```
import pandas as pd
import numpy as np
from keras import utils
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 to_categorical
%matplotlib inline
```

In [4]:

```
df = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
df.head()
```

Out[4]:

	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

Preprocessing

In [5]:

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

```
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
```

In [6]:

```
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWar
ning: Pass the following variable as a keyword arg: x. From version 0.12, t
```

```
he only valid positional argument will be `data`, and passing other argumen
ts without an explicit keyword will result in an error or misinterpretation
.
  FutureWarning
```

Out[6]:

```
Text(0.5, 1.0, 'Number of ham and spam messages')
```

In [7]:

```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

In [8]:

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

In [9]:

```
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 = utils.pad_sequences(sequences,maxlen=max_len)
```

In [10]:

```
sequences_matrix.shape
```

Out[10]:

```
(4736, 150)
```

In [11]:

```
sequences_matrix.ndim
```

Out[11]:

```
2
```

In [12]:

```
sequences_matrix = np.reshape(sequences_matrix,(4736,150,1))
```

In [13]:

```
sequences_matrix.ndim #3d shape verification to proceed to RNN LSTM
```

Out[13]:

```
3
```

RNN Construction

In [14]:

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
```

In [15]:

```
model = Sequential()
model.add(Embedding(max_words,50,input_length=max_len))
```

In [16]:

```
model.add(LSTM(units=64,input_shape =
(sequences_matrix.shape[1],1),return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64))
model.add(Dense(units = 256,activation = 'relu'))
model.add(Dense(units = 1,activation = 'sigmoid'))
```

In [17]:

```
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accu
racy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 150, 50)           50000     
                                                                 
 lstm (LSTM)                 (None, 150, 64)           29440     
                                                                 
 lstm_1 (LSTM)               (None, 150, 64)           33024     
                                                                 
 lstm_2 (LSTM)               (None, 150, 64)           33024     
                                                                 
 lstm_3 (LSTM)               (None, 64)                33024     
                                                                 
 dense (Dense)               (None, 256)               16640     
                                                                 
 dense_1 (Dense)             (None, 1)                 257       
                                                                 
=================================================================
Total params: 195,409
Trainable params: 195,409
Non-trainable params: 0
_________________________________________________________________
```

Fit on the training data

In [18]:

```
M =
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_split
=0.2)
```

```
Epoch 1/5
30/30 [==============================] - 39s 1s/step - loss: 0.3358 - accur
acy: 0.8691 - val_loss: 0.1724 - val_accuracy: 0.9536
Epoch 2/5
30/30 [==============================] - 29s 972ms/step - loss: 0.0913 - ac
curacy: 0.9736 - val_loss: 0.0774 - val_accuracy: 0.9768
Epoch 3/5
30/30 [==============================] - 32s 1s/step - loss: 0.0592 - accur
acy: 0.9842 - val_loss: 0.0669 - val_accuracy: 0.9831
Epoch 4/5
30/30 [==============================] - 29s 959ms/step - loss: 0.0458 - ac
curacy: 0.9865 - val_loss: 0.0678 - val_accuracy: 0.9810
Epoch 5/5
30/30 [==============================] - 29s 980ms/step - loss: 0.0378 - ac
curacy: 0.9889 - val_loss: 0.0700 - val_accuracy: 0.9810
```

Saving the model

In [19]:

```
model.save
```

Out[19]:

>

Evaluate the model on test set data

In [20]:

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = utils.pad_sequences(test_sequences,maxlen=max_len)
```

In [21]:

```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 4s 81ms/step - loss: 0.0649 - accu
racy: 0.9785
```

In [22]:

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

```
Test set
  Loss: 0.065
  Accuracy: 0.978
```

Accuracy and Loss Graph

In [23]:

```
results = pd.DataFrame({"Train Loss": M.history['loss'], "Validation Loss":
M.history['val_loss'],
              "Train Accuracy": M.history['accuracy'], "Validation
Accuracy": M.history['val_accuracy']
             })
fig, ax = plt.subplots(nrows=2, figsize=(16, 9))
results[["Train Loss", "Validation Loss"]].plot(ax=ax[0])
results[["Train Accuracy", "Validation Accuracy"]].plot(ax=ax[1])
ax[0].set_xlabel("Epoch")
ax[1].set_xlabel("Epoch")
plt.show()
```

In [ ]: