Assignment 4

SMS SPAM Classification

Assignment Date	08 November 2022
Student Name	K.Mohammed Suhail
Student Register Number	620619106016
Maximum Marks	2

1. Downloading the Dataset

https://www.kaggle.com/code/kredy10/simple-lstm-for-text-classification/data

2. Import required library

```
import pandas as pd
import numpy as np
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 to_categorical
from keras.models import load_model
```

3. Read Dataset and do preprocessing

```
df = pd.read_csv('/content/drive/MyDrive/Assignment 3/spam.csv',
delimiter=',',encoding='latin-1')
df.head()
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True) #d
ropping unwanted columns
df.info()
# Count of Spam and Ham values
df.groupby(['v1']).size()
# Label Encoding target column
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
# Test and train split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0
.15)
# 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 = sequence.pad_sequences(sequences,maxlen=max_l
en)
```

```
[3] df = pd.read_csv('/content/drive/MyDrive/Assignment 3/spam.csv',delimiter=',',encoding='latin-1')
```

```
[4] df.head()
```

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

```
[5] df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True) #dropping unwanted columns
```

```
[6] 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
```

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

     v1
     ham     4825
     spam     747
     dtype: int64
```

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

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

```
[10] # 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 = sequence.pad_sequences(sequences,maxlen=max_len)
```

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

```
# Creating LSTM model
inputs = Input(name='InputLayer',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FullyConnectedLayer1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='OutputLayer')(layer)
layer = Activation('sigmoid')(layer)
```

```
[11] # Creating LSTM model
     inputs = Input(name='InputLayer',shape=[max_len])
```

```
[12] layer = Embedding(max_words,50,input_length=max_len)(inputs)
     layer = LSTM(64)(layer)
     layer = Dense(256,name='FullyConnectedLayer1')(layer)
     layer = Activation('relu')(layer)
     layer = Dropout(0.5)(layer)
     layer = Dense(1,name='OutputLayer')(layer)
     layer = Activation('sigmoid')(layer)
```

6. Compile the model

```
model = Model(inputs=inputs,outputs=layer)
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
[13] model = Model(inputs=inputs,outputs=layer)
```

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

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 InputLayer (InputLayer)     [(None, 150)]             0

 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 64)                29440

 FullyConnectedLayer1 (Dense  (None, 256)              16640
 )

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 OutputLayer (Dense)         (None, 1)                 257

 activation_1 (Activation)   (None, 1)                 0


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

_________________________________________________________________
```

```
[15] model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

7. Fit the Model

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
          validation_split=0.2)
```

```
[16] model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
                validation_split=0.2)
```

```
Epoch 1/10
30/30 [==============================] - 13s 288ms/step - loss: 0.3222 - accuracy: 0.8801 - val_loss: 0.1518 - val_accuracy: 0.9515
Epoch 2/10
30/30 [==============================] - 8s 267ms/step - loss: 0.0850 - accuracy: 0.9791 - val_loss: 0.0584 - val_accuracy: 0.9863
Epoch 3/10
30/30 [==============================] - 11s 363ms/step - loss: 0.0390 - accuracy: 0.9900 - val_loss: 0.0530 - val_accuracy: 0.9884
Epoch 4/10
30/30 [==============================] - 8s 266ms/step - loss: 0.0293 - accuracy: 0.9910 - val_loss: 0.0516 - val_accuracy: 0.9895
Epoch 5/10
30/30 [==============================] - 8s 268ms/step - loss: 0.0230 - accuracy: 0.9926 - val_loss: 0.0677 - val_accuracy: 0.9852
Epoch 6/10
30/30 [==============================] - 8s 268ms/step - loss: 0.0164 - accuracy: 0.9947 - val_loss: 0.0723 - val_accuracy: 0.9852
Epoch 7/10
30/30 [==============================] - 8s 268ms/step - loss: 0.0122 - accuracy: 0.9963 - val_loss: 0.0754 - val_accuracy: 0.9852
Epoch 8/10
30/30 [==============================] - 8s 265ms/step - loss: 0.0109 - accuracy: 0.9958 - val_loss: 0.0904 - val_accuracy: 0.9852
Epoch 9/10
30/30 [==============================] - 8s 265ms/step - loss: 0.0060 - accuracy: 0.9979 - val_loss: 0.1023 - val_accuracy: 0.9852
Epoch 10/10
30/30 [==============================] - 8s 267ms/step - loss: 0.0042 - accuracy: 0.9987 - val_loss: 0.1120 - val_accuracy: 0.9873
<keras.callbacks.History at 0x7fa615cafad0>
```

8. Save the Model

```
model.save("model_1")
```

```
[17] model.save("model_1")
```

```
WARNING:absl:Function `_wrapped_model` contains input name(s) InputLayer with unsupported characters which will be renamed to inputlayer in the SavedModel.
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_layer_call_and_return_conditional_losses while saving (showing 2 of 2). These fu
```

9. Test the Model

```
test_sequences = tok.texts_to_sequences(X_test)

test_sequences_matrix  = sequence.pad_sequences(test_sequences,m
axlen=max_len)
accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:0.3f}'.format(accuracy[1]))
y_pred = model.predict(test_sequences_matrix)
y_pred = model.predict(test_sequences_matrix)
print(y_pred[25:40].round(3))
print(Y_test[25:40])
```

```
[18] test_sequences = tok.texts_to_sequences(X_test)
```

```
[19] test_sequences_matrix  = sequence.pad_sequences(test_sequences,maxlen=max_len)
```

```
[20] accuracy = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 1s 34ms/step - loss: 0.0923 - accuracy: 0.9809
```

```
[21] print('Accuracy: {:0.3f}'.format(accuracy[1]))
```

```
Accuracy: 0.981
```

```
[22] y_pred = model.predict(test_sequences_matrix)
```

```
27/27 [==============================] - 2s 34ms/step
```

```
[23] print(y_pred[25:40].round(3))
```

```
[[0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.001]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [1.   ]
 [0.   ]
 [0.   ]
 [0.   ]]
```

```
[24] print(Y_test[25:40])
```

```
[[0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]]
```