Proj74ThesisReport_SuryaMunjal3

October 9, 2022

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
#from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.metrics import log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
#from sklearn.cross_validation import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
#from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
!pip install -q kaggle
import re
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

from wordcloud import WordCloud, STOPWORDS
from os import path

import pandas as pd
import matplotlib.pyplot as plt
import re
import time
import warnings
import sqlite3
from sqlalchemy import create_engine # database connection
import csv
import os
warnings.filterwarnings("ignore")
import datetime as dt
import numpy as np
from nltk.corpus import stopwords
from sklearn.decomposition import TruncatedSVD
from sklearn.preprocessing import normalize
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.manifold import TSNE
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
#from sklearn.metrics.classification import accuracy_score, log_loss
from sklearn.metrics import log_loss
from sklearn.feature_extraction.text import TfidfVectorizer
from collections import Counter
from scipy.sparse import hstack
from sklearn.multiclass import OneVsRestClassifier
```

```
from sklearn.svm import SVC
#from sklearn.cross_validation import StratifiedKFold
from collections import Counter, defaultdict
from sklearn.calibration import CalibratedClassifierCV
from sklearn.naive_bayes import MultinomialNB
from sklearn.naive_bayes import GaussianNB
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
import math
from sklearn.metrics import normalized_mutual_info_score
from sklearn.ensemble import RandomForestClassifier



from sklearn.model_selection import cross_val_score
from sklearn.linear_model import SGDClassifier
#from mlxtend.classifier import StackingClassifier

from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import precision_recall_curve, auc, roc_curve
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
```

[2]:
```
from google.colab import files
files.upload()
```

```
<IPython.core.display.HTML object>


Saving kaggle.json to kaggle.json
```

[2]:
```
{'kaggle.json':
b'{"username":"suryamunjal","key":"376b73a1d8c76f470856f9ab7d5318e5"}'}
```

[3]:
```
!mkdir ~/.kaggle
```

[4]:
```
!cp kaggle.json ~/.kaggle/
```

[5]:
```
!chmod 600 ~/.kaggle/kaggle.json
```

[6]:
```
!kaggle datasets download -d akshayaki/fakenews
```

```
Downloading fakenews.zip to /content
 48% 17.0M/35.6M [00:01<00:01, 9.97MB/s]
100% 35.6M/35.6M [00:01<00:00, 21.7MB/s]
```

[7]:
```
!unzip fakenews.zip
```

```
Archive:  fakenews.zip
  inflating: News.csv
```

[8]:
```
df=pd.read_csv('News.csv',nrows=2000)
```

[9]:
```
df['combined']=df['title']+ " " + df["text"]
```

[10]:
```
df['Labels']=df['Labels'].map({'Fake': 0, 'True': 1})
```

[11]:
```
df=df[['combined','Labels']]
```

[12]:
```
df.head(2)
```

[12]:

	combined	Labels
0	GRAPHIC RIOT VIDEOS EXPOSE THUGS ATTACKING ELD…	0
1	BIG BROTHER: FEDS WANT YOUR DOCTOR TO WARN YOU…	0

[13]:
```
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')

def decontracted(phrase):
    # specific
    phrase = re.sub(r"won't", "will not", phrase)
    phrase = re.sub(r"can\'t", "can not", phrase)

    # general
    phrase = re.sub(r"n\'t", " not", phrase)
    phrase = re.sub(r"\'re", " are", phrase)
    phrase = re.sub(r"\'s", " is", phrase)
    phrase = re.sub(r"\'d", " would", phrase)
    phrase = re.sub(r"\'ll", " will", phrase)
    phrase = re.sub(r"\'t", " not", phrase)
    phrase = re.sub(r"\'ve", " have", phrase)
    phrase = re.sub(r"\'m", " am", phrase)
    return phrase

def preproc(sentance):
```

```
#from tqdm import tqdm
# tqdm is for printing the status bar
#for sentance in tqdm(df['combined'].values):
    sentance = re.sub(r"http\S+", "url", sentance)  #removing urls with space
    sentance = BeautifulSoup(sentance, 'lxml').get_text() # removes tags like␣
↪<br>
    sentance = decontracted(sentance)
    sentance = re.sub("\S*\d\S*", "", sentance).strip()  # remove words with␣
↪numbers
    sentance = re.sub('[^A-Za-z]+', ' ', sentance)  ##remove spacial character:
    sentance = ' '.join(e.lower() for e in sentance.split() if e.lower() not in␣
↪stopwords)

    return(sentance.strip())
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[14]:
```
df['Final_text']=df['combined'].apply(lambda x: preproc(str(x)))


#dropping the original combined column
df.drop(['combined'],inplace=True,axis=1)
```

