CS5590/490 - Python and Deep Learning Programming
In Class Programming Report - 12
Class ID 24 - Anurag Thantharate
Date Submitted: 04/26/2020

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”)

```
# Import libraries
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re
from sklearn.preprocessing import LabelEncoder

# Read the input data from csv file
data = pd.read_csv('Sentiment.csv')

# cleaning the data set to identify features that are important
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():
    row[0] = row[0].replace('rt', ' ')

max_features = 2000
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values)

X = pad_sequences(X)

# Creating the model to be fit
embed_dim = 128
lstm_out = 196
```

```
def createmodel():
    model = Sequential()
    model.add(Embedding(max_features, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
    return model
# print(model.summary())

# Identify the data into training and test sets
labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
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)

# Fitting the training data on the model defined
batch_size = 32
model = createmodel()
history = model.fit(X_train, Y_train, epochs = 10, batch_size=batch_size, verbose = 2)

# Evaluation of the performance of the model fit
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size)
print('The obtained score from the model fit is ', score)
print('The accuracy of the model fit is ', acc)
print(model.metrics_names)

# Saving the model to be applied on varying test data
modelFit = model.save('modelFit.h5')

# Reading data input as a new sentence
newData = "A lot of good things are happening. We are respected again throughout the world, and that's a great thing.@realDonaldTrump"
text = newData

# Evaluation of the performance of the model fit
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size)
print('The score of the new test is ', score)
print('The accuracy of the new text is ', acc)
```

```
Epoch 1/10
 - 28s - loss: 0.8322 - accuracy: 0.6475
Epoch 2/10
 - 29s - loss: 0.6863 - accuracy: 0.7101
Epoch 3/10
 - 30s - loss: 0.6204 - accuracy: 0.7450
Epoch 4/10
 - 30s - loss: 0.5778 - accuracy: 0.7585
Epoch 5/10
 - 24s - loss: 0.5325 - accuracy: 0.7834
Epoch 6/10
 - 22s - loss: 0.4918 - accuracy: 0.7996
Epoch 7/10
 - 26s - loss: 0.4577 - accuracy: 0.8139
Epoch 8/10
 - 27s - loss: 0.4263 - accuracy: 0.8264
Epoch 9/10
 - 24s - loss: 0.3992 - accuracy: 0.8399
Epoch 10/10
 - 23s - loss: 0.3719 - accuracy: 0.8493
The obtained score from the model fit is  1.1187564817356095
The accuracy of the model fit is  0.6546527147293091
['loss', 'accuracy']
The score of the new test is  1.1187564817356095
The accuracy of the new text is  0.6546527147293091
```

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

```
# Import libraries
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from keras.wrappers.scikit_learn import KerasClassifier
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re
from sklearn.preprocessing import LabelEncoder

# Read the input data from csv file
data = pd.read_csv('Sentiment.csv')

# Keeping only the necessary columns - cleaning the data set to identify features that are important
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():
    row[0] = row[0].replace('rt', ' ')

max_features = 2000
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(data['text'].values)
X = tokenizer.texts_to_sequences(data['text'].values)

X = pad_sequences(X)

# Creating the model to be fit
embed_dim = 128
lstm_out = 196
```

```python
# Define model used along with the appropriate layers
def createmodel():
    model = Sequential()
    model.add(Embedding(max_features, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',metrics=['accuracy'])
    return model
# print(model.summary())

# Identify the data into training and test sets
labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['sentiment'])
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)

# Fitting the training data on the model defined
batch_size = 32
model = createmodel()
history = model.fit(X_train, Y_train, epochs = 1, batch_size=batch_size, verbose = 2, validation_data=(X_test, Y_test))

# Evaluation of the performance of the model fit
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size)
print('The score obtained from the model fit is ', score)
print('The accuracy of the model fit is ', acc)
print(model.metrics_names)

# Saving the model to be applied on varying test data
modelFit = model.save('modelFit.h5')

# Performing grid search analysis
model = KerasClassifier(build_fn=createmodel, verbose=0)
batch_size = [10, 20, 40]
epochs = [1, 2, 3]
param_grid = dict(batch_size=batch_size, epochs=epochs)
# Import library used to do grid search
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid_result = grid.fit(X_train, Y_train)

# Summarize results obtained from the grid search
print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
```

```
C:\ProgramData\Anaconda3\envs\tensorflow_env\python.exe "C:/Users/ld630534/Documents/Anurag Projects/Spring 2020/Python with DL/DLICI
Using TensorFlow backend.
2020-04-26 19:13:25.504951: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU supports instructions that this TensorFlow
C:\ProgramData\Anaconda3\envs\tensorflow_env\lib\site-packages\tensorflow_core\python\framework\indexed_slices.py:433: UserWarning:
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
Train on 9293 samples, validate on 4578 samples
Epoch 1/1
 - 29s - loss: 0.8260 - accuracy: 0.6452 - val_loss: 0.7730 - val_accuracy: 0.6745
The score obtained from the model fit is  0.7729733882446693
The accuracy of the model fit is  0.6745303869247437
['loss', 'accuracy']
C:\ProgramData\Anaconda3\envs\tensorflow_env\lib\site-packages\tensorflow_core\python\framework\indexed_slices.py:433: UserWarning:
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
```

"Converting sparse IndexedSlices to a dense Tensor of unknown shape. "

```
Best: 0.680404 using {'batch_size': 20, 'epochs': 3}

Process finished with exit code 0
```

3.Apply the code on spam data set available in the source code (text classification on the spam.csvdata set)

```
# Import Libraries
import pandas as pd
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
from keras.utils.np_utils import to_categorical
import re
from sklearn.preprocessing import LabelEncoder

# Read the input data from csv file
data = pd.read_csv('spam.csv', encoding='latin1')

# cleaning the data set to identify features that are important
data = data[['v1','v2']]

data['v2'] = data['v2'].apply(lambda x: x.lower())
data['v2'] = data['v2'].apply((lambda x: re.sub('[^a-zA-z0-9\s]', '', x)))

for idx, row in data.iterrows():
    row[0] = row[0].replace('rt', ' ')

max_features = 2000
tokenizer = Tokenizer(num_words=max_features, split=' ')
tokenizer.fit_on_texts(data['v2'].values)
X = tokenizer.texts_to_sequences(data['v2'].values)

X = pad_sequences(X)
```

```
# Creating the model to be fit
embed_dim = 128
lstm_out = 196
# Define model used along with the appropriate layers
def createmodel():
    model = Sequential()
    model.add(Embedding(max_features, embed_dim,input_length = X.shape[1]))
    model.add(LSTM(lstm_out, dropout=0.2, recurrent_dropout=0.2))
    model.add(Dense(2,activation='softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer='adam',metrics = ['accuracy'])
    return model
# print(model.summary())

# Identify the data into training and test sets
labelencoder = LabelEncoder()
integer_encoded = labelencoder.fit_transform(data['v1'])
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)

# Fitting the training data on the model defined
batch_size = 32
model = createmodel()
model.fit(X_train, Y_train, epochs = 10, batch_size=batch_size, verbose = 2)

# Evaluation of the performance of the model fit
score, acc = model.evaluate(X_test, Y_test, verbose=2, batch_size=batch_size)
print('The score obtained from the model fit is ', score)
print('The accuracy of the model fit is ', acc)
print(model.metrics_names)
```

```
C:\ProgramData\Anaconda3\envs\tensorflow_env\python.exe "C:/Users/1d630534/Documents/Anu
Using TensorFlow backend.
2020-04-26 20:26:21.955707: I tensorflow/core/platform/cpu_feature_guard.cc:142] Your CPU
C:\ProgramData\Anaconda3\envs\tensorflow_env\lib\site-packages\tensorflow_core\python\fr
  "Converting sparse IndexedSlices to a dense Tensor of unknown shape. "
Epoch 1/10
 - 55s - loss: 0.1828 - accuracy: 0.9405
Epoch 2/10
 - 51s - loss: 0.0421 - accuracy: 0.9866
Epoch 3/10
 - 43s - loss: 0.0196 - accuracy: 0.9949
Epoch 4/10
 - 43s - loss: 0.0114 - accuracy: 0.9968
Epoch 5/10
 - 45s - loss: 0.0083 - accuracy: 0.9968
Epoch 6/10
 - 45s - loss: 0.0053 - accuracy: 0.9981
Epoch 7/10
 - 51s - loss: 0.0064 - accuracy: 0.9979
Epoch 8/10
 - 49s - loss: 0.0017 - accuracy: 0.9997
Epoch 9/10
 - 47s - loss: 8.6290e-04 - accuracy: 1.0000
Epoch 10/10
 - 44s - loss: 6.0114e-04 - accuracy: 1.0000
The score obtained from the model fit is  0.14568295737720183
The accuracy of the model fit is  0.9825992584228516
['loss', 'accuracy']

Process finished with exit code 0
```
