CS5720

Neural Networks & Deep Learning - ICP-5

Name: Madhavi Mamidala

#700: 700744319

Username: MXM43190

GitHub Link:
https://github.com/MadhaviMamidala/CS5720_ASSIGNMENT5_700744319/blob/main/700744319_Madhavi_NNDL_Assign5.ipynb

700744319_Madhavi_NNDL_Assign3.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
import pandas as pd #Basic packages for creating dataframes and loading dataset
import numpy as np

import matplotlib.pyplot as plt #Package for visualization

import re #importing package for Regular expression operations

from sklearn.model_selection import train_test_split #Package for splitting the data

from sklearn.preprocessing import LabelEncoder #Package for conversion of categorical to Numerical

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 keras.utils.np_utils import to_categorical
```

[2]
```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
[3] import pandas as pd

    path_to_csv = '/content/gdrive/MyDrive/Sentiment.csv'
    # Load the dataset as a Pandas DataFrame
    dataset = pd.read_csv(path_to_csv, header=0)

    # Select only the necessary columns 'text' and 'sentiment'
    mask = dataset.columns.isin(['text', 'sentiment'])
    data = dataset.loc[:, mask]

    # Keeping only the necessary columns
```

```
# cleaning text , removing all special characters

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

```
<ipython-input-4-8bc15de6ba7d>:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['text'] = data['text'].apply(lambda x: x.lower())
<ipython-input-4-8bc15de6ba7d>:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['text'] = data['text'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))
```

```
[5] for idx, row in data.iterrows():
        row[0] = row[0].replace('rt', ' ') #Removing Retweets
```

```
[6] max_fatures = 2000
    tokenizer = Tokenizer(num_words=max_fatures, split=' ') #Maximum words is 2000 to tokenize sentence
    tokenizer.fit_on_texts(data['text'].values)
    X = tokenizer.texts_to_sequences(data['text'].values) #taking values to feature matrix
```

```
[7] X = pad_sequences(X) #Padding the feature matrix

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

```
# Sequential Model Creation

def createmodel():
    model = Sequential() #Sequential Neural Network
    model.add(Embedding(max_fatures, embed_dim,input_length = X.shape[1])) #input dimension 2000 Neurons, output dimension 128 Neurons
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2)) #Drop out 20%, 196 output Neurons, recurrent dropout 20%
    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())
```

```
[9] # Label Encoding of values
    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) #67% training data, 33% test data split
```

```
[10] # Model Training
     batch_size = 32 #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) #verbose the higher, the more messages
     score,acc = model.evaluate(X_test,Y_test,verbose=2,batch_size=batch_size) #evaluating the model
     print(score)
     print(acc)
```

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
291/291 - 59s - loss: 0.8294 - accuracy: 0.6450 - 59s/epoch - 202ms/step
144/144 - 2s - loss: 0.7528 - accuracy: 0.6710 - 2s/epoch - 12ms/step
0.7528498768806458
0.6710354089736938
```

```
[11] print(model.metrics_names) #metrics of the model
```

```
['loss', 'accuracy']
```

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”)

```
[13] model.save('sentimentAnalysis.h5') #Saving the model
```

+ Code + Text

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

```
WARNING:tensorflow:Layer lstm will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
```

```
[14] 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
```

```python
# Predicting on the text data
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 - 370ms/epoch - 370ms/step
[0.733563   0.12827614 0.13816082]
Neutral
```

```python
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
```

```
<ipython-input-16-6c99b49150f4>:4: DeprecationWarning: KerasClassifier is deprecated, use Sci-Keras (https://github.com/adriangb/scikeras) instead. See https://www.adriangb.com/
  model = KerasClassifier(build_fn=createmodel,verbose=2) #initiating model to test performance by applying multiple hyper parameters
WARNING:tensorflow:Layer lstm_1 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 115s - loss: 0.8252 - accuracy: 0.6438 - 115s/epoch - 155ms/step
186/186 - 3s - loss: 0.7521 - accuracy: 0.6622 - 3s/epoch - 16ms/step
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 113s - loss: 0.8207 - accuracy: 0.6493 - 113s/epoch - 151ms/step
186/186 - 3s - loss: 0.7599 - accuracy: 0.6746 - 3s/epoch - 16ms/step
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 112s - loss: 0.8261 - accuracy: 0.6438 - 112s/epoch - 151ms/step
186/186 - 2s - loss: 0.7709 - accuracy: 0.6751 - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_4 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 111s - loss: 0.8249 - accuracy: 0.6469 - 111s/epoch - 149ms/step
186/186 - 2s - loss: 0.7637 - accuracy: 0.6738 - 2s/epoch - 12ms/step
```

```
186/186 - 3s - loss: 0.7521 - accuracy: 0.6622 - 3s/epoch - 16ms/step
WARNING:tensorflow:Layer lstm_2 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when runnir
744/744 - 113s - loss: 0.8207 - accuracy: 0.6493 - 113s/epoch - 151ms/step
186/186 - 3s - loss: 0.7599 - accuracy: 0.6746 - 3s/epoch - 16ms/step
WARNING:tensorflow:Layer lstm_3 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 112s - loss: 0.8261 - accuracy: 0.6438 - 112s/epoch - 151ms/step
186/186 - 2s - loss: 0.7709 - accuracy: 0.6751 - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_4 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 111s - loss: 0.8249 - accuracy: 0.6469 - 111s/epoch - 149ms/step
186/186 - 2s - loss: 0.7637 - accuracy: 0.6738 - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_5 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
744/744 - 124s - loss: 0.8216 - accuracy: 0.6433 - 124s/epoch - 166ms/step
186/186 - 2s - loss: 0.7893 - accuracy: 0.6464 - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_6 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Epoch 1/2
744/744 - 115s - loss: 0.8195 - accuracy: 0.6500 - 115s/epoch - 154ms/step
Epoch 2/2
744/744 - 100s - loss: 0.6742 - accuracy: 0.7154 - 100s/epoch - 134ms/step
186/186 - 2s - loss: 0.7385 - accuracy: 0.6789 - 2s/epoch - 12ms/step
WARNING:tensorflow:Layer lstm_7 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Epoch 1/2
744/744 - 114s - loss: 0.8251 - accuracy: 0.6469 - 114s/epoch - 153ms/step
Epoch 2/2
744/744 - 101s - loss: 0.6831 - accuracy: 0.7113 - 101s/epoch - 136ms/step
186/186 - 3s - loss: 0.7479 - accuracy: 0.6622 - 3s/epoch - 14ms/step
WARNING:tensorflow:Layer lstm_8 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Epoch 1/2
744/744 - 115s - loss: 0.8268 - accuracy: 0.6457 - 115s/epoch - 155ms/step
Epoch 2/2
744/744 - 102s - loss: 0.6753 - accuracy: 0.7146 - 102s/epoch - 138ms/step
186/186 - 3s - loss: 0.7547 - accuracy: 0.6772 - 3s/epoch - 18ms/step
WARNING:tensorflow:Layer lstm_9 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Epoch 1/2
744/744 - 114s - loss: 0.8222 - accuracy: 0.6451 - 114s/epoch - 154ms/step
Epoch 2/2
744/744 - 105s - loss: 0.6719 - accuracy: 0.7116 - 105s/epoch - 141ms/step
186/186 - 3s - loss: 0.7599 - accuracy: 0.6728 - 3s/epoch - 17ms/step
WARNING:tensorflow:Layer lstm_10 will not use cuDNN kernels since it doesn't meet the criteria. It will use a generic GPU kernel as fallback when running on GPU.
Epoch 1/2
```