CS640 PA4 Report

Details about the pre-processing (cleaning) step you performed

I preprocess the data to remove any tweets longer than the max tweet length, because that is bound to be garbage data. Since we're using bert-base-uncased, I convert all the tweets to lowercase, remove URL's and @mentions and replace them with spaces. I then remove nltk stopwords from the data to remove filler words and any words that don't necessarily add to the meaning of the tweet. This is followed by the removal of punctuation.

A short description of the layers you added and your reasoning in addition to the training hyperparameters. (Optimizer, Learning rate, batch size, Loss Function,...)

For the Naive Bayes portion of the assignment I chose to use the Complement Naive Bayes classifier since it's suited for imbalanced datasets. I tokenized the data using the RegexpTokenizer and vectorized the data using CountVectorizer.

For BERT, I chose to use the bert-base-uncased model from TFHub, along with its associated tokenizer. I chose to use a batch size of 32 because that's what the authors of BERT recommended be used for retraining the model with different data. I used the Adam optimizer with a learning rate of 2e-5, again at the author's recommendations. I chose the Binary Crossentropy Loss function because the task is essentially binary classification.

I appended a Batch Normalization layer to the pooled output from the BERT Classifier to normalize inputs for subsequent layers. This was followed by two sets of Dropout+FC Layers, and the final fully connected layer with a sigmoid activation function. The Dropout+FC layers improve the generalization performance of the model by randomly zeroing the inputs from a certain percentage of neurons making the features independent and not heavily correlated.

The performance of your models (BERT and Bayes Classifier) on the Test-CSV provided with the dataset. For this, you can include a classification report. You are encouraged to use library functions to do this

The Bayes Classifier had a test accuracy of 82%. The precision and recall numbers were almost identical representing a well balanced dataset, and no overfitting.

The BERT Classifier had a test accuracy of 84%. Again, the precision and recall numbers were close to each other, with the precision of a zero prediction slightly higher than that of that of a one, and the recall for one was slightly higher than that of zero. However, looking at the loss and accuracy plots, overfitting is apparent - loss keeps decreasing while validation loss keeps increasing, and validation accuracy stays the same. Changing the batch size, learning rate and dropout didn't do much to change this, which leads me to believe that the amount of data I'm training with - 0.01% of the training set (due to compute time limitations) was too small for the network to converge on, and with 100M parameters, the network tried to fit the data perfectly.

Naive_Bayes_Classification

December 1, 2021

```
[ ]: import re
     import pandas as pd
     import tensorflow as tf
     from sklearn.metrics import *
     import matplotlib.pyplot as plt
     from nltk.corpus import stopwords
     import matplotlib.patches as mpatches
     from nltk.tokenize import RegexpTokenizer
     from sklearn.naive_bayes import ComplementNB
     from sklearn.feature_extraction.text import CountVectorizer
```

1 Naive Bayes

1.1 Import Data

```
[ ]: df = pd.read_csv("data/training.1600000.processed.noemoticon.csv",
                     names=['polarity', 'id', 'date', 'query', 'user', 'text'],
                     encoding='latin-1',
                     usecols=['polarity', 'text'])

     test_df = pd.read_csv("data/testdata.manual.2009.06.14.csv",
                     names=['polarity', 'id', 'date', 'query', 'user', 'text'],
                     encoding='latin-1',
                     usecols=['polarity', 'text'])

     test_df = test_df[test_df['polarity'] != 2]
```

1.2 Preprocess Data

```
[ ]: # Remove tweets longer than 280 characters
     df = df.drop(df[df.text.str.len() > 280].index).reset_index(drop=True)
     test_df = test_df.drop(test_df[test_df.text.str.len() > 280].index).
      ↪reset_index(drop=True)

     # Convert to lowercase
     df['text'] = df['text'].str.lower()
     test_df['text'] = test_df['text'].str.lower()
```

```
# Remove URLs and @mentions
df['text'] = df['text'].apply(lambda t: re.sub('@[^ ]+|http[^ ]+', '', t).
 ↪strip())
test_df['text'] = test_df['text'].apply(lambda t: re.sub('@[^ ]+|http[^ ]+',␣
 ↪'', t).strip())

# Remove stopwords
stop = stopwords.words('english')
df['text'] = df['text'].apply(lambda t: ' '.join([w for w in t.split() if w not␣
 ↪in stop]))
test_df['text'] = test_df['text'].apply(lambda t: ' '.join([w for w in t.
 ↪split() if w not in stop]))

# Remove punctuation
df['text'] = df['text'].apply(lambda t: re.sub('[^a-zA-Z0-9\s]', '', t).strip())
test_df['text'] = test_df['text'].apply(lambda t: re.sub('[^a-zA-Z0-9\s]', '',␣
 ↪t).strip())

# Replace label 4 with 1
df['polarity'] = df.polarity.replace(4,1)
test_df['polarity'] = test_df.polarity.replace(4,1)
```

