Name: Vincent Nguyen

Class: CS 4395.001

Date: 24 April 2023

NetID: VTN180000

```
import pandas as panda
import tensorflow
from tensorflow import keras
from tensorflow.keras import layers, models, preprocessing
from tensorflow.keras.preprocessing.text import Tokenizer
import numpy
import pickle
import pandas
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report
from sklearn.preprocessing import OneHotEncoder


# Read in csv file
dataframes = panda.read_csv(\
        '/kaggle/input/amazon-reviews-dataset/cleaned_reviews.csv', \
                            header=0, usecols=[0,1], encoding='ISO-8859-1')


# Use 0 and 1 to replace the neutral, negative, and positive words in
# the sentiment column. Negative and neutral will be
# 0 to represent sentiments other than positive
dataframes.sentiments.replace('neutral', 0, inplace=True)
dataframes.sentiments.replace('positive', 1, inplace=True)
dataframes.sentiments.replace('negative', 0, inplace=True)

# Make sure the column is replaced with numeric values correctly
dataframes.head()
```

	sentiments	cleaned_review
0	1	i wish would have gotten one earlier love it a...
1	0	i ve learned this lesson again open the packag...
2	0	it is so slow and lags find better option
3	0	roller ball stopped working within months of m...
4	0	i like the color and size but it few days out ...

```
# Split into test and train
length = len(dataframes)
```

```
model_train = dataframes[split]
model_test = dataframes[~split]
print(model_train.shape)
print(model_test.shape)
```

```
(13831, 2)
(3509, 2)
```

```
label_x_y = 2
size_vocabulary = 30000
size_batches = 200

# Tokenize the test and train
model_tokenize = Tokenizer(num_words=size_vocabulary)
model_tokenize.fit_on_texts(model_train.cleaned_review.astype(str))


# Convert the trained and test texts to a matrix
modes = 'tfidf'
train_matrix_x = model_tokenize.texts_to_matrix(\
                     model_train.cleaned_review.astype(str), \
                      mode=modes)
test_matrix_x = model_tokenize.texts_to_matrix(\
                     model_test.cleaned_review.astype(str), \
                      mode=modes)

# Set label encoder and transform the sentiments
label_coder = LabelEncoder()
label_coder.fit(model_train.sentiments)
train_encode_y = label_coder.transform(model_train.sentiments)
test_encode_y = label_coder.transform(model_test.sentiments)


# Add layers, compile, and build the model
sequential = models.Sequential()
sequential.add(layers.Dense(13,input_dim=size_vocabulary, \
                            activation='relu', kernel_initializer='normal'))
sequential.add(layers.Dense(13,input_dim=size_vocabulary, \
                            activation='relu', kernel_initializer='normal'))
sequential.add(layers.Dense(1, activation='sigmoid', \
               kernel_initializer='normal'))
sequential.compile(optimizer='adam', metrics=['accuracy'], \
                   loss='binary_crossentropy')
fit_sequential = sequential.fit(train_matrix_x, train_encode_y,
                           batch_size=size_batches, epochs = 20,
                           validation_split=0.1, verbose=1)
```

```
Epoch 1/20
63/63 [==============================] - 3s 28ms/step - loss: 0.6534 - accuracy: 0.55
Epoch 2/20
```

```
63/63 [==============================] - 1s 18ms/step - loss: 0.4821 - accuracy: 0.79
Epoch 3/20
63/63 [==============================] - 1s 17ms/step - loss: 0.3117 - accuracy: 0.89
Epoch 4/20
63/63 [==============================] - 1s 18ms/step - loss: 0.1984 - accuracy: 0.94
Epoch 5/20
63/63 [==============================] - 1s 17ms/step - loss: 0.1337 - accuracy: 0.96
Epoch 6/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0970 - accuracy: 0.97
Epoch 7/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0719 - accuracy: 0.98
Epoch 8/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0556 - accuracy: 0.98
Epoch 9/20
63/63 [==============================] - 1s 19ms/step - loss: 0.0440 - accuracy: 0.99
Epoch 10/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0350 - accuracy: 0.99
Epoch 11/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0275 - accuracy: 0.99
Epoch 12/20
63/63 [==============================] - 1s 18ms/step - loss: 0.0226 - accuracy: 0.99
Epoch 13/20
63/63 [==============================] - 1s 18ms/step - loss: 0.0189 - accuracy: 0.99
Epoch 14/20
63/63 [==============================] - 1s 18ms/step - loss: 0.0155 - accuracy: 0.99
Epoch 15/20
63/63 [==============================] - 1s 18ms/step - loss: 0.0133 - accuracy: 0.99
Epoch 16/20
63/63 [==============================] - 1s 18ms/step - loss: 0.0114 - accuracy: 0.99
Epoch 17/20
63/63 [==============================] - 1s 18ms/step - loss: 0.0100 - accuracy: 0.99
Epoch 18/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0090 - accuracy: 0.99
Epoch 19/20
63/63 [==============================] - 1s 20ms/step - loss: 0.0077 - accuracy: 0.99
Epoch 20/20
63/63 [==============================] - 1s 17ms/step - loss: 0.0069 - accuracy: 0.99
```

```
# Get classification report for model
sequential_prediction = sequential.predict(test_matrix_x)
sequential_prediction = [1 if value >= 0.5 else 0 for value \
                         in sequential_prediction]
report = classification_report(test_encode_y, sequential_prediction)
print(report)
```

```
110/110 [==============================] - 0s 3ms/step
              precision    recall  f1-score   support

           0       0.90      0.88      0.89      1590
           1       0.91      0.92      0.91      1919

    accuracy                           0.90      3509
   macro avg       0.90      0.90      0.90      3509
```

```
        macro avg       0.90      0.90      0.90      3509
     weighted avg       0.90      0.90      0.90      3509
```

Analysis:

The sequential model does really well. Looking at the classification report, I see that the accuracy is 0.90, which is a very high score. Thus, I would say the sequential model is really good for deep learning.

