Assignment_4

November 17, 2022

Assignment Date	08 October 2022
Student Name	Harish G
Student Roll Number	2019504026
Maximum Marks	2 Marks

1.Download the dataset

2.Import required library

[4]:
```
import pandas as pd import
numpy as np
import matplotlib.pyplot as plt import
seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence from
keras.utils import pad_sequences from keras.utils
import to_categorical from keras.callbacks import
EarlyStopping
```

3.Read Dataset and do preprocessing

[6]:
```
data=pd.read_csv('/content/spam.csv',encoding='latin')
```

[7]:
```
df =pd.read_csv('/content/spam.csv',delimiter=',',encoding='latin-1') df.head()
```

[7]:

	v1	v2	Unnamed: 2	\
0	ham	Go until jurong point, crazy.. Available only …	Na N	
1	ham	Ok lar… Joking wif u oni…	NaN	
2	spam	Free entry in 2 a wkly comp to win FA Cup fina…	Na N	
3	ham	U dun say so early hor… U c already then say…	NaN	
4	ham	Nah I don't think he goes to usf, he lives aro…	Na N	

	Unnamed: 3	Unnamed: 4
0	Na N	NaN
1	Na N	NaN
2	Na N	NaN
3	Na N	NaN
4	Na N	NaN

[8]:
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed:4'],axis=1,inplace=True) df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571 Data
columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2) memory
usage: 87.2+ KB
```

[9]:
```
# Count of Spam and Ham values
df.groupby(['v1']).size()
```

[9]:
```
v1
ham     4825
spam     747
dtype: int64
```

[10]:
```
# Label Encoding target column
X = df.v2 Y =
df.v1
le = LabelEncoder()
Y =le.fit_transform(Y) Y =
Y.reshape(-1,1)
```

[11]:
```
# Test and train split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

[12]:
```
# Tokenisation function
max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train) sequences_matrix =
pad_sequences(sequences,maxlen=max_len)
```

4. Create Model and 5. Add Layers (LSTM, Dense-(Hidden Layers), Output)

[13]:
```
# Creating LSTM model
inputs = Input(name='inputs',shape=[max_len])
layer =Embedding(max_words,50,input_length=max_len)(inputs) layer =
LSTM(64)(layer)
layer = Dense(256,name='FC1')(layer) layer =
Activation('relu')(layer) layer =
Dropout(0.5)(layer)
layer =Dense(1,name='out_layer')(layer) layer =
Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
```

6.Compile the model & 7.Fit the Model

[14]:
```
model.summary()

model.

  ↪compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
                validation_split=0.2)
```

Model: "model"

Layer (type)	Output Shape	Param #
inputs (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_1 (Activation)	(None, 1)	0

Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0

Epoch 1/10
30/30 [==============================] - 12s 311ms/step - loss: 0.3269 - accuracy: 0.8746 - val_loss: 0.1554 - val_accuracy: 0.9778

```
Epoch 2/10
30/30 [==============================] - 12s 380ms/step - loss: 0.0819 -
accuracy: 0.9794 - val_loss: 0.0461 - val_accuracy:0.9831 Epoch 3/10
30/30 [==============================] - 8s 259ms/step - loss: 0.0474 -
accuracy: 0.9873 - val_loss: 0.0343 - val_accuracy:0.9905 Epoch 4/10
30/30 [==============================] - 8s 261ms/step - loss: 0.0370 -
accuracy: 0.9894 - val_loss: 0.0447 - val_accuracy:0.9895 Epoch 5/10
30/30 [==============================] - 9s 310ms/step - loss: 0.0276 -
accuracy: 0.9918 - val_loss: 0.0340 - val_accuracy:0.9905 Epoch 6/10
30/30 [==============================] - 8s 260ms/step - loss: 0.0224 -
accuracy: 0.9929 - val_loss: 0.0402 - val_accuracy:0.9905 Epoch 7/10
30/30 [==============================] - 8s 260ms/step - loss: 0.0159 -
accuracy: 0.9958 - val_loss: 0.0442 - val_accuracy:0.9916 Epoch 8/10
30/30 [==============================] - 8s 257ms/step - loss: 0.0141 -
accuracy: 0.9960 - val_loss: 0.0433 - val_accuracy:0.9905 Epoch 9/10
30/30 [==============================] - 8s 261ms/step - loss: 0.0108 -
accuracy: 0.9974 - val_loss: 0.0952 - val_accuracy:0.9736 Epoch 10/10
30/30 [==============================] - 8s 260ms/step - loss: 0.0089 -
accuracy: 0.9979 - val_loss: 0.0607 - val_accuracy:0.9884 [14]:
```

<keras.callbacks.History at 0x7f823de6acd0>

8. Save the Model

[15]:
```
model.save('sms_classifier.h5')
```

9. Test the model

[16]:
```
test_sequences = tok.texts_to_sequences(X_test) test_sequences_matrix =
pad_sequences(test_sequences,maxlen=max_len)
```

[17]:
```
accr =model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 1s 23ms/step - loss: 0.1038 -accuracy:
0.9856
```

[18]:
```
print('Test set\n          Loss: {:0.3f}\n          Accuracy:{:0.3f}'.format(accr[0],accr[1]))
```

```
Test set Loss:
  0.104
  Accuracy: 0.986
```