1.3 Tokenize and Create Train/Test Sets

[ ]:
```
token = RegexpTokenizer(r'[a-zA-Z0-9]+')
cv = CountVectorizer(stop_words='english',ngram_range = (1,2),tokenizer = token.
 ↪tokenize)
```

[ ]:
```
X_train=cv.fit_transform(df['text'].values.astype('U'))
y_train=df['polarity']
X_test=cv.transform(test_df['text'].values.astype('U'))
y_test=test_df['polarity']
```

1.4 Training and Results

[ ]:
```
cnb = ComplementNB()
cnb.fit(X_train, y_train)
print ("Train accuracy ={:.2f}%".format(cnb.score(X_train,y_train)*100))
print ("Test accuracy ={:.2f}%".format(cnb.score(X_test,y_test)*100))
train_acc_cnb=cnb.score(X_train,y_train)
test_acc_cnb=cnb.score(X_test,y_test)
```

```
Train accuracy =91.09%
Test accuracy =82.17%
```

[ ]:
```
y_pred_cnb =cnb.predict(X_test)
print(classification_report(y_test, y_pred_cnb))
```

```
              precision    recall  f1-score   support

           0       0.82      0.82      0.82       177
           1       0.83      0.82      0.82       182

    accuracy                           0.82       359
   macro avg       0.82      0.82      0.82       359
weighted avg       0.82      0.82      0.82       359
```

[ ]:

BERT_Classification

December 1, 2021

[ ]:
```
!pip install tensorflow-text
!wget http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip
!mkdir data
!unzip trainingandtestdata.zip
```

```
Collecting tensorflow-text
  Downloading tensorflow_text-2.7.3-cp37-cp37m-manylinux2010_x86_64.whl (4.9 MB)
     |                 | 4.9 MB 12.6 MB/s
Requirement already satisfied: tensorflow<2.8,>=2.7.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow-text) (2.7.0)
Requirement already satisfied: tensorflow-hub>=0.8.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow-text) (0.12.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (1.1.0)
Requirement already satisfied: keras-preprocessing>=1.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (1.1.2)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (3.1.0)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (1.13.3)
Requirement already satisfied: flatbuffers<3.0,>=1.12 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (2.0)
Requirement already satisfied: absl-py>=0.4.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (0.12.0)
Requirement already satisfied: libclang>=9.0.1 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (12.0.0)
Requirement already satisfied: wheel<1.0,>=0.32.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (0.37.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (1.42.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (0.2.0)
```

```
Requirement already satisfied: protobuf>=3.9.2 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (3.17.3)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.21.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (0.22.0)
Requirement already satisfied: tensorflow-estimator<2.8,~=2.7.0rc0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (2.7.0)
Requirement already satisfied: tensorboard~=2.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (2.7.0)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (3.3.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (1.6.3)
Requirement already satisfied: keras<2.8,>=2.7.0rc0 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (2.7.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (3.10.0.2)
Requirement already satisfied: numpy>=1.14.5 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (1.19.5)
Requirement already satisfied: gast<0.5.0,>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from tensorflow<2.8,>=2.7.0->tensorflow-
text) (0.4.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.7/dist-
packages (from tensorflow<2.8,>=2.7.0->tensorflow-text) (1.15.0)
Requirement already satisfied: cached-property in /usr/local/lib/python3.7/dist-
packages (from h5py>=2.9.0->tensorflow<2.8,>=2.7.0->tensorflow-text) (1.5.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (1.35.0)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (2.23.0)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (0.4.6)
Requirement already satisfied: setuptools>=41.0.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (57.4.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in
```

```
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (1.8.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.7/dist-
packages (from tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(3.3.6)
Requirement already satisfied: werkzeug>=0.11.15 in
/usr/local/lib/python3.7/dist-packages (from
tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text) (1.0.1)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(0.2.8)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.7/dist-
packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(4.7.2)
Requirement already satisfied: cachetools<5.0,>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(4.2.4)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.7/dist-packages (from google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(1.3.0)
Requirement already satisfied: importlib-metadata>=4.4 in
/usr/local/lib/python3.7/dist-packages (from
markdown>=2.6.8->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(4.8.2)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata>=4.4->markdown>=2.6.8->tensorboard~=2.6->tenso
rflow<2.8,>=2.7.0->tensorflow-text) (3.6.0)
Requirement already satisfied: pyasn1<0.5.0,>=0.4.6 in
/usr/local/lib/python3.7/dist-packages (from pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(0.4.8)
Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(2.10)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.7/dist-packages (from
requests<3,>=2.21.0->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(2021.10.8)
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<3,>=2.21.0->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(1.24.3)
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.7/dist-
packages (from requests-oauthlib>=0.7.0->google-auth-
oauthlib<0.5,>=0.4.1->tensorboard~=2.6->tensorflow<2.8,>=2.7.0->tensorflow-text)
(3.1.1)
Installing collected packages: tensorflow-text
Successfully installed tensorflow-text-2.7.3
--2021-11-30 11:25:50--
http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip
Resolving cs.stanford.edu (cs.stanford.edu)… 171.64.64.64
Connecting to cs.stanford.edu (cs.stanford.edu)|171.64.64.64|:80… connected.
HTTP request sent, awaiting response… 301 Moved Permanently
Location: https://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip
[following]
--2021-11-30 11:25:50--
https://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip
Connecting to cs.stanford.edu (cs.stanford.edu)|171.64.64.64|:443… connected.
HTTP request sent, awaiting response… 200 OK
Length: 81363704 (78M) [application/zip]
Saving to: ‘trainingandtestdata.zip’

trainingandtestdata 100%[===================>]  77.59M  19.4MB/s    in 4.9s

2021-11-30 11:25:55 (15.8 MB/s) - ‘trainingandtestdata.zip’ saved
[81363704/81363704]

Archive:  trainingandtestdata.zip
  inflating: testdata.manual.2009.06.14.csv
  inflating: training.1600000.processed.noemoticon.csv
```

[ ]:
```
import nltk
nltk.download('stopwords')
```

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

[ ]:
```
True
```

[ ]:
```
import re
import numpy as np
import pandas as pd
import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
from nltk.corpus import stopwords
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

[ ]:
```
df = pd.read_csv("data/training.1600000.processed.noemoticon.csv",
                names=['polarity', 'id', 'date', 'query', 'user', 'text'],
                encoding='latin-1',
                usecols=['polarity', 'text'])

test_df = pd.read_csv("data/testdata.manual.2009.06.14.csv",
                names=['polarity', 'id', 'date', 'query', 'user', 'text'],
                encoding='latin-1',
                usecols=['polarity', 'text'])

test_df = test_df[test_df['polarity'] != 2]
```

[ ]:
```
# Remove tweets longer than 280 characters
df = df.drop(df[df.text.str.len() > 280].index).reset_index(drop=True)
test_df = test_df.drop(test_df[test_df.text.str.len() > 280].index).
 ↪reset_index(drop=True)

# Convert to lowercase
df['text'] = df['text'].str.lower()
test_df['text'] = test_df['text'].str.lower()

# Remove URLs and @mentions
df['text'] = df['text'].apply(lambda t: re.sub('@[^ ]+|http[^ ]+', '', t).
 ↪strip())
test_df['text'] = test_df['text'].apply(lambda t: re.sub('@[^ ]+|http[^ ]+',␣
 ↪'', t).strip())

# Remove stopwords
stop = stopwords.words('english')
df['text'] = df['text'].apply(lambda t: ' '.join([w for w in t.split() if w not␣
 ↪in stop]))
test_df['text'] = test_df['text'].apply(lambda t: ' '.join([w for w in t.
 ↪split() if w not in stop]))

# Remove punctuation
df['text'] = df['text'].apply(lambda t: re.sub('[^a-zA-Z0-9\s]', '', t).strip())
test_df['text'] = test_df['text'].apply(lambda t: re.sub('[^a-zA-Z0-9\s]', '',␣
 ↪t).strip())

# Replace label 4 with 1
df['polarity'] = df.polarity.replace(4,1)
test_df['polarity'] = test_df.polarity.replace(4,1)
```

[ ]:
```
train_df, remaining = train_test_split(df, train_size = 0.01, stratify =␣
↪df['polarity'])
val_df, _ = train_test_split(remaining,train_size = 0.001, stratify =␣
↪remaining['polarity'])
```

[ ]:
```
with tf.device('/cpu:0'):
  train_data = tf.data.Dataset.from_tensor_slices((train_df['text'].values,␣
↪train_df['polarity'].values))
  val_data = tf.data.Dataset.from_tensor_slices((val_df['text'].values,␣
↪val_df['polarity'].values))
  test_data = tf.data.Dataset.from_tensor_slices((test_df['text'].values,␣
↪test_df['polarity'].values))
```

[ ]:
```
bert_layer = 'https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4'
tokenizer = 'https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3'
```

[ ]:
```
with tf.device('/cpu:0'):
  train_data = train_data.batch(32, drop_remainder = True).prefetch(tf.data.
↪experimental.AUTOTUNE)
  val_data = val_data.batch(32, drop_remainder = True).prefetch(tf.data.
↪experimental.AUTOTUNE)
  test_data = test_data.batch(32, drop_remainder = False).prefetch(tf.data.
↪experimental.AUTOTUNE)
```

[ ]:
```
def create_model():

  text_input = tf.keras.layers.Input(shape=(), dtype=tf.string)

  preprocessor = hub.KerasLayer(tokenizer)
  encoder_inputs = preprocessor(text_input)

  encoder = hub.KerasLayer(bert_layer, trainable=True)
  outputs = encoder(encoder_inputs)

  pooled_output = outputs['pooled_output']

  bnorm = tf.keras.layers.BatchNormalization()(pooled_output)
  dropout1 = tf.keras.layers.Dropout(0.4)(bnorm)
  fullyconnected1 = tf.keras.layers.Dense(128, activation='relu')(dropout1)
  dropout2 = tf.keras.layers.Dropout(0.4)(fullyconnected1)
  fullyconnected2 = tf.keras.layers.Dense(32, activation='relu')(dropout2)
  final_output = tf.keras.layers.Dense(1, activation='sigmoid')(fullyconnected2)

  return tf.keras.Model(text_input, final_output)
```

[ ]:
```
model = create_model()
model.compile(optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5),
              loss = tf.keras.losses.BinaryCrossentropy(),
              metrics = [tf.keras.metrics.BinaryAccuracy()])
```

[ ]:
```
callback = tf.keras.callbacks.EarlyStopping(monitor='val_binary_accuracy',␣
 ↪min_delta=0.01, patience=2, restore_best_weights=True)
```

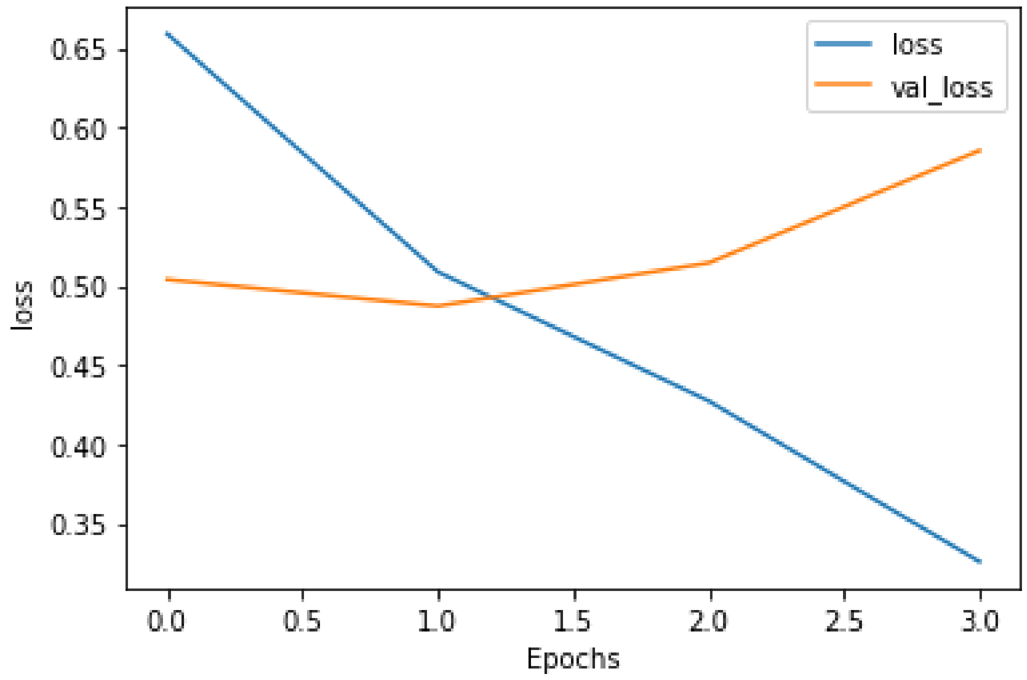
[ ]:
```
epochs = 10
history = model.fit(train_data,
                    validation_data=val_data,
                    epochs=epochs,
                    callbacks=[callback],
                    verbose=1)
```

