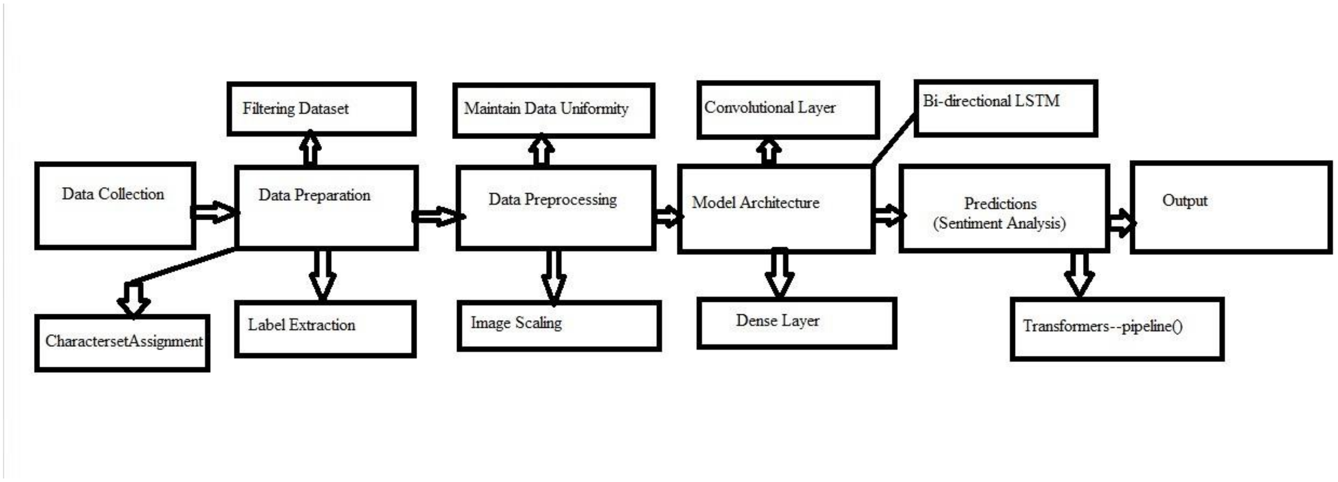
5.Matplotlib

6.Transformers Etc.

Unlike the classical structured datasets, we have image dataset as input. Hence we wont be having the numeric or text related columns. But any model cannot process an image directly to compute the predictions. We first need to perform the featurization. Featurzation is nothing but to convert the image into an n-dimensional vector and we pass that n-dimensional vector to the neural network in order to get the predictions.Typically we convert the image into numpy array.OpenCV makes use of Numpy for numerical operations with a MATLAB style syntax. All the OpenCV array structures are converted to and from Numpy arrays.OpenCv is generally used for image processing.By using OpenCV we can process images and videos to identify objects, faces, text .When OpenCV is integrated with other libraries such as Numpy, then python will be able to process the OpenCV array structure for analysis.From Transformers package we will use the pipeline() for completing majority of tasks.

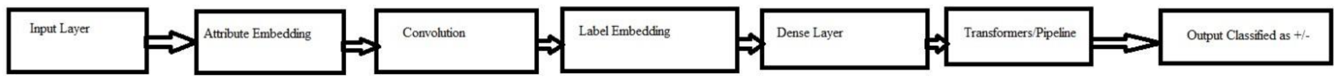
Project Workflow :

The below screenshot is the workflow of our project. As we can see the first step we have done is dataset collection and then we are doing data preparation by filtering the data, defining the character set and converting them to id's and label extraction. As part of Data preprocessing, we have rescaled all the images to 128x32. For the model we have used convolutional layers, Bi-directional LSTM and dense layers. Using all these layers, we are extracting the text from images. Then we are passing the text to piprline() class to classify the text as positive or negative.



Project Architecture:

For building our model we have used convolutional recurrent netwroks for extracting the text from image. Convolutional Recurrent layer uses multiple layers like input layer, convolutional layer, bi-directional LSTM and dense layer. Output from each layer is passed as input to the second layer. After text extraction is done, we are installing transformers package and using pipeline class from it to do the sentiment analysis.



3. Related Work (Background)

When it comes to optical character recognition (OCR) there are mainly two kinds of approaches that are available currently.

A) There are some freely available python packages by using which we can perform OCR, but more often we will get less accuracy. Some example packages are: 1) Pytessract and 2) Paddleocr Etc.,

B) We need to train deep learning model from scratch. For optical character recognization most popular deep learning architecture is Convolution recurrent neural network.

C) We are using pipeline() method from transformers package which contains pre-trained models for classification, sentiment analysis. As of now, we are doing binary classification.

4. <u>Dataset</u>

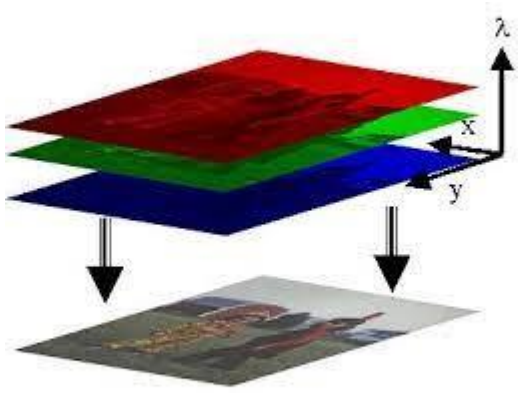
In order to find the relevant good dataset we did lot of research. Most of the available datasets in the internet are having lot of noisy data and blurry images. Finally we found a good dataset. It has with huge number of gray scale word level images. The link to the dataset is available in the below.
https://www.robots.ox.ac.uk/~vgg/data/text/#sec-synth

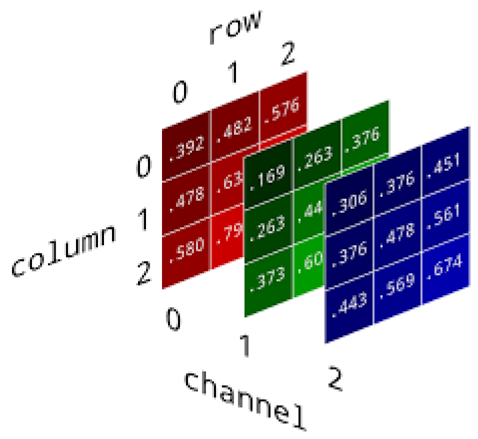
If you visit the above link you can find the download the synthetic English word dataset.. IN the dataset they have more than 9 Million images and 90k English words. This data is very huge and it consumes lot of time and space complexity. Hence we downloaded the whole dataset and filtered around 150,000 images. We are using the same dataset for both training and testing.

5. <u>Detail design of Features</u>

If we have text dataset they are lot if featurization techniques available. Since we are dealing with the image dataset here we have to convert the given image into a multi dimensional numpy array. In general a coloured image contains 3 layers RGB with shape (height, width,3). But in our training dataset we have gray scale image. Hence the dataset shape will be (height, width)

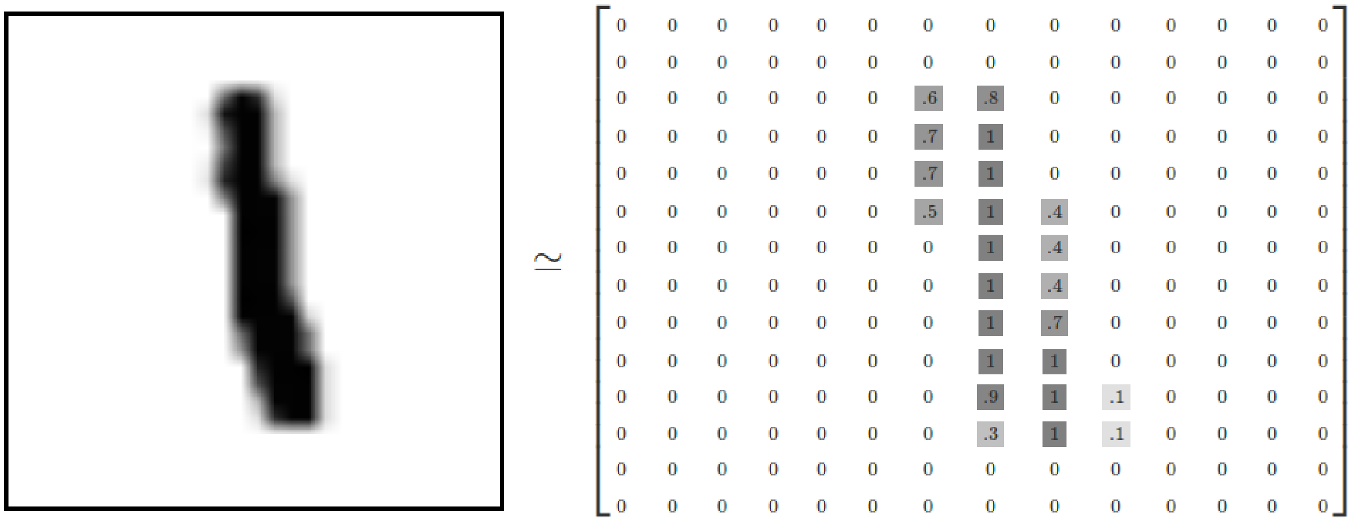
<u>RGB Image Representation in Numpy array</u>





<u>Gray Scale to array representation:</u>

The below image represents the featurization of the Gray Scale image to Numpy array. When an image is represented as array, the X-asis coordinates will ve the first axis and secind axis will be the Y-axis coordinates.

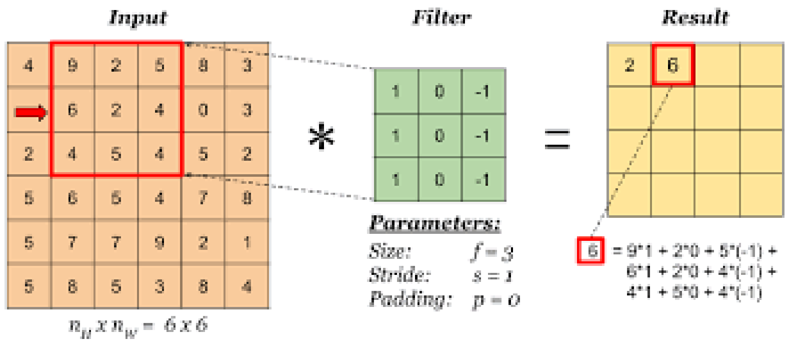


6. Analysis

As we already mentioned we are using convolution recurrent neural network. It contains mainly two parts 1) Convolution and 2) Recurrent neural network 3)Transformers for Sentiment Analysis.

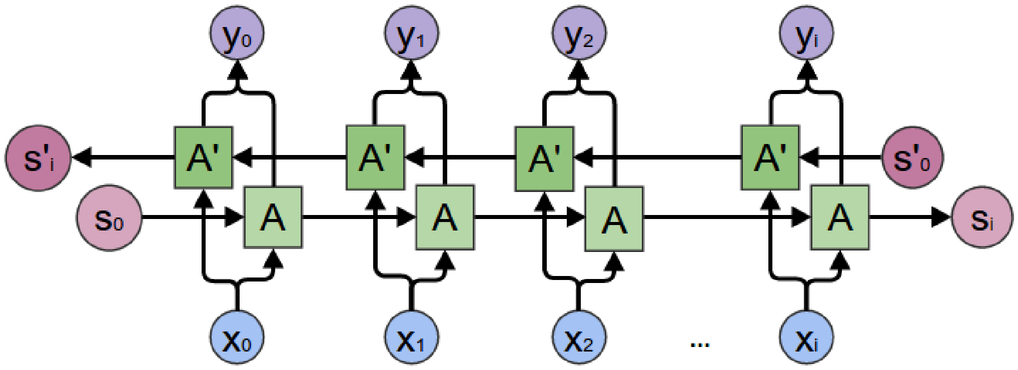
1) Convolution:

The convolution is an operation that mainly helps to understand the shapes in the given image. In simple words it will remove the unnecessary objects in the image and focuses more on the character/ Object in the image.



2) Recurrent neural network:

It is a very popular and widely used networks when it comes to dealing with NLP tasks. Recurrent neural got popular when the research paper called attention all you need is published. Recurrent neural network basically focus on both previous and upcoming characters (Bi directional LSTM). The below screenshot is the architecture of Bi-directional LSTM.

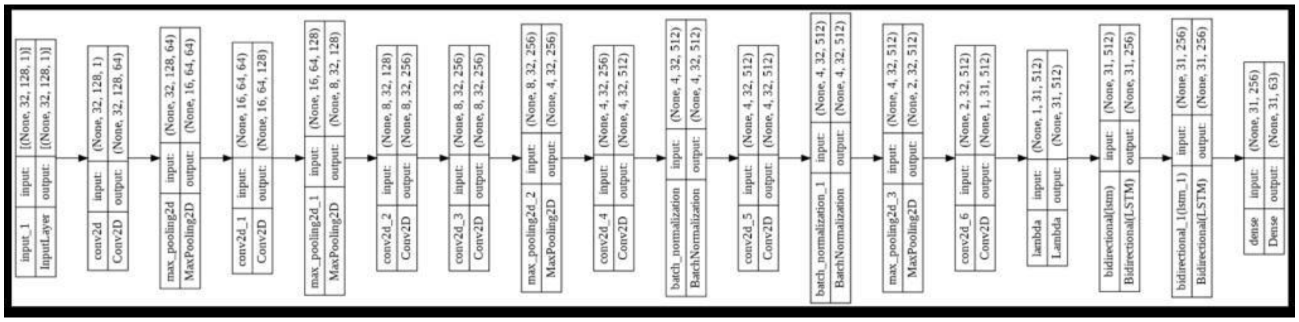


3) Transformers for Sentiment Analysis:

Transformers library consists of around thirty pre-trained models which we can use for text classification, text summarization, sentiment analysis. From this transformers package we will use the pipeline() for completing majority of tasks. Firstly we need to install the transformers package and then we need to import the pipeline() class. We are using this class for our model that is already trained. When using pipeline() class we have to specify the model you are using and what tokenizer you are using. Here, we are using pipeline() class for sentiment analysis and the text provided to this will be classified as either positive or negative class and it also gives the confidence score.

Graph model :

We have plotted summary of our layers as a graph which shows the input dataset for the layers and the how the data is taken to layers. We can see in the below screenshot, each layer and how the image is converted as text is extracted.



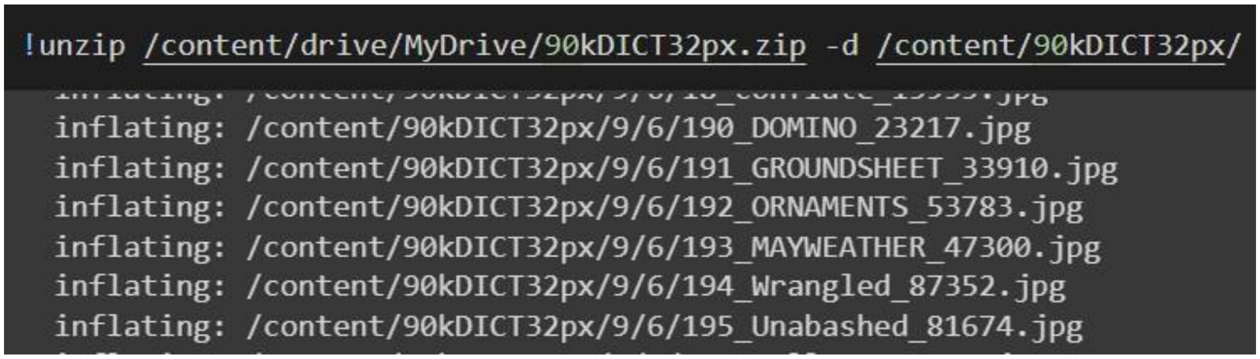
input_1 | InputLayer | input: | output: | [(None, 32, 128, 1)] | [(None, 32, 128, 1)]
conv2d | Conv2D | input: | output: | (None, 32, 128, 1) | (None, 32, 128, 64)
max_pooling2d | MaxPooling2D | input: | output: | (None, 32, 128, 64) | (None, 16, 64, 64)
conv2d_1 | Conv2D | input: | output: | (None, 16, 64, 64) | (None, 16, 64, 128)
max_pooling2d_1 | MaxPooling2D | input: | output: | (None, 16, 64, 128) | (None, 8, 32, 128)
conv2d_2 | Conv2D | input: | output: | (None, 8, 32, 128) | (None, 8, 32, 256)
conv2d_3 | Conv2D | input: | output: | (None, 8, 32, 256) | (None, 8, 32, 256)
max_pooling2d_2 | MaxPooling2D | input: | output: | (None, 8, 32, 256) | (None, 4, 32, 256)
conv2d_4 | Conv2D | input: | output: | (None, 4, 32, 256) | (None, 4, 32, 512)
batch_normalization | BatchNormalization | input: | output: | (None, 4, 32, 512) | (None, 4, 32, 512)
conv2d_5 | Conv2D | input: | output: | (None, 4, 32, 512) | (None, 4, 32, 512)
batch_normalization_1 | BatchNormalization | input: | output: | (None, 4, 32, 512) | (None, 4, 32, 512)
max_pooling2d_3 | MaxPooling2D | input: | output: | (None, 4, 32, 512) | (None, 2, 32, 512)
conv2d_6 | Conv2D | input: | output: | (None, 2, 32, 512) | (None, 1, 31, 512)
lambda | Lambda | input: | output: | (None, 1, 31, 512) | (None, 31, 512)
bidirectional(lstm) | Bidirectional(LSTM) | input: | output: | (None, 31, 512) | (None, 31, 256)
bidirectional_1(lstm_1) | Bidirectional(LSTM) | input: | output: | (None, 31, 256) | (None, 31, 256)
dense | Dense | input: | output: | (None, 31, 256) | (None, 31, 63)

7. <u>Implementation</u>

Here in this section we have mainly 4 Modules.

1) Loading dataset into Google Colab.

As we already discussed we have filtered approximately 150,000 images and created a Zip File for those images. The best suggested way is to upload the corresponding Zip file into google drive and we can access to the Zip file by mounting the google drive using google colabaratory. It is easy to un-zip a Zip file it just requires a simple Linux Command



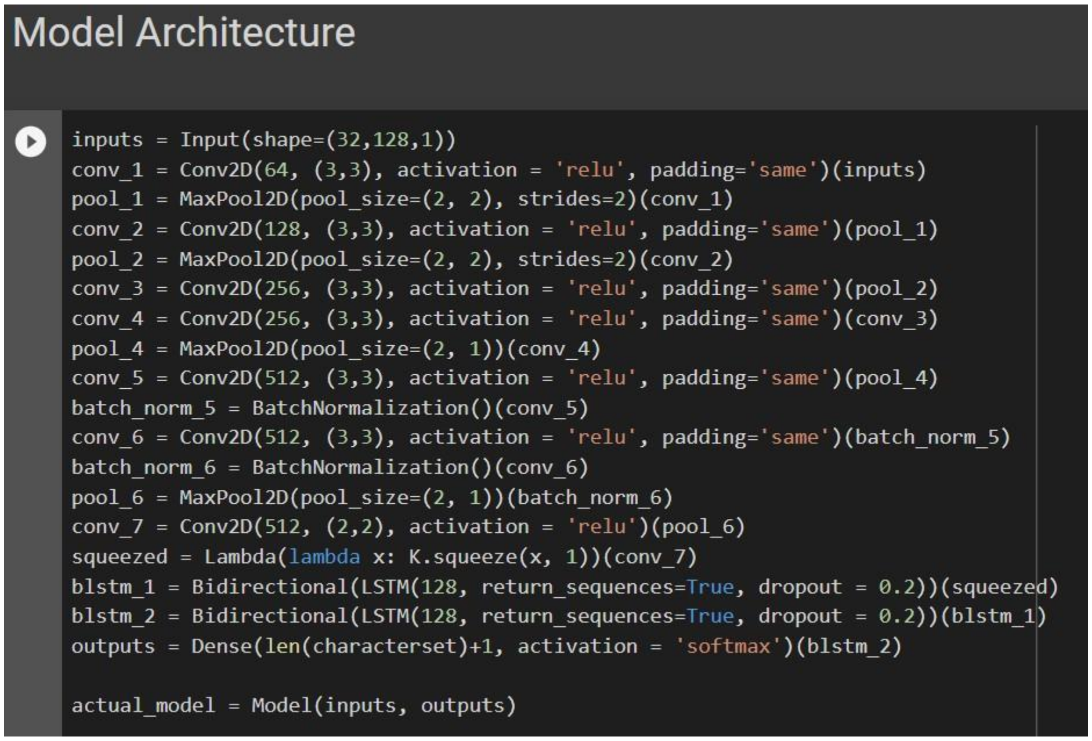
```
!unzip /content/drive/MyDrive/90kDICT32px.zip -d /content/90kDICT32px/
  inflating: /content/90kDICT32px/9/6/190_DOMINO_23217.jpg
  inflating: /content/90kDICT32px/9/6/191_GROUNDSHEET_33910.jpg
  inflating: /content/90kDICT32px/9/6/192_ORNAMENTS_53783.jpg
  inflating: /content/90kDICT32px/9/6/193_MAYWEATHER_47300.jpg
  inflating: /content/90kDICT32px/9/6/194_Wrangled_87352.jpg
  inflating: /content/90kDICT32px/9/6/195_Unabashed_81674.jpg
```

2) Data Preprocessing:

In this section there are some necessary prepossessing steps required to get the important variables like label, array representation of the image, input length , padding the label etc. If you observe the above screenshot, the label name is present in the file name itself. Hence we can extract the label name using “filename.split(“_”)[1]”. This will return the label for the corresponding image.

3) Model Building:

As discussed earlier, we are using the Convolution Recurrent Neural Network for our model building. We have tried several combinations of layers of both convolution and Bi-directional LSTM layers. We have chosen a particular model which gives us the minimal loss. In our model we have used the 7 convolution layers and 2 bi-directional LSTM layers. Convolutional layers are capable of extracting various image features such as edges, textures, objects, and scenes . As a result, low-level features are essential for identifying manipulated regions. The filters in the convolutional layer will generate feature maps that are linked to the previous layer's local region.
Bidirectional long-short term memory (bi-lstm) is the method of enabling any neural network to store sequence information in both directions, forward and backward.BiLSTMs increase the amount of information available to the network, improving the algorithm's context.
Batch normalization is a technique for standardizing network inputs that can be applied to either prior layer activations or direct inputs. Batch Normalization helps to speed up training by halving the epochs and provides some regularization, reducing generalization error.Batch normalization is a training technique for deep neural networks that standardizes the inputs to a layer for each mini-batch. This stabilizes the learning process and significantly reduces the number of training epochs required to train deep networks.
A dense layer is one that is deeply connected to the layer before it, which means that the neurons in the layer are connected to every neuron in the layer before it.The below screenshot represents the model that gives best result:

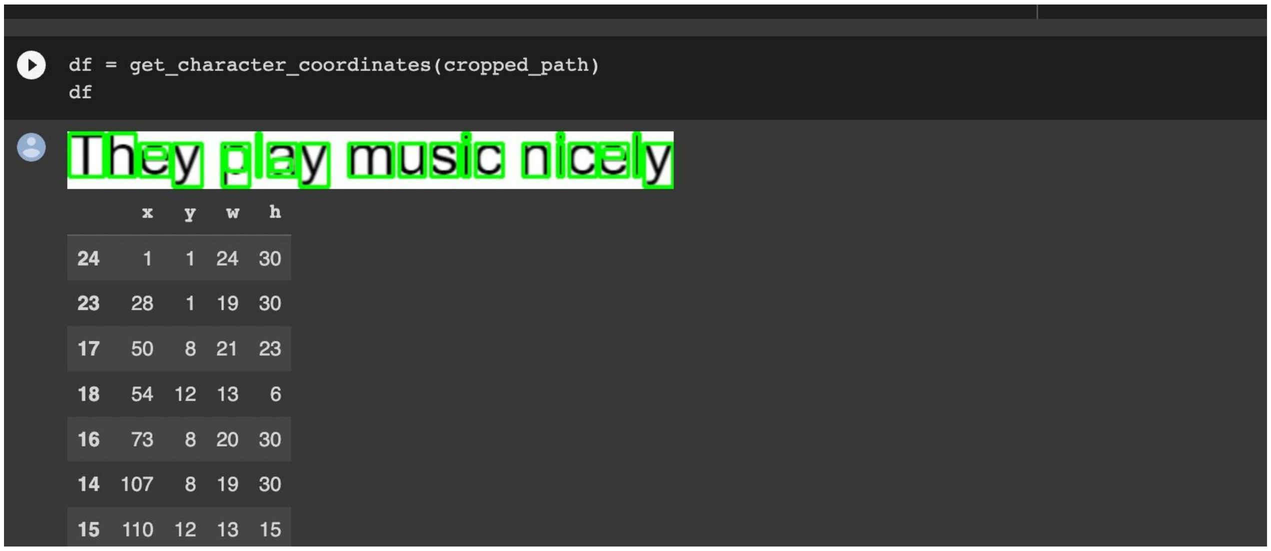


Model Architecture

```
inputs = Input(shape=(32,128,1))
conv_1 = Conv2D(64, (3,3), activation = 'relu', padding='same')(inputs)
pool_1 = MaxPool2D(pool_size=(2, 2), strides=2)(conv_1)
conv_2 = Conv2D(128, (3,3), activation = 'relu', padding='same')(pool_1)
pool_2 = MaxPool2D(pool_size=(2, 2), strides=2)(conv_2)
conv_3 = Conv2D(256, (3,3), activation = 'relu', padding='same')(pool_2)
conv_4 = Conv2D(256, (3,3), activation = 'relu', padding='same')(conv_3)
pool_4 = MaxPool2D(pool_size=(2, 1))(conv_4)
conv_5 = Conv2D(512, (3,3), activation = 'relu', padding='same')(pool_4)
batch_norm_5 = BatchNormalization()(conv_5)
conv_6 = Conv2D(512, (3,3), activation = 'relu', padding='same')(batch_norm_5)
batch_norm_6 = BatchNormalization()(conv_6)
pool_6 = MaxPool2D(pool_size=(2, 1))(batch_norm_6)
conv_7 = Conv2D(512, (2,2), activation = 'relu')(pool_6)
squeezed = Lambda(lambda x: K.squeeze(x, 1))(conv_7)
blstm_1 = Bidirectional(LSTM(128, return_sequences=True, dropout = 0.2))(squeezed)
blstm_2 = Bidirectional(LSTM(128, return_sequences=True, dropout = 0.2))(blstm_1)
outputs = Dense(len(characterset)+1, activation = 'softmax')(blstm_2)

actual_model = Model(inputs, outputs)
```

4) Sentiment Analysis:

As part of sentiment analysis, we have taken a few images as our dataset with sentences in it. Then we divide the sentence to words based on the pixel distance like if the distance between two characters is less than 20 then we consider it as a single word and if the pixel distance is greater than twenty we consider it as separate words. Based on the words the entire image is cropped as individual images.

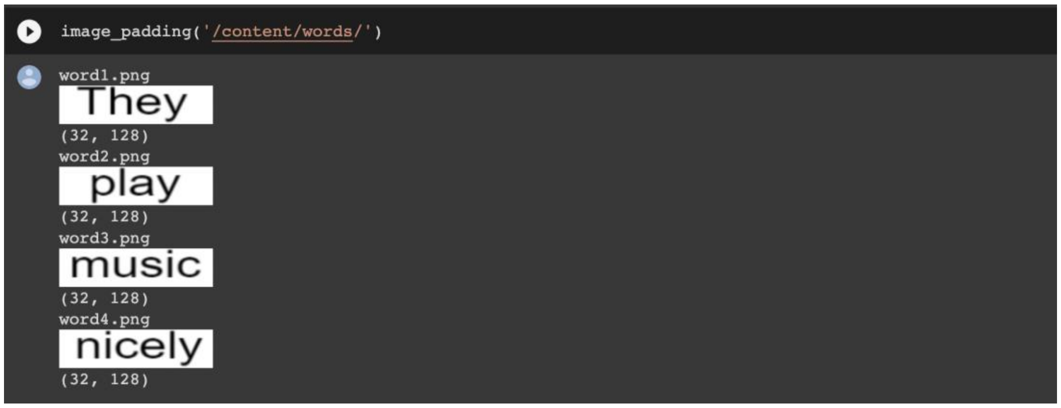


```
df = get_character_coordinates(cropped_path)
df
```

They play music nicely

	x	y	w	h
24	1	1	24	30
23	28	1	19	30
17	50	8	21	23
18	54	12	13	6
16	73	8	20	30
14	107	8	19	30
15	110	12	13	15

As our model checks for padding of 128x32, we then convert out cropped images of individual words to the images having padding as 128x32 and then adjust the dimensions.



```
image_padding('/content/words/')
word1.png
```

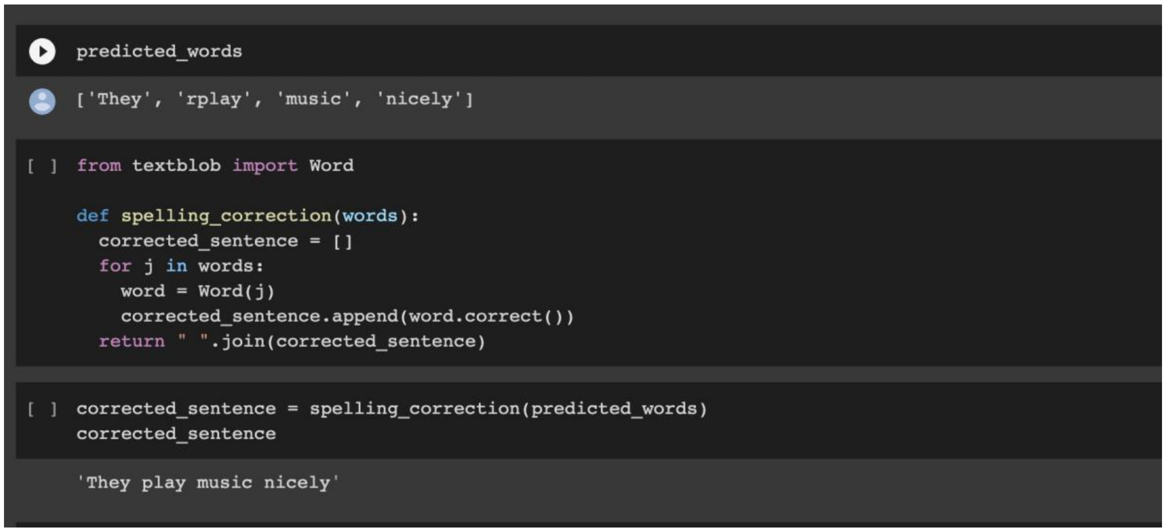
```
(32, 128)
word2.png
```

```
(32, 128)
word3.png
```

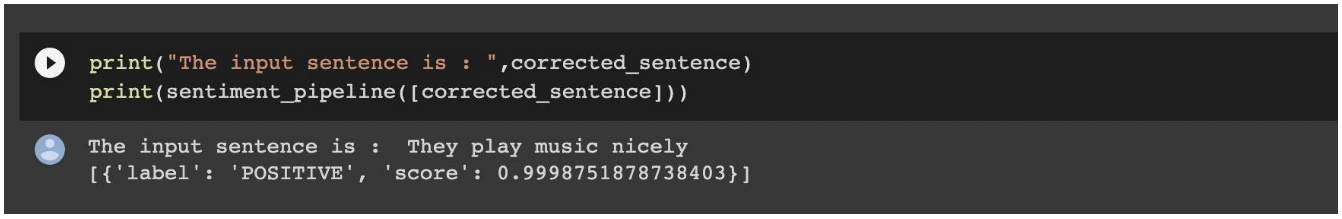
```
(32, 128)
word4.png
```

```
(32, 128)
```

We then pass these individual images to our model it then predicts the words. Here, the words in the images are predicted. But there are few spelling mistakes for example play is predicted as play here. Now, using using word correction methods we are changing the spelling to a properly readable english word or a word that has an appropriate english meaning. Then, we are passing these sentences to pre-trained sentiment analysis models which doesn't require train data. We are classifying the sentence provided in the image is either classifies as positive or negative. Here, the sentence we took They play music nicely is classified as positive with confidence score 0.99.

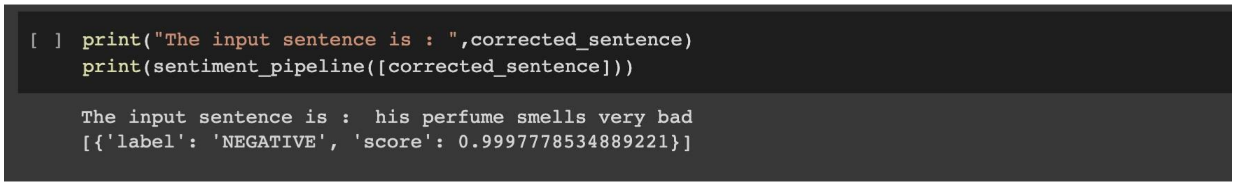


```
predicted_words
['They', 'rplay', 'music', 'nicely']

[ ] from textblob import Word

    def spelling_correction(words):
      corrected_sentence = []
      for j in words:
        word = Word(j)
        corrected_sentence.append(word.correct())
      return " ".join(corrected_sentence)

[ ] corrected_sentence = spelling_correction(predicted_words)
    corrected_sentence

    'They play music nicely'
```



```
print("The input sentence is : ",corrected_sentence)
print(sentiment_pipeline([corrected_sentence]))

The input sentence is :  They play music nicely
[{'label': 'POSITIVE', 'score': 0.9998751878738403}]
```

Similarly if we consider another image with text His perfume smells very bad and repeating the process it has been classified as negative with confidence score 0.99



```
[ ]  print("The input sentence is : ",corrected_sentence)
     print(sentiment_pipeline([corrected_sentence]))

     The input sentence is :  his perfume smells very bad
     [{'label': 'NEGATIVE', 'score': 0.9997778534889221}]
```

8. Preliminary Results:

After training the model, we have done prediction some sample images. The below screenshots shows someof the predicted outputs.

1.Out of bunch of images we have in our dataset we took a set of only 50-60 images for prediction, and we have showed some of them below.

2. The choice of images is based upon us, if we want to obtain different type of words/ texts, we can change the range as 20-30 for example and take i=20 so that we observe the same text as the image otherwise the image and the text gets mismatched.

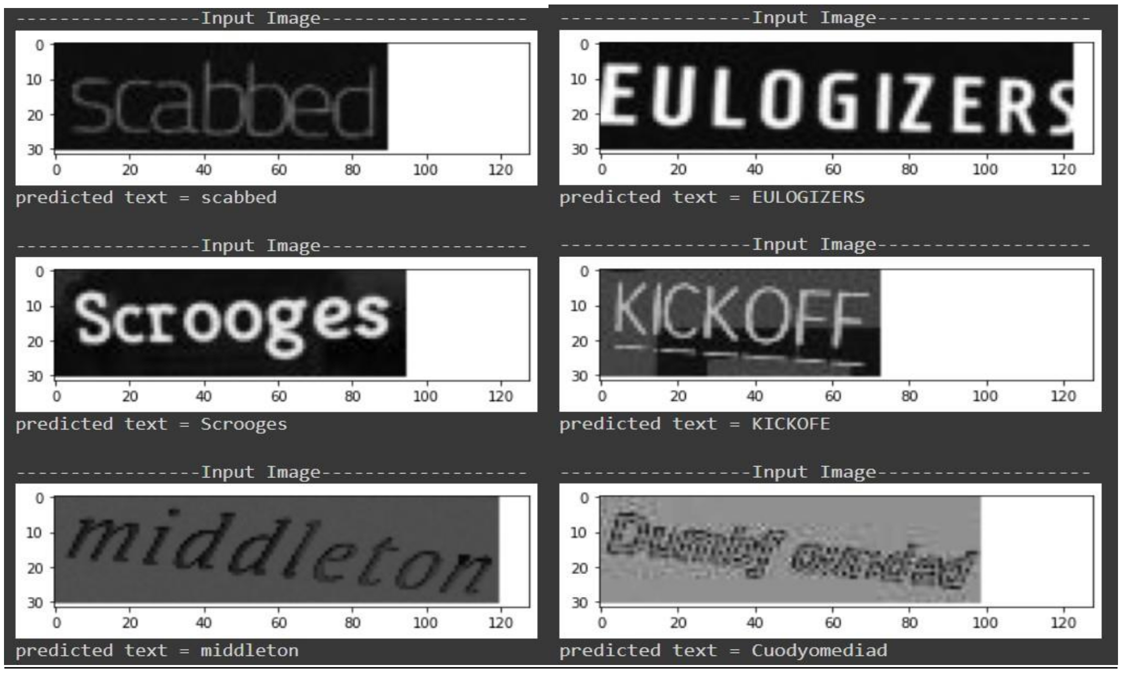
3. To Predict the text, first the input has been passed to the model for training.

4. There were multiple Convolution Layers Conv2D and at each layer we have so many internal operations that took place.

5. At the final output Layer, each character has been predicted from the character set={abcde…ABCD…0123...} we declared in code and the one which has the highest probability (greatest number) has been predicted/ chosen.

6. Below are some of the examples of the images like **scabbed, Scrooges, middleton** which have been correctly predicted as the **same text** as in the image.

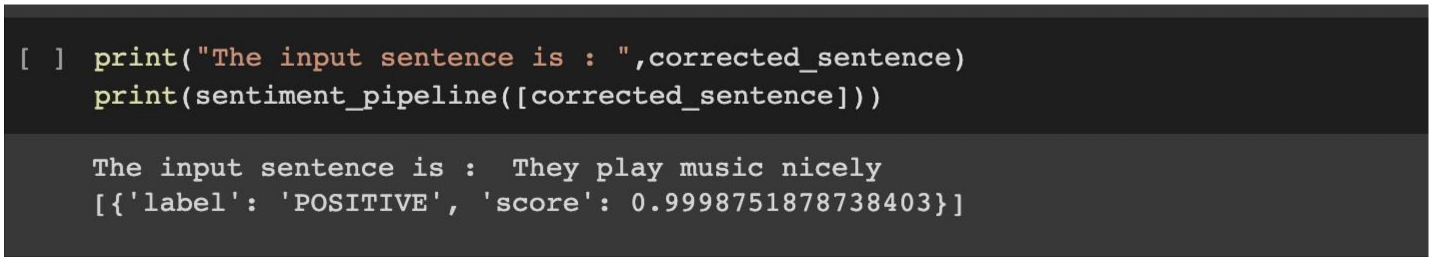
7.Using Model Training most of the predictions were correct but there were also some wrong predictions like KICKOFF has been predicted as KICKOFE.



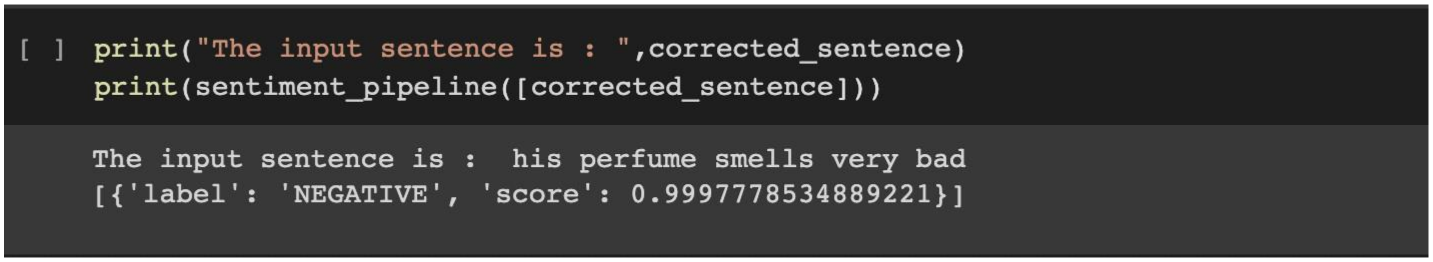
```
-----------------Input Image-------------------
predicted text = scabbed

-----------------Input Image-------------------
predicted text = EULOGIZERS

-----------------Input Image-------------------
predicted text = Scrooges

-----------------Input Image-------------------
predicted text = KICKOFE

-----------------Input Image-------------------
predicted text = middleton

-----------------Input Image-------------------
predicted text = Cuodyomediad
```

Sentiment analysis results:

1. For our Sentiment Analysis, a cropped image has been taken from the dataset which contains a text/ sentence.
2. The Words in the sentence from the cropped image have then been divided into individuals depending upon their pixel distance. Therefore, the initial cropped image has become sub images with individual words in it.
3. All the output images should be of the same size so for that word padding concept has been used.
4. These cropped images have been converted to images with words having padding of 128x32 and dimensions have also been adjusted.
5. Then, the words in the images been predicted when passed to our model and if any spelling mistakes, using the word correction methods those were re-corrected.
6. Transformers have played a major role here. By importing pipeline (sentiment analysis"), our text/sentences in the images have been classified as either positive or negative.
7. Below is the first example of sentiment analysis, our sentence is "**They play music nicely**". After implementing the sentiment Analysis, we classified it to be a **positive** sentence. It also provided us with the **confidence score** of the text as **0.99.**
8. The second example of sentiment analysis is "**his perfume smells very bad**". For this example, it was classified



```
[ ] print("The input sentence is : ",corrected_sentence)
    print(sentiment_pipeline([corrected_sentence]))

    The input sentence is :  They play music nicely
    [{'label': 'POSITIVE', 'score': 0.9998751878738403}]
```



```
[ ] print("The input sentence is : ",corrected_sentence)
    print(sentiment_pipeline([corrected_sentence]))

    The input sentence is :  his perfume smells very bad
    [{'label': 'NEGATIVE', 'score': 0.9997778534889221}]
```

9.Project Management

Everyone has contributed equally to our project.

Work completed: We have taken a dataset having images and then extracted the text present in the images. For that we have created a model and completed the tasks using convolution and Bi-directional LSTM. We have also done several preprocessing steps as mentioned above. For the sentiment analysis part, we have taken images with sentences in it and passed to the model then we have extracted the words and used piprline() class to classify the data as either positive or negative.

Responsibility and Contributions:

• All the datasets required for the project are analyzed and created by Neha Kalluri.

• Converting images to a multidimensional array is done by Lahari Kunduru. This is done to train the ma- chine learning model based on the features of the image.

• Data Preprocessing is done by Keerthana Pinikeshi which includes an array representation of the image and padding of the label.

• Convolution Recurrent Neural Network implementation for model building is taken care of by Neha.

• Sentiment Analysis coding part is done by the whole group.

• Changes in Data Preprocessing is done by Keerthana Pinikeshi.

• The documentation part is done by Keerthana Pinikeshi and Lahari Kunduru.

• Powerpoint presentation is done by Neha Kalluri.

Issues/Concerns: Since we are not allowed to change the dataset after increment 1, we have fol lowed this approach to do sentiment analysis.

10. References:

The below are the some of the initial references that we have researched as part of this project.

https://en.wikipedia.org/wiki/Optical_character_recognition

https://www.flatworldsolutions.com/data-management/articles/key-advantages-ocr-based-data-entry.php

https://www.cloudfactory.com/machine-learning/optical-character-recognition-ocr

https://www.linkedin.com/pulse/15-best-ocr-handwriting-datasets-machine-learning-limarc-ambalina/

https://towardsdatascience.com/the-most-favorable-pre-trained-sentiment-classifiers-in-python-9107c06442c6