A REPORT
ON

‘MEME SENTIMENT ANALYSIS’

BY

Name of the Students	ID Number
ANKIT AGARWAL	***2016B5A70468G***
YASH SHROFF	***2016B4A80495G***
KESANI MEHER MANOJ	***2013B5A80458G***

In partial fulfilment of the course

BITS 312

NEURAL NETWORKS AND FUZZY LOGIC

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

November, 2019

ACKNOWLEDGEMENT

We would like to thank Prof.Tirtharaj Dash and the TAs of the course BITS F312 for this excellent opportunity to do this project and guided us through the difficulties.We would also like to thank the developers of Python API which enabled us to do the project.We would also like to thank google colab for giving us the GPU required to run the code.

Without the help and contribution of the above-mentioned people, this project would not have been a success.

ABSTRACT

.Information on social media comprises of various modalities such as textual, visual and audio. NLP and Computer Vision communities often leverage only one prominent modality in isolation to study social media. However, computational processing of Internet memes needs a hybrid approach. The growing ubiquity of Internet memes on social media platforms such as Facebook, Instagram, and Twitter further suggests that we can not ignore such multimodal content anymore. This task has ~8K annotated memes - with human annotated tags namely sentiment, and type of humor that is, sarcastic, humorous, or offensive.

TABLE OF CONTENTS

1 INTRODUCTION

Memes are one of the most typed English words in recent times. Memes are often derived from our prior social and cultural experiences such as TV series or a popular cartoon character (think: One Does Not Simply - a now immensely popular meme taken from the movie Lord of the Rings). These digital constructs are so deeply ingrained in our Internet culture that to understand the opinion of a community, we need to understand the type of memes it shares.

The prevalence of hate speech in online social media is a nightmare and a great societal responsibility for many social media companies. When malicious users upload something offensive to torment or disturb people, it traditionally has to be seen and flagged by at least one human, either a user or a paid worker. Even today, companies like Facebook and Twitter rely extensively on outside human contractors from start-ups like CrowdFlower, or companies in the Philippines. But with the growing volume of multimodal social media it is becoming impossible to scale. The detection of offensive content on online social media is an ongoing struggle. Detecting an offensive meme is more complex than detecting an offensive text – it involves visual cue and language understanding. This is one of the motivating aspects which encourages us to propose this task.

1.1 AIM

The aim of this project is "***Motivation Classification ,Sentiment Classification and quantify the extent to which a meme is offensive.***"

1.2 PURPOSE

The purpose of this project is to develop a model which classifies the motivation, the sentiment and the offensiveness of a meme in a consistent and error-free manner.

1.3 SCOPE

This project helps to classify texts as offensive or non offensive.

2 Preprocessing And Loading Data

2.1 Introduction

We need to normalise the data at the beginning before feeding it to the algorithm.We extract the raw data and convert it to clean dataset.

2.1.1 Loading the glove library

In this project we use the glove or wordtovec library by the following code:

```
!wget http://nlp.stanford.edu/data/glove.6B.zip

!unzip glove.6B.zip
```

2.1.2 Loading dataset to be used in the project

The dataset has been stored in the google drive we import the dataset from google drive

```
from google.colab import drive

drive.mount('/content/drive')
```

2.1.3 Importing other libraries useful for the project

```
import numpy as np

import warnings

warnings.filterwarnings('ignore')

import pandas as pd

from keras.layers import  Flatten, Dense, Dropout
```

Training the data

Loading the given training set from Google Drive

```
train=pd.read_csv("/content/drive/My Drive/NNFL Fall'19 Project/data_7000_new.csv",header=None)

train.head(1)
```

Loading the images from the folder

Here if an image is not available in the dataset we remove it.

```
images={}

i=0

not_aval=[]

for file in train[0]:

  a=cv2.imread(os.path.join("/content/drive/My Drive/NNFL Fall'19 Project/data_7000/", file))

  print(i)

  if a is not None:

    images[i] = cv2.resize(a,(240,240))

    i+=1

  else:

    print(file,i)

    i+=1

    not_aval.append(i-1)
```

INITIALISING THE VALUES FOR TRAINING THE DATA

```
print(train[0][not_aval])

trainFinal=train.drop(not_aval,axis=0)

trainFinal.shape

x_text=trainFinal[3]

x_text.shape

y3=trainFinal[6]

y4=trainFinal[7]

y5=trainFinal[8]
```

COUNTING FREQUENCY OF OCCURRENCE OF EACH OUTPUT

```
a3={}

a4={}

a5={}


for i in y3:

  if i in a3.keys():

    a3[i]+=1

  else:

    a3[i]=1

for i in y4:

  if i in a4.keys():

    a4[i]+=1

  else:

    a4[i]=1
```

```
for i in y5:
  if i in a5.keys():
    a5[i]+=1
  else:
    a5[i]=1
```

Replacing outputs with numerical values

```
y4f= y4.replace({"motivational":+1,
"not_motivational":0,'negative':None,'neutral':None,'positive':None,'very_
negative':None,'very_positive':None})

a4f={}

for i in y4f:
  if i in a4f.keys():
    a4f[i]+=1
  else:
    a4f[i]=1

y5f= y5.replace({'negative': 1,'neutral': 2,'positive':
3,'positivechandler_Friday-Mood-AF.-meme-Friends-ChandlerBing.jpg':3,'very
_negative': 0,'very_positive': 4})

a5f={}

for i in y5f:
  if i in a5f.keys():
    a5f[i]+=1
  else:
    a5f[i]=1

print(a5f)


y3f= y3.replace({'hateful_offensive': 3,'very_offensive': 2,'slight':
1,'not_offensive': 0,'very_positive':None,'motivational':
```

```
None,'not_motivational':None,'positive':None})

a3f={}

for i in y3f:

  if i in a3f.keys():

    a3f[i]+=1

  else:

    a3f[i]=1

print(a3f)
```

Tokenizing the dictionary of words

```
word_index=[]

num_words = 12880

tokenizer = Tokenizer(num_words=num_words,
filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',lower=True,split=' ')

tokenizer.fit_on_texts(x_text.values)

X = tokenizer.texts_to_sequences(x_text.values)

word_index = tokenizer.word_index

print('Found %s unique tokens.' % len(word_index))

max_length_of_text = 50

X = pad_sequences(X, maxlen=max_length_of_text)

print(word_index)
```

Loading the weights from GloVe

```
import numpy as np

embeddings_index = {}

f = open( 'glove.6B.200d.txt')

for line in f:
```

```
    values = line.split()
    word = values[0]
    coefs = np.asarray(values[1:], dtype='float32')
    embeddings_index[word] = coefs
f.close()
print('Found %s word vectors.' % len(embeddings_index))
embedding_matrix = np.zeros((len(word_index) + 1, 200))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
print(embedding_matrix.shape)
```

REMOVING THE ROWS WITH MOTIVATION COLUMN NULL

```
moti_Text=X[y4f.notnull()]
moti_images=pd.Series(images)[y4f.notnull()]
y4f=y4f[y4f.notnull()]
y4final=y4f
```

TEST-VALIDATION SPLIT

```
moti_train_text,moti_test_text,moti_train_image,moti_test_image,
y_moti_train, y_moti_test =
train_test_split(moti_Text,moti_images,y4final,test_size = 0.1)
```

MAKING THE MODEL

```
from keras.layers import BatchNormalization as BatchNorm
```

```
embed_dim = 300 #Change to observe effects

lstm_out = 512 #Change to observe effects

batch_size = 32

input1 = keras.layers.Input(shape=(240,240,3,))

input2 = keras.layers.Input(shape=(50,))

iml=Conv2D(64, (3, 3), activation='relu')(input1)

iml=MaxPool2D(pool_size=(2, 2))(iml)

iml=Conv2D(64, (3, 3), activation='relu')(iml)

iml=MaxPool2D(pool_size=(2, 2))(iml)

iml=Conv2D(128, (3,3), activation='relu')(iml)

iml=MaxPool2D(pool_size=(2, 2))(iml)

iml = Flatten()(iml)

import keras

from keras.layers import Convolution1D as Conv1D

from  keras.layers import MaxPooling1D as MaxPooling1D

x = Embedding(12881, embed_dim,weights=[embedding_matrix])(input2)

x = Conv1D(512, 5, activation='relu', kernel_initializer='normal')(x)

x = Conv1D(256, 5, activation='relu', kernel_initializer='normal')(x)

x = MaxPooling1D(5)(x)

x = Conv1D(256, 5, activation='relu', kernel_initializer='normal')(x)

x = Flatten()(x)

added=keras.layers.Concatenate(axis=-1)([x,iml])

x=Dropout(0.1)(x)

x= Dense(128,activation='relu', kernel_initializer='normal')(added)

x=Dropout(0.1)(x)

preds = Dense(1, activation='sigmoid', kernel_initializer='normal')(x)

model = Model(inputs=[input1, input2], outputs=preds)
```

```
model.summary()
```

F1-SCORE CUSTOM METRIC

Copied from:-https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model

```
from keras import backend as K

def recall_m(y_true, y_pred):

        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))

        possible_positives = K.sum(K.round(K.clip(y_true, 0, 1)))

        recall = true_positives / (possible_positives + K.epsilon())

        return recall

def precision_m(y_true, y_pred):

        true_positives = K.sum(K.round(K.clip(y_true * y_pred, 0, 1)))

        predicted_positives = K.sum(K.round(K.clip(y_pred, 0, 1)))

        precision = true_positives / (predicted_positives + K.epsilon())

        return precision


def f1_m(y_true, y_pred):

    precision = precision_m(y_true, y_pred)

    recall = recall_m(y_true, y_pred)

    return 2*((precision*recall)/(precision+recall+K.epsilon()))
```

COMPILING THE MODEL AND DIVIDING IMAGES BY 255

```
model.compile(loss = 'binary_crossentropy', optimizer='sgd', metrics = ['mae','accuracy',f1_m])

moti_train_image=np.array(moti_train_image.tolist())

moti_test_image=np.array(moti_test_image.tolist())
```

```
moti_train_image=np.true_divide(moti_train_image,255)

moti_test_image=np.true_divide(moti_test_image,255)
```

TRAINING THE MODEL

```
from keras.callbacks import ModelCheckpoint

filepath="weights-improvement.hdf5"

checkpoint = ModelCheckpoint(filepath, monitor='val_f1_m', verbose=1,
save_best_only=True, mode='max')callbacks_list = [checkpoint]

model.fit(x=[moti_train_image,moti_train_text], y=y_moti_train,
batch_size=32, epochs=30,callbacks=callbacks_list,
verbose=1,validation_data=([moti_test_image,moti_test_text],y_moti_test))

model.load_weights("weights-improvement.hdf5")

moti_train_text.shape

y_predicted=model.predict([moti_test_image,moti_test_text])

y_predicted=y_predicted.reshape(-1)

ans=[]

for i in y_predicted:

  if i>=.5:

    ans.append(1)

  else:

    ans.append(0)

print(ans)

np.average(ans)

print(ans)
```

We have used the same model for all three tasks. The Only difference is that in the last dense layer for the

hate and positive tasks the activation is "Linear".

Plotting the Model Diagram

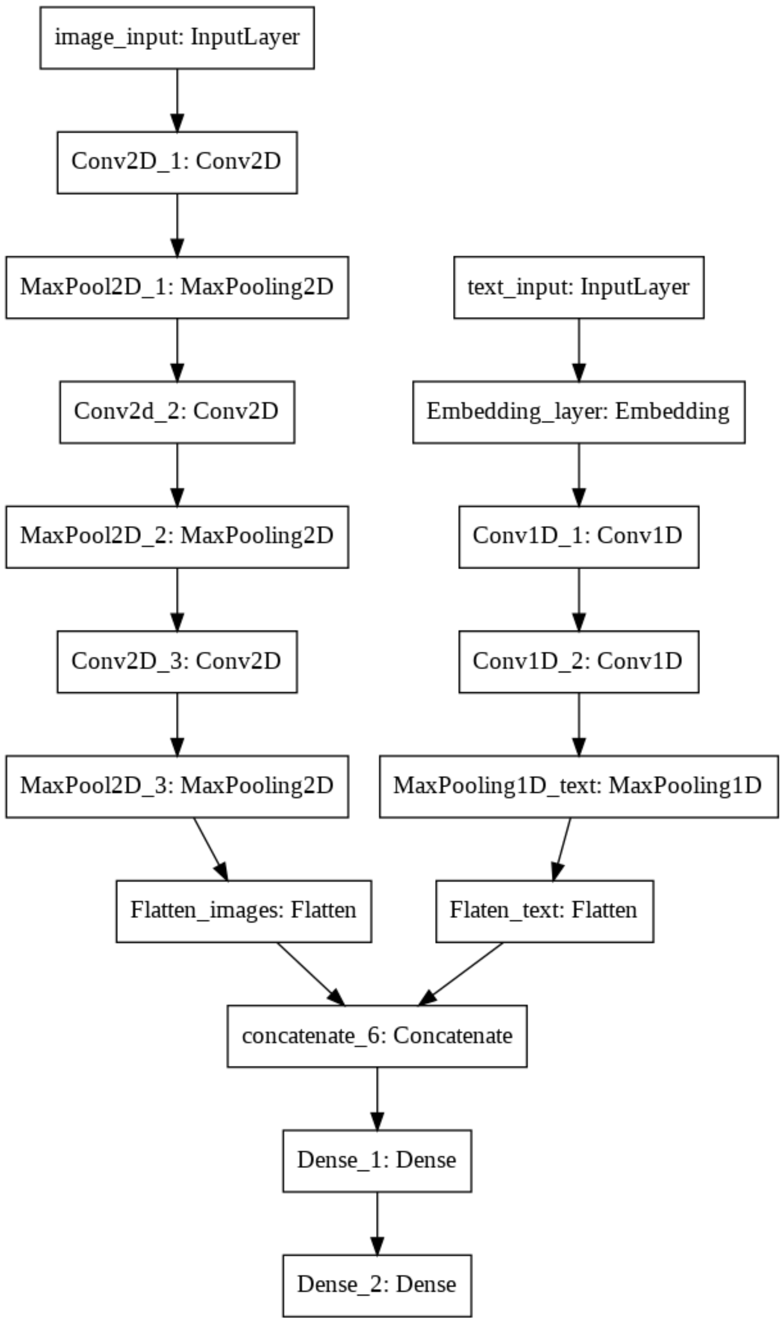
```
from keras.utils import plot_model

plot_model(model, to_file='model.png')
```

F1-Score for the Hate Task

```
from keras.callbacks import Callback

from sklearn.metrics import confusion_matrix, f1_score, precision_score,
recall_score

class Metrics(Callback):


  def __init__(self, valid_data, *args, **kwargs):

    self.valid_data = valid_data


    super(Metrics, self).__init__(*args, **kwargs)


  def on_train_begin(self, logs={}):

    pass


  def on_epoch_end(self, epoch, logs={}):

    global temp

    val_predict =self.model.predict(self.valid_data[0])

    val_targ =self.valid_data[1]


    _val_f1 = f1_score(val_targ, np.round(val_predict), average = 'macro')
```

```
    print(" — val_macro_f1: {}".format(_val_f1))
    return


metric = Metrics(([off_test_image,off_test_text],y3_off_test))
```

Model Diagram



SCOPE FOR IMPROVEMENT:-

We can use bert(with pre-trained weights) for text classification and bigger network for image classification.

CONCLUSION

F1 score for motivation task0.49340

F1 score for offensive task=0.2504

MAE for POSITIVE/NEGATIVE TASK=0.69991

REFERENCES

Keras-Documentation-https://keras.io/

numpy-https://numpy.org/

F1 score-
https://datascience.stackexchange.com/questions/45165/how-to-get-accuracy-f1-precision-and-recall-for-a-keras-model