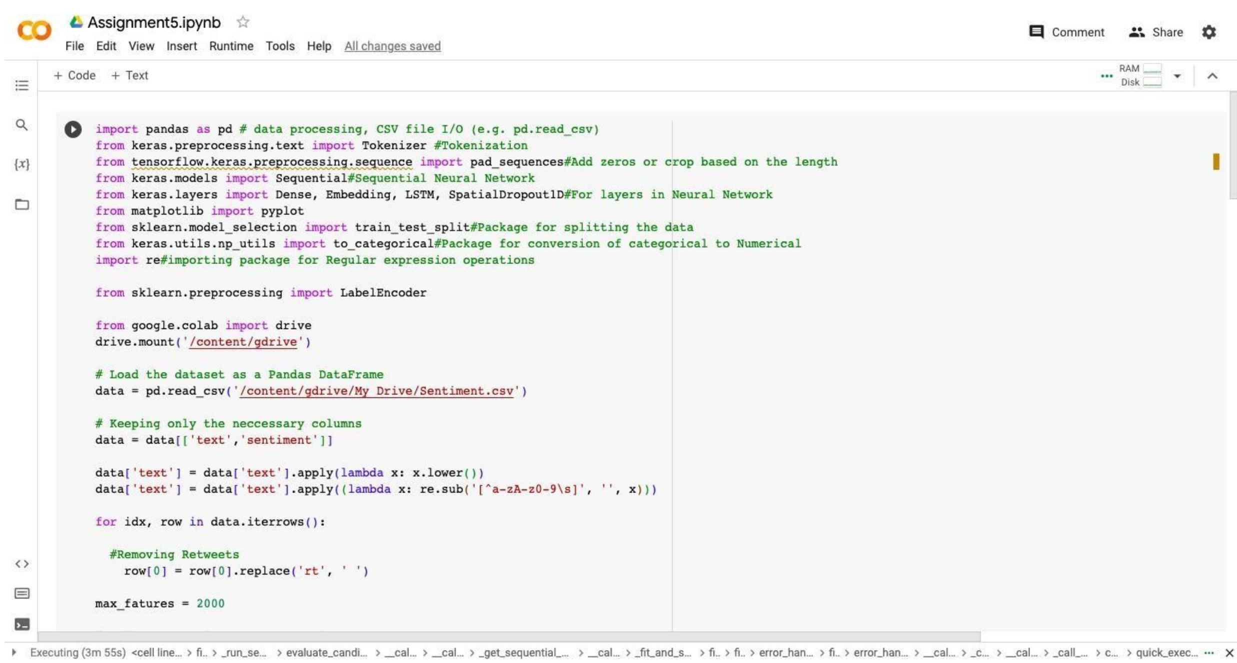
NN&DeepLearning_ ASSIGNMENT 5

Name: Rishitha Reddy Likki
ID:700748512
Github link:
https://github.com/rishithalikki/assign5
video link:
https://drive.google.com/drive/my-drive

1. Save the model and use the saved **model to predict on new text data (ex, "A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump")**

Ans:
Running the provided code SentimentAnalysis.py and the output is :



```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.text import Tokenizer #Tokenization
from tensorflow.keras.preprocessing.sequence import pad_sequences#Add zeros or crop based on the length
from keras.models import Sequential#Sequential Neural Network
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D#For layers in Neural Network
from matplotlib import pyplot
from sklearn.model_selection import train_test_split#Package for splitting the data
from keras.utils.np_utils import to_categorical#Package for conversion of categorical to Numerical
import re#importing package for Regular expression operations

from sklearn.preprocessing import LabelEncoder

from google.colab import drive
drive.mount('/content/gdrive')

# Load the dataset as a Pandas DataFrame
data = pd.read_csv('/content/gdrive/My Drive/Sentiment.csv')

# Keeping only the neccessary columns
data = data[['text','sentiment']]

data['text'] = data['text'].apply(lambda x: x.lower())
data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

for idx, row in data.iterrows():

  #Removing Retweets
    row[0] = row[0].replace('rt', ' ')

max_fatures = 2000
```

Assignment5.ipynb

```python
max_fatures = 2000

#Maximum words is 2000 to tokenize sentence
tokenizer = Tokenizer(num_words=max_fatures, split=' ')
tokenizer.fit_on_texts(data['text'].values)

#taking values to feature matrix
X = tokenizer.texts_to_sequences(data['text'].values)

#Padding the feature matrix
X = pad_sequences(X)

embed_dim = 128#Dimension of the Embedded layer
lstm_out = 196#Long short-term memory (LSTM) layer neurons

def createmodel():
    model = Sequential()#Sequential Neural Network
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1]))#input dimension 2000 Neurons, output
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))#Drop out 20%, 196 output Neurons, recurrent dropout
    model.add(Dense(3,activation='softmax'))#3 output neurons[positive, Neutral, Negative], softmax as activation
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])#Compiling the model
    return model
# print(model.summary())

labelencoder = LabelEncoder()#Applying label Encoding on the label matrix
integer_encoded = labelencoder.fit_transform(data['sentiment'])#fitting the model
y = to_categorical(integer_encoded)
X_train, X_test, Y_train, Y_test = train_test_split(X,y, test_size = 0.33, random_state = 42)

batch_size = 32
model = createmodel()#Function call to Sequential Neural Network
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size) #evaluating the model
```

Executing (4m 25s) <cell line: 12> > fit() > _run_search() > evaluate_candidates() > __call__() > __call__() > _get_sequential_output() > __call__() > _fit_and_score() > fit() > fit() > error_handler()

Assignment5.ipynb

```python
batch_size = 32
model = createmodel()#Function call to Sequential Neural Network
model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2)
score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size) #evaluating the model
print(score)
print(acc)
print(model.metrics_names)#metrics of the model
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
291/291 - 52s - loss: 0.8245 - accuracy: 0.6449 - 52s/epoch - 179ms/step
144/144 - 3s - loss: 0.7550 - accuracy: 0.6765 - 3s/epoch - 21ms/step
0.7550472021102905
0.6764962673187256
['loss', 'accuracy']
```

#1. Saving the model and using it to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump”) and the output is:

Assignment5.ipynb

```python
#1. Save the model and use the saved model to predict on new text data (ex, “A lot of good things are happening. We are respected again throughout the world

model.save('sentimentAnalysis.h5') #Saving the model
from keras.models import load_model #Importing the package for importing the saved model
model= load_model('sentimentAnalysis.h5') #loading the saved model

print(integer_encoded)
print(data['sentiment'])
```

```
[1 2 1 ... 2 0 2]
0         Neutral
1        Positive
2         Neutral
3        Positive
4        Positive
           ...
13866    Negative
13867    Positive
13868    Positive
13869    Negative
13870    Positive
Name: sentiment, Length: 13871, dtype: object
```

Predicting on the test data

```
# Predicting on the text data

import numpy as np
sentence = ['A lot of good things are happening. We are respected again throughout the world, and that is a great thing.@realDonaldTrump']
sentence = tokenizer.texts_to_sequences(sentence) # Tokenizing the sentence
sentence = pad_sequences(sentence, maxlen=28, dtype='int32', value=0) # Padding the sentence
sentiment_probs = model.predict(sentence, batch_size=1, verbose=2)[0] # Predicting the sentence text
sentiment = np.argmax(sentiment_probs)

print(sentiment_probs)
if sentiment == 0:
    print("Neutral")
elif sentiment < 0:
    print("Negative")
elif sentiment > 0:
    print("Positive")
else:
    print("Cannot be determined")
```

```
1/1 - 0s - 107ms/epoch - 107ms/step
[0.6263405  0.16177209 0.2118874 ]
Neutral
```

2. Apply GridSearchCV on the source code provided in the class

Ans:

```
#2. Apply GridSearchCV on the source code provided in the class

from keras.wrappers.scikit_learn import KerasClassifier #importing Keras classifier
from sklearn.model_selection import GridSearchCV #importing Grid search CV

model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper parameters
batch_size= [10, 20, 40] #hyper parameter batch_size
epochs = [1, 2] #hyper parameter no. of epochs
param_grid= {'batch_size':batch_size, 'epochs':epochs} #creating dictionary for batch size, no. of epochs
grid  = GridSearchCV(estimator=model, param_grid=param_grid) #Applying dictionary with hyper parameters

grid_result= grid.fit(X_train,Y_train) #Fitting the model

# summarize results
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_)) #best score, best hyper parameters
```

Output:

Assignment5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
<ipython-input-10-7e316dc5024d>:6: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See https
  model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper parameters
744/744 - 106s - loss: 0.8242 - accuracy: 0.6503 - 106s/epoch - 142ms/step
186/186 - 3s - loss: 0.7687 - accuracy: 0.6654 - 3s/epoch - 18ms/step
744/744 - 103s - loss: 0.8196 - accuracy: 0.6476 - 103s/epoch - 139ms/step
186/186 - 3s - loss: 0.7717 - accuracy: 0.6767 - 3s/epoch - 17ms/step
744/744 - 102s - loss: 0.8247 - accuracy: 0.6458 - 102s/epoch - 137ms/step
186/186 - 3s - loss: 0.7555 - accuracy: 0.6789 - 3s/epoch - 15ms/step
744/744 - 104s - loss: 0.8249 - accuracy: 0.6445 - 104s/epoch - 140ms/step
186/186 - 3s - loss: 0.7552 - accuracy: 0.6765 - 3s/epoch - 15ms/step
744/744 - 104s - loss: 0.8185 - accuracy: 0.6464 - 104s/epoch - 140ms/step
186/186 - 3s - loss: 0.7675 - accuracy: 0.6712 - 3s/epoch - 15ms/step
Epoch 1/2
744/744 - 103s - loss: 0.8267 - accuracy: 0.6504 - 103s/epoch - 139ms/step
Epoch 2/2
744/744 - 101s - loss: 0.6804 - accuracy: 0.7139 - 101s/epoch - 136ms/step
186/186 - 3s - loss: 0.7677 - accuracy: 0.6885 - 3s/epoch - 15ms/step
Epoch 1/2
744/744 - 103s - loss: 0.8183 - accuracy: 0.6427 - 103s/epoch - 139ms/step
Epoch 2/2
744/744 - 99s - loss: 0.6746 - accuracy: 0.7101 - 99s/epoch - 133ms/step
186/186 - 3s - loss: 0.7454 - accuracy: 0.6724 - 3s/epoch - 14ms/step
Epoch 1/2
744/744 - 104s - loss: 0.8223 - accuracy: 0.6473 - 104s/epoch - 140ms/step
Epoch 2/2
744/744 - 99s - loss: 0.6802 - accuracy: 0.7139 - 99s/epoch - 133ms/step
186/186 - 4s - loss: 0.7583 - accuracy: 0.6783 - 4s/epoch - 24ms/step
Epoch 1/2
744/744 - 102s - loss: 0.8254 - accuracy: 0.6437 - 102s/epoch - 137ms/step
Epoch 2/2
744/744 - 99s - loss: 0.6748 - accuracy: 0.7139 - 99s/epoch - 132ms/step
186/186 - 3s - loss: 0.7467 - accuracy: 0.6830 - 3s/epoch - 15ms/step
Epoch 1/2
744/744 - 102s - loss: 0.8176 - accuracy: 0.6436 - 102s/epoch - 137ms/step
Epoch 2/2
744/744 - 98s - loss: 0.6675 - accuracy: 0.7186 - 98s/epoch - 132ms/step
```

✓ 59m 2s completed at 12:13

Assignment5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
Epoch 2/2
744/744 - 98s - loss: 0.6675 - accuracy: 0.7186 - 98s/epoch - 132ms/step
186/186 - 3s - loss: 0.7822 - accuracy: 0.6679 - 3s/epoch - 14ms/step
372/372 - 58s - loss: 0.8363 - accuracy: 0.6457 - 58s/epoch - 156ms/step
93/93 - 2s - loss: 0.7821 - accuracy: 0.6751 - 2s/epoch - 20ms/step
372/372 - 60s - loss: 0.8313 - accuracy: 0.6451 - 60s/epoch - 162ms/step
93/93 - 2s - loss: 0.7537 - accuracy: 0.6740 - 2s/epoch - 24ms/step
372/372 - 62s - loss: 0.8323 - accuracy: 0.6407 - 62s/epoch - 166ms/step
93/93 - 2s - loss: 0.7546 - accuracy: 0.6902 - 2s/epoch - 21ms/step
372/372 - 60s - loss: 0.8351 - accuracy: 0.6399 - 60s/epoch - 162ms/step
93/93 - 2s - loss: 0.7620 - accuracy: 0.6679 - 2s/epoch - 21ms/step
372/372 - 59s - loss: 0.8299 - accuracy: 0.6447 - 59s/epoch - 160ms/step
93/93 - 2s - loss: 0.7785 - accuracy: 0.6604 - 2s/epoch - 20ms/step
Epoch 1/2
372/372 - 59s - loss: 0.8354 - accuracy: 0.6414 - 59s/epoch - 159ms/step
Epoch 2/2
372/372 - 55s - loss: 0.6815 - accuracy: 0.7104 - 55s/epoch - 147ms/step
93/93 - 2s - loss: 0.7310 - accuracy: 0.6740 - 2s/epoch - 19ms/step
Epoch 1/2
372/372 - 58s - loss: 0.8307 - accuracy: 0.6384 - 58s/epoch - 156ms/step
Epoch 2/2
372/372 - 55s - loss: 0.6830 - accuracy: 0.7140 - 55s/epoch - 149ms/step
93/93 - 2s - loss: 0.7280 - accuracy: 0.6923 - 2s/epoch - 19ms/step
Epoch 1/2
372/372 - 59s - loss: 0.8267 - accuracy: 0.6476 - 59s/epoch - 158ms/step
Epoch 2/2
372/372 - 55s - loss: 0.6776 - accuracy: 0.7125 - 55s/epoch - 148ms/step
93/93 - 3s - loss: 0.7528 - accuracy: 0.6842 - 3s/epoch - 33ms/step
Epoch 1/2
372/372 - 58s - loss: 0.8360 - accuracy: 0.6432 - 58s/epoch - 157ms/step
Epoch 2/2
372/372 - 55s - loss: 0.6799 - accuracy: 0.7119 - 55s/epoch - 147ms/step
93/93 - 2s - loss: 0.7390 - accuracy: 0.6787 - 2s/epoch - 20ms/step
Epoch 1/2
372/372 - 59s - loss: 0.8336 - accuracy: 0.6410 - 59s/epoch - 159ms/step
Epoch 2/2
```

✓ 59m 2s completed at 12:13

Assignment5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
Epoch 2/2
372/372 - 54s - loss: 0.6698 - accuracy: 0.7189 - 54s/epoch - 144ms/step
93/93 - 2s - loss: 0.7805 - accuracy: 0.6749 - 2s/epoch - 20ms/step
186/186 - 39s - loss: 0.8414 - accuracy: 0.6395 - 39s/epoch - 208ms/step
47/47 - 2s - loss: 0.7664 - accuracy: 0.6638 - 2s/epoch - 33ms/step
186/186 - 38s - loss: 0.8411 - accuracy: 0.6357 - 38s/epoch - 204ms/step
47/47 - 1s - loss: 0.7856 - accuracy: 0.6697 - 1s/epoch - 29ms/step
186/186 - 38s - loss: 0.8400 - accuracy: 0.6349 - 38s/epoch - 206ms/step
47/47 - 1s - loss: 0.7640 - accuracy: 0.6810 - 1s/epoch - 28ms/step
186/186 - 39s - loss: 0.8496 - accuracy: 0.6374 - 39s/epoch - 209ms/step
47/47 - 1s - loss: 0.7559 - accuracy: 0.6733 - 1s/epoch - 30ms/step
186/186 - 36s - loss: 0.8521 - accuracy: 0.6308 - 36s/epoch - 196ms/step
47/47 - 1s - loss: 0.7824 - accuracy: 0.6631 - 1s/epoch - 28ms/step
Epoch 1/2
186/186 - 38s - loss: 0.8545 - accuracy: 0.6359 - 38s/epoch - 204ms/step
Epoch 2/2
186/186 - 33s - loss: 0.6910 - accuracy: 0.7011 - 33s/epoch - 177ms/step
47/47 - 2s - loss: 0.7376 - accuracy: 0.6794 - 2s/epoch - 35ms/step
Epoch 1/2
186/186 - 37s - loss: 0.8379 - accuracy: 0.6410 - 37s/epoch - 201ms/step
Epoch 2/2
186/186 - 35s - loss: 0.6960 - accuracy: 0.7024 - 35s/epoch - 188ms/step
47/47 - 1s - loss: 0.7402 - accuracy: 0.6794 - 1s/epoch - 30ms/step
Epoch 1/2
186/186 - 37s - loss: 0.8417 - accuracy: 0.6377 - 37s/epoch - 199ms/step
Epoch 2/2
186/186 - 34s - loss: 0.6891 - accuracy: 0.7053 - 34s/epoch - 184ms/step
47/47 - 1s - loss: 0.7421 - accuracy: 0.6837 - 1s/epoch - 28ms/step
Epoch 1/2
186/186 - 38s - loss: 0.8535 - accuracy: 0.6289 - 38s/epoch - 203ms/step
Epoch 2/2
186/186 - 33s - loss: 0.6904 - accuracy: 0.7053 - 33s/epoch - 176ms/step
47/47 - 1s - loss: 0.7478 - accuracy: 0.6846 - 1s/epoch - 28ms/step
Epoch 1/2
186/186 - 37s - loss: 0.8399 - accuracy: 0.6373 - 37s/epoch - 201ms/step
Epoch 2/2
```

✓ 59m 2s completed at 12:13

Assignment5.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

```
[10]
Epoch 1/2
186/186 - 37s - loss: 0.8399 - accuracy: 0.6373 - 37s/epoch - 201ms/step
Epoch 2/2
186/186 - 33s - loss: 0.6701 - accuracy: 0.7154 - 33s/epoch - 176ms/step
47/47 - 1s - loss: 0.7817 - accuracy: 0.6787 - 1s/epoch - 29ms/step
Epoch 1/2
233/233 - 47s - loss: 0.8280 - accuracy: 0.6453 - 47s/epoch - 200ms/step
Epoch 2/2
233/233 - 42s - loss: 0.6778 - accuracy: 0.7130 - 42s/epoch - 180ms/step
Best: 0.681158 using {'batch_size': 40, 'epochs': 2}
```

Colab paid products - Cancel contracts here

✓ 59m 2s completed at 12:13