[15]:
```
df.head(2)
```

[15]:

	Labels	Final_text
0	0	graphic riot videos expose thugs attacking eld…
1	0	big brother feds want doctor warn global warmi…

TRAIN TEST SPLIT

[16]:
```
# split data into test and train

X_train, X_test, y_train, y_test = train_test_split(df['Final_text'], df.
↪Labels, test_size = 0.3)
```

[17]:
```
X_train.shape
```

[17]:
```
(1400,)
```

[18]:
```
X_test.shape
```

[18]:
```
(600,)
```

[19]:
```
! pip install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting transformers
  Downloading transformers-4.22.2-py3-none-any.whl (4.9 MB)
     |                  | 4.9 MB 12.7 MB/s
Collecting huggingface-hub<1.0,>=0.9.0
  Downloading huggingface_hub-0.10.0-py3-none-any.whl (163 kB)
     |                  | 163 kB 28.3 MB/s
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Collecting tokenizers!=0.11.3,<0.13,>=0.11.1
  Downloading
tokenizers-0.12.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (6.6
MB)
     |                  | 6.6 MB 47.9 MB/s
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.21.6)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-
packages (from transformers) (4.64.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.7/dist-
packages (from transformers) (6.0)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.8.0)
Requirement already satisfied: importlib-metadata in
/usr/local/lib/python3.7/dist-packages (from transformers) (4.12.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.23.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.7/dist-packages (from huggingface-
hub<1.0,>=0.9.0->transformers) (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers)
(3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->transformers) (3.8.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->transformers) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (2022.6.15)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.10.0 tokenizers-0.12.1
transformers-4.22.2
```

[20]:
```
from transformers import AutoTokenizer,TFBertModel
tokenizer = AutoTokenizer.from_pretrained('bert-large-uncased')
bert = TFBertModel.from_pretrained('bert-large-uncased')
```

```
Downloading:   0%|          | 0.00/28.0 [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/571 [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/232k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/466k [00:00<?, ?B/s]

Downloading:   0%|          | 0.00/1.47G [00:00<?, ?B/s]

Some layers from the model checkpoint at bert-large-uncased were not used when
initializing TFBertModel: ['nsp___cls', 'mlm___cls']
- This IS expected if you are initializing TFBertModel from the checkpoint of a
model trained on another task or with another architecture (e.g. initializing a
BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertModel from the checkpoint
of a model that you expect to be exactly identical (initializing a
BertForSequenceClassification model from a BertForSequenceClassification model).
All the layers of TFBertModel were initialized from the model checkpoint at
bert-large-uncased.
If your task is similar to the task the model of the checkpoint was trained on,
you can already use TFBertModel for predictions without further training.
```

[21]:
```
tokenizer('hi surya are u having nice day')
```

[21]:
```
{'input_ids': [101, 7632, 7505, 3148, 2024, 1057, 2383, 3835, 2154, 102],
'token_type_ids': [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], 'attention_mask': [1, 1, 1, 1,
1, 1, 1, 1, 1, 1]}
```

[22]:
```
print("max len of tweets",max([len(x.split()) for x in df['Final_text']]))
max_length = 512
```

```
max len of tweets 2170
```