LSTM

```
label_x_y = 2
size_vocabulary = 15000
size_batches = 100

# Tokenize the test and train
model_tokenize = Tokenizer(num_words=size_vocabulary)
model_tokenize.fit_on_texts(model_train.cleaned_review.astype(str))



# Convert the trained and test texts to a matrix
modes = 'tfidf'
train_matrix_x = model_tokenize.texts_to_matrix(\
                      model_train.cleaned_review.astype(str), \
                       mode=modes)
test_matrix_x = model_tokenize.texts_to_matrix(\
                      model_test.cleaned_review.astype(str), \
                       mode=modes)

# Set label encoder and transform the sentiments
label_coder = LabelEncoder()
label_coder.fit(model_train.sentiments)
train_encode_y = label_coder.transform(model_train.sentiments)
test_encode_y = label_coder.transform(model_test.sentiments)


from keras.utils import pad_sequences
from keras.layers import LSTM


# Get the matrix to same array of shape
sequential = models.Sequential()
```

```
sequential = models.Sequential()
train = preprocessing.sequence.pad_sequences(train_matrix_x,maxlen=50)
test = preprocessing.sequence.pad_sequences(test_matrix_x,maxlen=50)


# Add layers to the model
sequential.add(layers.Embedding(size_vocabulary, 15))
#sequential.add(layers.Flatten())
#sequential.add(layers.Dropout(rate=0.20))
sequential.add(layers.LSTM(15))
sequential.add(layers.Dropout(rate=0.20))
sequential.add(layers.Dense(1, activation='sigmoid'))


sequential.summary()
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_15 (Embedding)    (None, None, 15)          225000

 lstm_7 (LSTM)               (None, 15)                1860

 dropout_8 (Dropout)         (None, 15)                0

 dense_27 (Dense)            (None, 1)                 16

=================================================================
Total params: 226,876
Trainable params: 226,876
Non-trainable params: 0
_________________________________________________________________
```

```
# Compile the model
sequential.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])


# Build the model
LSTM_epoch = sequential.fit(train_matrix_x,
                    train_encode_y,
                    epochs=3,
                    batch_size=128,
                    validation_split=0.2)
```

```
Epoch 1/3
87/87 [==============================] - 40s 436ms/step - loss: 0.6933 - accuracy: 0.
Epoch 2/3
87/87 [==============================] - 37s 421ms/step - loss: 0.6932 - accuracy: 0.
Epoch 3/3
87/87 [==============================] - 35s 399ms/step - loss: 0.6931 - accuracy: 0.
```

```
# Get classification report for the model
sequential_prediction = sequential.predict(test)
label = [1 if value >= 0.5 else 0.0 for value in sequential_prediction]
print(classification_report(test_encode_y, label))
```

```
     110/110 [==============================] - 1s 2ms/step
                   precision    recall  f1-score   support

                0       0.00      0.00      0.00      1590
                1       0.55      1.00      0.71      1919

            accuracy                           0.55      3509
           macro avg       0.27      0.50      0.35      3509
        weighted avg       0.30      0.55      0.39      3509

     /opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: Undef
       _warn_prf(average, modifier, msg_start, len(result))
     /opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: Undef
       _warn_prf(average, modifier, msg_start, len(result))
     /opt/conda/lib/python3.7/site-packages/sklearn/metrics/_classification.py:1318: Undef
       _warn_prf(average, modifier, msg_start, len(result))
```

LSTM Analysis:

After comparing the two results, I have concluded that the LSTM has less accuracy than sequential model. However, I noted this is because the classification report for the LSTM has some errors since the 0 in the classification report does not have a precision, recall, or f1-score. Furthermore, I could not train the LSTM a lot, since it would have token hours to train it (which is why I had put epoch to 3).

Embeddings

```
# Get the matrix to same array of shape
sequential = models.Sequential()
train = preprocessing.sequence.pad_sequences(train_matrix_x,maxlen=50)
test = preprocessing.sequence.pad_sequences(test_matrix_x,maxlen=50)


sequential.add(layers.Embedding(20000, 5 , input_length=15))
sequential.add(layers.Flatten())
sequential.add(layers.Dropout(rate=0.20))
sequential.add(layers.Dense(15))
```

```
sequential.add(layers.Dense(1, activation='sigmoid'))

sequential.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])

LSTM_epoch = sequential.fit(train_matrix_x,
                    test_matrix_x,
                    epochs=3,
                    batch_size=128,
                    validation_split=0.2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_479/2267718780.py in <module>
     13                     epochs=3,
     14                     batch_size=128,
---> 15                     validation_split=0.2)

 1 frames
/opt/conda/lib/python3.7/site-packages/keras/engine/data_adapter.py in
_check_data_cardinality(data)
   1846             )
   1847         msg += "Make sure all arrays contain the same number of samples."
-> 1848         raise ValueError(msg)
   1849
   1850

ValueError: Data cardinality is ambiguous:
  x sizes: 11064
  y sizes: 3509
Make sure all arrays contain the same number of samples.
```

Colab paid products - Cancel contracts here