```
Epoch 1/10
499/499 [==============================] - 801s 2s/step - loss: 0.6590 -
binary_accuracy: 0.6663 - val_loss: 0.5039 - val_binary_accuracy: 0.7621
Epoch 2/10
499/499 [==============================] - 782s 2s/step - loss: 0.5087 -
binary_accuracy: 0.7589 - val_loss: 0.4871 - val_binary_accuracy: 0.7736
Epoch 3/10
499/499 [==============================] - 782s 2s/step - loss: 0.4272 -
binary_accuracy: 0.8092 - val_loss: 0.5144 - val_binary_accuracy: 0.7755
Epoch 4/10
499/499 [==============================] - 782s 2s/step - loss: 0.3258 -
binary_accuracy: 0.8608 - val_loss: 0.5853 - val_binary_accuracy: 0.7761
```

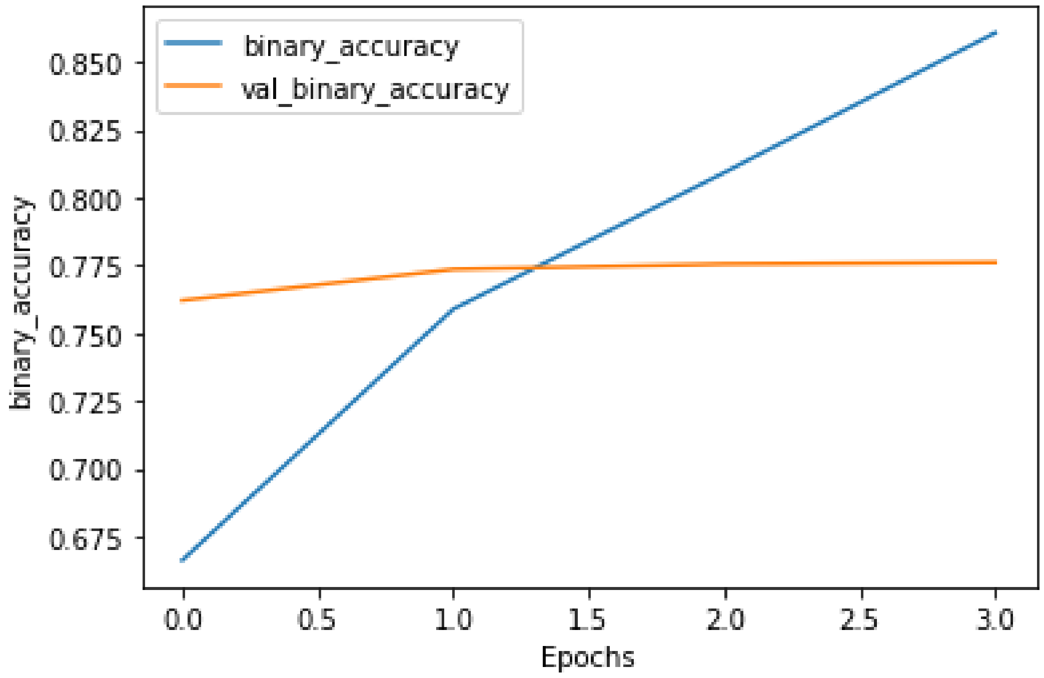
[ ]:
```
import matplotlib.pyplot as plt

def plot_graphs(history, metric):
  plt.plot(history.history[metric])
  plt.plot(history.history['val_'+metric], '')
  plt.xlabel("Epochs")
  plt.ylabel(metric)
  plt.legend([metric, 'val_'+metric])
  plt.show()
```

[ ]:
```
plot_graphs(history=history, metric='loss')
```



```
[ ]: plot_graphs(history,'binary_accuracy')
```



```
[ ]: y_true = test_df['polarity']
     y_pred = model.predict(test_data)

     y_pred = y_pred.flatten()

     for i in range(len(y_pred)):
       if y_pred[i] < 0.5:
         y_pred[i] = 0
       else:
         y_pred[i] = 1

     print(classification_report(y_true, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.81      0.84       177
           1       0.83      0.87      0.85       182

    accuracy                           0.84       359
   macro avg       0.85      0.84      0.84       359
weighted avg       0.84      0.84      0.84       359
```