[23]:
```
X_train = tokenizer(
    text=X_train.tolist(),
    add_special_tokens=True,
    max_length=512,
    truncation=True,
    padding=True,
    return_tensors='tf',
```

```
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

[24]:
```
X_train['input_ids'].shape
```

[24]:
```
TensorShape([1400, 512])
```

[25]:
```
X_train['attention_mask'].shape
```

[25]:
```
TensorShape([1400, 512])
```

[26]:
```
X_test = tokenizer(
    text=X_test.tolist(),
    add_special_tokens=True,
    max_length=512,
    truncation=True,
    padding=True,
    return_tensors='tf',
    return_token_type_ids = False,
    return_attention_mask = True,
    verbose = True)
```

BUild the model

[27]:
```
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.initializers import TruncatedNormal
from tensorflow.keras.losses import CategoricalCrossentropy,BinaryCrossentropy
from tensorflow.keras.metrics import CategoricalAccuracy,BinaryAccuracy
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
```

[28]:
```
max_len = 512
import tensorflow as tf
from tensorflow.keras.layers import Input, Dense

input_ids = Input(shape=(max_len,), dtype=tf.int32, name="input_ids")
input_mask = Input(shape=(max_len,), dtype=tf.int32, name="attention_mask")
# embeddings = dbert_model(input_ids,attention_mask = input_mask)[0]

last_hidden_state = bert(input_ids,attention_mask = input_mask)[0]

cls_token=last_hidden_state[:, 0, :]
                                                    #(0 is the last␣
 ↪hidden states,1 means output)
# out = tf.keras.layers.GlobalMaxPool1D()(embeddings)
```

```
out = tf.keras.layers.Dropout(0.1)(cls_token)

out = Dense(32, activation='relu')(out)
out = tf.keras.layers.Dropout(0.1)(out)
y = Dense(1,activation = 'sigmoid')(out)

model = tf.keras.Model(inputs=[input_ids, input_mask], outputs=y)
model.layers[2].trainable = False
```

[29]:
```
model.summary()
```

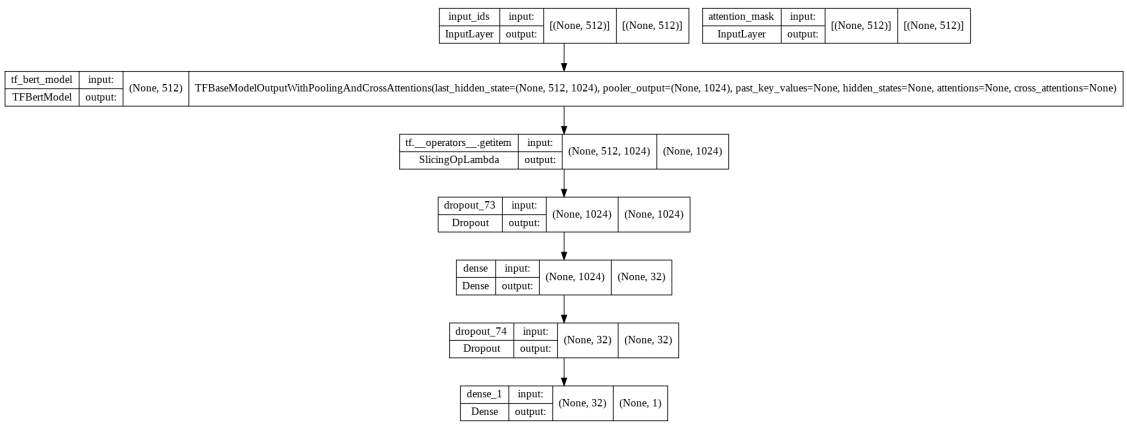
```
Model: "model"
__________________________________________________________________________________
__________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================
==================
 input_ids (InputLayer)         [(None, 512)]        0           []

 attention_mask (InputLayer)    [(None, 512)]        0           []

 tf_bert_model (TFBertModel)    TFBaseModelOutputWi  335141888
['input_ids[0][0]',
                                thPoolingAndCrossAt
'attention_mask[0][0]']
                                tentions(last_hidde
                                n_state=(None, 512,
                                 1024),
                                 pooler_output=(Non
                                e, 1024),
                                 past_key_values=No
                                ne, hidden_states=N
                                one, attentions=Non
                                e, cross_attentions
                                =None)

 tf.__operators__.getitem (Slic  (None, 1024)        0
['tf_bert_model[0][0]']
 ingOpLambda)

 dropout_73 (Dropout)           (None, 1024)         0
['tf.__operators__.getitem[0][0]'
                                                                  ]

 dense (Dense)                  (None, 32)           32800
['dropout_73[0][0]']
```

```
 dropout_74 (Dropout)           (None, 32)           0           ['dense[0][0]']

 dense_1 (Dense)                (None, 1)            33
['dropout_74[0][0]']

==================================================================================
==================
Total params: 335,174,721
Trainable params: 32,833
Non-trainable params: 335,141,888

----------------------------------------------------------------------------------
------------------
```

[30]:
```
'''
#optimizer = Adam(
    learning_rate=6e-06, # this learning rate is for bert model , taken from␣
 ↪huggingface website
    epsilon=1e-08,
    decay=0.01,
    clipnorm=1.0)
    '''

# Set loss and metrics
loss = BinaryCrossentropy(from_logits = True)
metric = BinaryAccuracy('accuracy'),
# Compile the model
model.compile(
    optimizer = 'adam',
    loss = loss,
    metrics = metric)
```

[31]:
```
plot_model(model, show_shapes = True)
```

[31]:



[32]:
```
y_train = np.asarray(y_train).astype('float32').reshape((-1,1))
y_test = np.asarray(y_test).astype('float32').reshape((-1,1))
```

[33]:
```
y_test.shape
```

[33]:
```
(600, 1)
```

[34]:
```
import tensorflow as tf
tf.config.experimental.list_physical_devices('GPU')
```

[34]:
```
[PhysicalDevice(name='/physical_device:GPU:0', device_type='GPU')]
```

[35]:
```
history = model.fit(
    x ={'input_ids':X_train['input_ids'],'attention_mask':
 ↪X_train['attention_mask']} ,
    y = y_train,
    #validation_split = 0.2,
    epochs=5,
    batch_size=128
)
```

```
Epoch 1/5
11/11 [==============================] - 230s 18s/step - loss: 0.4209 -
accuracy: 0.8436
Epoch 2/5
11/11 [==============================] - 197s 18s/step - loss: 0.2879 -
accuracy: 0.9150
Epoch 3/5
11/11 [==============================] - 198s 18s/step - loss: 0.2596 -
accuracy: 0.9150
Epoch 4/5
11/11 [==============================] - 197s 18s/step - loss: 0.2361 -
accuracy: 0.9150
Epoch 5/5
11/11 [==============================] - 197s 18s/step - loss: 0.2201 -
accuracy: 0.9150
```

[44]:
```
history = model.fit(
    x ={'input_ids':X_train['input_ids'],'attention_mask':
 ↪X_train['attention_mask']} ,
    y = y_train,
    #validation_split = 0.2,
    epochs=5,
    batch_size=128
)
```

```
Epoch 1/5
```

```
11/11 [==============================] - 197s 18s/step - loss: 0.1978 -
accuracy: 0.9150
Epoch 2/5
11/11 [==============================] - 197s 18s/step - loss: 0.1819 -
accuracy: 0.9157
Epoch 3/5
11/11 [==============================] - 198s 18s/step - loss: 0.1707 -
accuracy: 0.9171
Epoch 4/5
11/11 [==============================] - 198s 18s/step - loss: 0.1567 -
accuracy: 0.9200
Epoch 5/5
11/11 [==============================] - 198s 18s/step - loss: 0.1475 -
accuracy: 0.9279
```

[36]:
```
predicted = model.predict({'input_ids':X_test['input_ids'],'attention_mask':
 ↪X_test['attention_mask']})
```

[37]:
```
predicted[0:10]
```

[37]:
```
array([[0.04273619],
       [0.1782984 ],
       [0.10104788],
       [0.04077688],
       [0.01482965],
       [0.04904471],
       [0.03211592],
       [0.13681045],
       [0.05249479],
       [0.05802174]], dtype=float32)
```

[38]:
```
predicted[0]
```

[38]:
```
array([0.04273619], dtype=float32)
```

[39]:
```
y_pred_thresh = np.where(predicted >= 0.5, 1, 0)
```

[40]:
```
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_test, y_pred_thresh)
print(accuracy)
```

```
0.93
```

[41]:
```
y_pred_thresh[0:20]
```

[41]:
```
array([[0],
       [0],
       [0],
```

```
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0],
       [0]])
```

[42]:
```
y_test[0:20]
```

[42]:
```
array([[0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.],
       [0.]], dtype=float32)
```