Real_Comments_and_Advertisement_Comment_Prediction

October 3, 2021

[1]:
```
import pandas as pd
```

[2]:
```
#Reading dataset
data = pd.read_csv('/train.csv')
```

[3]:
```
#We will check the shape of the dataset and the top five elements of the␣
 ↪dataset.
data.shape
```

[3]:
```
(1157, 5)
```

[4]:
```
#Head of the dataset
data.head()
```

[4]:
```
                                    COMMENT_ID  ... CLASS
0  LZQPQhLyRh80UYxNuaDWhIGQYNQ96IuCg-AYWqNPjpU  ...     1
1          z13jhp0bxqncu512g22wvzkasxmvvzjaz04  ...     1
2          z13fwbwp1oujthgqj04chlngpvzmtt3r3dw  ...     1
3          z13lfzdo5vmdi1cm123te5uz2mqig1brz04  ...     1
4          z12avveb4xqiirsix04chxviiljryduwxg0  ...     1

[5 rows x 5 columns]
```

[5]:
```
data.describe()
```

[5]:
```
             CLASS
count  1157.000000
mean      0.506482
std       0.500174
min       0.000000
25%       0.000000
50%       1.000000
75%       1.000000
max       1.000000
```

[6]:
```
data.isna().sum()
```

[6]:
```
COMMENT_ID      0
AUTHOR          0
DATE          138
CONTENT         0
```

```
CLASS           0
dtype: int64
```

[7]:
```
#Fill NaN values with mode values
data["DATE"] = data["DATE"].fillna(data["DATE"].mode())

data.isna().sum()
```

[7]:
```
COMMENT_ID    0
AUTHOR        0
DATE          0
CONTENT       0
CLASS         0
dtype: int64
```

[8]:
```
df=data[['CONTENT','CLASS']]
df.head()
```

[8]:
```
                                             CONTENT  CLASS
0  Huh, anyway check out this you[tube] channel: ...      1
1   me shaking my sexy ass on my channel enjoy ^_^        1
2            watch?v=vtaRGgvGtWQ   Check this out .      1
3                          Subscribe to my channel       1
4  and u should.d check my channel and tell me wh...      1
```

[9]:
```
## Get the Independent Features

X=df.drop('CLASS',axis=1)
```

[10]:
```
## Get the Dependent features
y=df['CLASS']
```

[11]:
```
y.value_counts()
```

[11]:
```
1    586
0    571
Name: CLASS, dtype: int64
```

[12]:
```
import tensorflow as tf
tf.__version__
```

[12]:
```
'2.6.0'
```

[13]:
```
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.text import one_hot
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense, Conv2D
from tensorflow.keras.layers import Bidirectional
from tensorflow.keras.layers import Dropout
```

[14]:
```
voc_size=5000
```

[15]:
```
message = X.copy()
```

[16]:
```
message['CONTENT'][1]
```

[16]:
```
'me shaking my sexy ass on my channel enjoy ^_^ \ufeff'
```

[17]:
```
message.reset_index(inplace=True)
```

[18]:
```
import nltk
import re
from nltk.corpus import stopwords
```

[19]:
```
nltk.download('stopwords')
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(message)):
    review = re.sub('[^a-zA-Z]', ' ', message['CONTENT'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.
 ↪words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

[20]:
```
corpus[1]
```

[20]:
```
'shake sexi ass channel enjoy'
```

[21]:
```
onehot_repr=[one_hot(words,voc_size)for words in corpus]
onehot_repr[1]
```

[21]:
```
[4493, 4213, 6, 4486, 4924]
```

[22]:
```
sent_length=40
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 2834 4486 2592]
 [   0    0    0 ...    6 4486 4924]
 [   0    0    0 ... 4138 4206  158]
 ...
 [   0    0    0 ... 2151 1654  802]
 [   0    0    0 ...    0    0 2468]
 [   0    0    0 ... 3795 2504 2463]]
```

[23]:
```
embedded_docs[0]
```

[23]:
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0, 3022, 2429,  158, 2834, 4486, 2592], dtype=int32)
```

