```
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 to_categorical
from keras.callbacks import EarlyStopping
from keras.models import load_model
%matplotlib inline
```

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

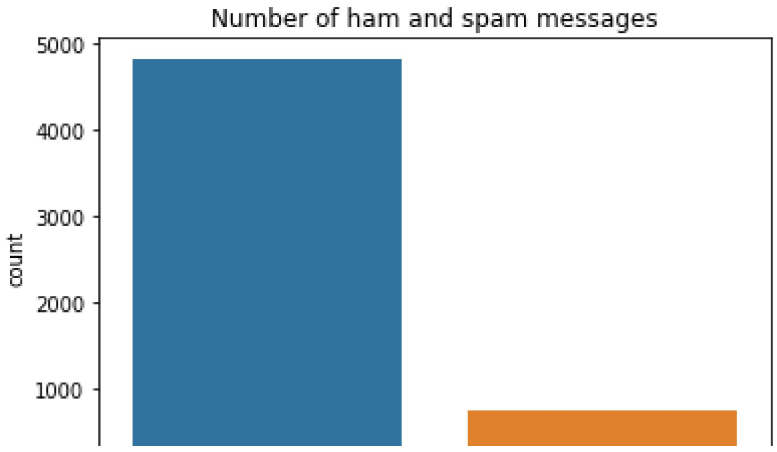
| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| 1 | ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| 3 | ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |

```
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
```

```
# data distribution
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: FutureWarning: Pas
  FutureWarning
Text(0.5, 1.0, 'Number of ham and spam messages')
```




```
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)


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)



#layers of the 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)
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0
```

```
 embedding_1 (Embedding)     (None, 150, 50)           50000

 lstm_1 (LSTM)               (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation_2 (Activation)   (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_3 (Activation)   (None, 1)                 0

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

```
model.fit(sequences_matrix,y_train,batch_size=128,epochs=10,
          validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.000
```

```
Epoch 1/10
30/30 [==============================] - 3s 40ms/step - loss: 0.3257 - accuracy: 0.8
Epoch 2/10
30/30 [==============================] - 0s 15ms/step - loss: 0.0776 - accuracy: 0.9
<keras.callbacks.History at 0x7fb2500eba10>
```

```
# saving a model
model.save("model.h5")
```

# Testing The model

```
test_sequences = tok.texts_to_sequences(x_test)
test_sequences_matrix  = sequence.pad_sequences(test_sequences,maxlen=max_len)
```

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

```
27/27 [==============================] - 0s 5ms/step - loss: 0.0392 - accuracy: 0.98
```

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

```
Test set
  Loss: 0.039
  Accuracy: 0.988
```

```
y_pred = model.predict(test_sequences_matrix)
print(y_pred[0:10])
```

```
27/27 [==============================] - 0s 4ms/step
[[1.9289477e-03]
 [7.8968434e-03]
 [7.8617368e-04]
 [1.5687625e-03]
 [1.0894409e-03]
 [9.9515355e-01]
 [1.0563295e-02]
 [6.3485336e-03]
 [2.2582253e-03]
 [3.2360202e-03]]
```

```
y_test[0:10][0][0]
```

```
0
```

```
labels = {0:'ham',1:'spam'}
for i in range(0,10):
  print(labels[y_test[10:20][i][0]]);
```

```
ham
ham
ham
ham
spam
spam
ham
ham
ham
ham
```

Colab paid products - Cancel contracts here

✓ 0s completed at 11:16 AM