NEURAL NETWORK & DEEP LEARNING(CS-5720)

(CRN:31196)

ASSIGNMENT - 5

Name : Vamsi Krishna Mekala
ID : 700742751
Date : 08/01/2023
Github : https://github.com/vamsi-mekala/Neural-networks-assignment-5
Google Drive: https://drive.google.com/file/d/1SxG8MKnPB1TdPGQ6m-VfwCsstZ69mz70/view?usp=sharing

Question 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”)

The above specification are coded in as the model shown below:

```
In [5]: print(model.metrics_names) #metrics of the model

['loss', 'acc']

In [6]: model.save('sentimentAnalysis.h5') #Saving the model

In [7]: from keras.models import load_model #Importing the package for importing the saved model
        model= load_model('sentimentAnalysis.h5') #loading the saved model

In [8]: 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
```

In [9]:

```
sentence = ['A lot of good things are happening. We are respected again throughout the world, and that is a great thing.@realDona
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")
```

```
[0.6813752  0.1598489  0.15877591]
Neutral
```

GridSearchCV on the source code

Apply GridSearchCV on the source code:

In [10]:

```
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 paramete
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
```

```
Epoch 1/1
 - 42s - loss: 0.8364 - acc: 0.6385
Epoch 1/1
 - 62s - loss: 0.8354 - acc: 0.6409
Epoch 1/2
 - 73s - loss: 0.8361 - acc: 0.6402
Epoch 2/2
 - 71s - loss: 0.6902 - acc: 0.7018
Epoch 1/2
 - 76s - loss: 0.8396 - acc: 0.6371
Epoch 2/2
 - 73s - loss: 0.7015 - acc: 0.7041
Epoch 1/2
 - 83s - loss: 0.8327 - acc: 0.6415
Epoch 2/2
```