[24]:
```
## Creating model
embedding_vector_features=50
model1=Sequential()
model1.
 ↪add(Embedding(voc_size,embedding_vector_features,input_length=sent_length))
model1.add(Dropout(0.25))
model1.add(Bidirectional(LSTM(100))) ##Just add bidirectional!!, except it␣
 ↪would just behave as normal LSTM Model
model1.add(Dropout(0.25))
model1.add(Dense(32, activation='relu'))
model1.add(Dense(64,activation='relu'))
model1.add(Dropout(0.25))
model1.add(Dense(1,activation='sigmoid'))
model1.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])
print(model1.summary())
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 40, 50)            250000
_________________________________________________________________
dropout (Dropout)            (None, 40, 50)            0
_________________________________________________________________
bidirectional (Bidirectional (None, 200)               120800
_________________________________________________________________
dropout_1 (Dropout)          (None, 200)               0
_________________________________________________________________
dense (Dense)                (None, 32)                6432
_________________________________________________________________
dense_1 (Dense)              (None, 64)                2112
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 65
=================================================================
Total params: 379,409
Trainable params: 379,409
Non-trainable params: 0
_________________________________________________________________
None
```

[25]:
```
len(embedded_docs),y.shape
```

[25]:
```
(1157, (1157,))
```

[26]:
```
import numpy as np
X_final=np.array(embedded_docs)
y_final=np.array(y)
```

[27]:
```
X_final[1]
```

[27]:
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0, 4493, 4213,    6, 4486, 4924], dtype=int32)
```

[28]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_final, y_final,␣
 ↪test_size=0.25, random_state=32)
```

[29]:
```
model1.
 ↪fit(X_train,y_train,validation_data=(X_test,y_test),epochs=25,batch_size=64)
```

```
Epoch 1/25
14/14 [==============================] - 6s 164ms/step - loss: 0.6883 -
accuracy: 0.5802 - val_loss: 0.6732 - val_accuracy: 0.6207
Epoch 2/25
14/14 [==============================] - 1s 99ms/step - loss: 0.6396 - accuracy:
0.6805 - val_loss: 0.5901 - val_accuracy: 0.6897
Epoch 3/25
14/14 [==============================] - 1s 100ms/step - loss: 0.4982 -
accuracy: 0.7762 - val_loss: 0.4310 - val_accuracy: 0.8310
Epoch 4/25
14/14 [==============================] - 1s 101ms/step - loss: 0.3086 -
accuracy: 0.8870 - val_loss: 0.2819 - val_accuracy: 0.8793
Epoch 5/25
14/14 [==============================] - 1s 96ms/step - loss: 0.1604 - accuracy:
0.9516 - val_loss: 0.2022 - val_accuracy: 0.9207
Epoch 6/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0877 - accuracy:
0.9689 - val_loss: 0.1665 - val_accuracy: 0.9276
Epoch 7/25
14/14 [==============================] - 1s 99ms/step - loss: 0.0432 - accuracy:
0.9885 - val_loss: 0.1710 - val_accuracy: 0.9345
Epoch 8/25
14/14 [==============================] - 1s 95ms/step - loss: 0.0327 - accuracy:
0.9908 - val_loss: 0.1805 - val_accuracy: 0.9379
Epoch 9/25
14/14 [==============================] - 1s 98ms/step - loss: 0.0189 - accuracy:
0.9942 - val_loss: 0.1941 - val_accuracy: 0.9345
Epoch 10/25
```

```
14/14 [==============================] - 1s 96ms/step - loss: 0.0187 - accuracy:
0.9942 - val_loss: 0.2045 - val_accuracy: 0.9379
Epoch 11/25
14/14 [==============================] - 1s 99ms/step - loss: 0.0176 - accuracy:
0.9965 - val_loss: 0.1871 - val_accuracy: 0.9414
Epoch 12/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0164 - accuracy:
0.9942 - val_loss: 0.1956 - val_accuracy: 0.9379
Epoch 13/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0139 - accuracy:
0.9977 - val_loss: 0.1993 - val_accuracy: 0.9345
Epoch 14/25
14/14 [==============================] - 1s 98ms/step - loss: 0.0118 - accuracy:
0.9977 - val_loss: 0.2272 - val_accuracy: 0.9414
Epoch 15/25
14/14 [==============================] - 1s 96ms/step - loss: 0.0131 - accuracy:
0.9977 - val_loss: 0.2308 - val_accuracy: 0.9448
Epoch 16/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0144 - accuracy:
0.9977 - val_loss: 0.2278 - val_accuracy: 0.9379
Epoch 17/25
14/14 [==============================] - 1s 98ms/step - loss: 0.0113 - accuracy:
0.9977 - val_loss: 0.2362 - val_accuracy: 0.9345
Epoch 18/25
14/14 [==============================] - 1s 98ms/step - loss: 0.0094 - accuracy:
0.9988 - val_loss: 0.2459 - val_accuracy: 0.9345
Epoch 19/25
14/14 [==============================] - 1s 98ms/step - loss: 0.0104 - accuracy:
0.9977 - val_loss: 0.2491 - val_accuracy: 0.9379
Epoch 20/25
14/14 [==============================] - 1s 98ms/step - loss: 0.0120 - accuracy:
0.9977 - val_loss: 0.2554 - val_accuracy: 0.9345
Epoch 21/25
14/14 [==============================] - 1s 96ms/step - loss: 0.0080 - accuracy:
0.9988 - val_loss: 0.2686 - val_accuracy: 0.9310
Epoch 22/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0124 - accuracy:
0.9965 - val_loss: 0.2727 - val_accuracy: 0.9345
Epoch 23/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0128 - accuracy:
0.9942 - val_loss: 0.2792 - val_accuracy: 0.9345
Epoch 24/25
14/14 [==============================] - 1s 97ms/step - loss: 0.0140 - accuracy:
0.9977 - val_loss: 0.2911 - val_accuracy: 0.9241
Epoch 25/25
14/14 [==============================] - 1s 95ms/step - loss: 0.0110 - accuracy:
0.9954 - val_loss: 0.3305 - val_accuracy: 0.9138
```

[29]:
```
<keras.callbacks.History at 0x7fe12fe06c10>
```

[30]:
```
y_pred=model1.predict(X_test)
```

[31]:
```
y_test[0:5]
```

[31]:
```
array([0, 0, 1, 1, 1])
```

[32]:
```
y_pred[0:5]
```

[32]:
```
array([[3.70466709e-03],
       [1.06304564e-04],
       [9.90338981e-01],
       [9.99973416e-01],
       [9.99997139e-01]], dtype=float32)
```

[33]:
```
type(y_test), type(y_pred)
```

[33]:
```
(numpy.ndarray, numpy.ndarray)
```

[34]:
```
y_pred.round()[0:5]
```

[34]:
```
array([[0.],
       [0.],
       [1.],
       [1.],
       [1.]], dtype=float32)
```

[35]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred.round())
```

[35]:
```
array([[134,  16],
       [  9, 131]])
```

[36]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred.round(decimals=0))
```

[36]:
```
0.9137931034482759
```

[37]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred.round(decimals=0)))
```

```
              precision    recall  f1-score   support

           0       0.94      0.89      0.91       150
           1       0.89      0.94      0.91       140

    accuracy                           0.91       290
   macro avg       0.91      0.91      0.91       290
weighted avg       0.91      0.91      0.91       290
```

[38]:
```
#Reading dataset
df_test = pd.read_csv('/test.csv')
```

[39]:
```
#We will check the shape of the dataset and the top five elements of the␣
 ↪dataset.
df_test.shape
```

[39]:
```
(799, 5)
```

[40]:
```
#Head of the dataset
df_test.head()
```

[40]:
```
   ID  ...                                            CONTENT
0   0  ...  Hey guys check out my new channel and our firs...
1   1  ...              just for test I have to say murdev.com
2   2  ...  Hey, check out my new website!! This site is a...
3   3  ...  i turned it on mute as soon is i came on i jus...
4   4  ...    You should check my channel for Funny VIDEOS!!

[5 rows x 5 columns]
```

[41]:
```
df_test1=df_test[['ID','CONTENT']]
```

[42]:
```
df_test1.head()
```

[42]:
```
   ID                                            CONTENT
0   0  Hey guys check out my new channel and our firs...
1   1              just for test I have to say murdev.com
2   2  Hey, check out my new website!! This site is a...
3   3  i turned it on mute as soon is i came on i jus...
4   4    You should check my channel for Funny VIDEOS!!
```

[43]:
```
df_test1.isna().sum()
```

[43]:
```
ID         0
CONTENT    0
dtype: int64
```

[44]:
```
from nltk.stem.porter import PorterStemmer
ps = PorterStemmer()
corpus = []
for i in range(0, len(df_test1)):
    review = re.sub('[^a-zA-Z]', ' ', df_test1['CONTENT'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.
 ↪words('english')]
    review = ' '.join(review)
    corpus.append(review)
```

[45]:
```
onehot_repr=[one_hot(words,voc_size)for words in corpus]
onehot_repr[1]
```

[45]:
```
[4245, 3390, 4017, 1853]
```

[46]:
```
sent_length=40
embedded_docs=pad_sequences(onehot_repr,padding='pre',maxlen=sent_length)
print(embedded_docs)
```

```
[[   0    0    0 ... 1284  963 1823]
 [   0    0    0 ... 3390 4017 1853]
 [   0    0    0 ... 4714 1290 1853]
 ...
 [   0    0    0 ... 2395 3284 3795]
 [   0    0    0 ... 4752 2599 3971]
 [   0    0    0 ... 3795 3999  130]]
```

[47]:
```
embedded_docs.shape
```

[47]:
```
(799, 40)
```

[48]:
```
embedded_docs[0]
```

[48]:
```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
       2723, 4348,  158, 1217, 4486, 2415, 2649, 3971, 4627, 4627, 4638,
         52,  963, 1882, 2940, 1284,  963, 1823], dtype=int32)
```

[49]:
```
len(sorted(model1.predict(embedded_docs[0]), reverse=True))
```

```
WARNING:tensorflow:Model was constructed with shape (None, 40) for input
KerasTensor(type_spec=TensorSpec(shape=(None, 40), dtype=tf.float32,
name='embedding_input'), name='embedding_input', description="created by layer
'embedding_input'"), but it was called on an input with incompatible shape
(None, 1).
```

[49]:
```
40
```

[50]:
```
corpus = []
sLength = len(df_test1['ID'])
df_test1['CLASS'] = 0 #pd.Series(np.random.randn(sLength), index=df_test1.
 ↪index)
for i in range(0, len(df_test1)-1):
    review = re.sub('[^a-zA-Z]', ' ', df_test1['CONTENT'][i])
    review = review.lower()
    review = review.split()

    review = [ps.stem(word) for word in review if not word in stopwords.
 ↪words('english')]
    review = ' '.join(review)
    #corpus.append(review)
    onehot_repr1=[one_hot(words,voc_size)for words in review]
    embedded_docs1=pad_sequences(onehot_repr1,padding='pre',maxlen=sent_length)
    if (len(embedded_docs1) > 0):
```

```
    if ((model1.predict(embedded_docs1)).all() > 0.95):
      df_test1['CLASS'][i] = 1
    else:
      df_test1['CLASS'][i] = 0
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  This is separate from the ipykernel package so we can avoid doing imports
until
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:16:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  app.launch_new_instance()
/usr/local/lib/python3.7/dist-packages/IPython/core/interactiveshell.py:2882:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  exec(code_obj, self.user_global_ns, self.user_ns)
```

[55]:
```
df_test1.head(20)
```

[55]:

	ID	CONTENT	CLASS
0	0	Hey guys check out my new channel and our firs...	1
1	1	just for test I have to say murdev.com	1
2	2	Hey, check out my new website!! This site is a...	1
3	3	i turned it on mute as soon is i came on i jus...	1
4	4	You should check my channel for Funny VIDEOS!!	1
5	5	Hey subscribe to me	1
6	6	subscribe like comment	1
7	7	http://www.ebay.com/itm/171183229277?ssPageNam...	1
8	8	http://ubuntuone.com/40beUutVu2ZKxK4uTgPZ8K	1
9	9	We are an EDM apparel company dedicated to bri...	1
10	10	i think about 100 millions of the views come f...	1
11	11	subscribe to my channel people :D	1
12	12	just checking the views	1
13	13	marketglory . com/strategygame/andrijamatf ear...	1
14	14	Check me out! I'm kyle. I rap so yeah	1
15	15	Came here to check the views, goodbye.	1

```
16  16  Why dafuq is a Korean song so big in the USA. ...      1
17  17  Check my channel please! And listen to the bes...      1
18  18  SUB 4 SUB PLEASE LIKE THIS COMMENT I WANT A SU...      1
19  19   Hey everyone!! I have just started my first Y...      1
```

[52]:
```
df_test1['CLASS'].value_counts()
```

[52]:
```
1    789
0     10
Name: CLASS, dtype: int64
```

[53]:
```
df2=df_test1[['ID','CLASS']]
df2.head()
```

[53]:
```
   ID  CLASS
0   0      1
1   1      1
2   2      1
3   3      1
4   4      1
```

[54]:
```
# saving the dataframe

df2.to_csv('/file1.csv', index=False)
```

[54]: