Import the necessary libraries

In [45]:

```
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
%matplotlib inline
```

Load the data into Pandas dataframe

In [14]:

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

Out[14]:

	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

Drop the columns that are not required for the neural network.

In [15]:

```
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
```

Understand the distribution better.

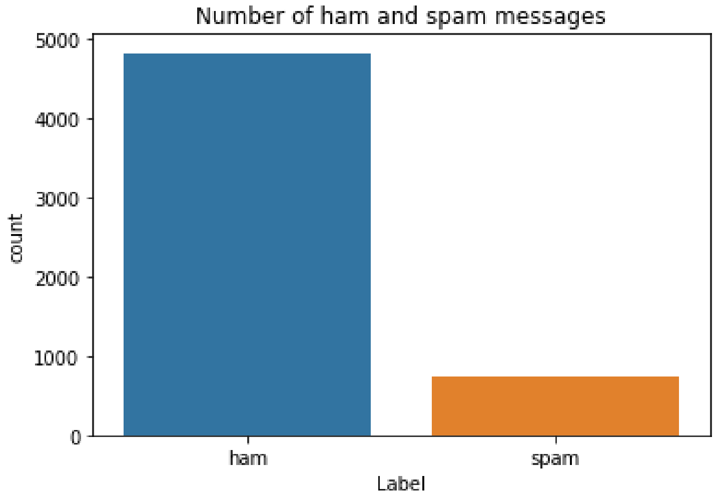
In [16]:

```
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: Pass the
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result
in an error or misinterpretation.
  FutureWarning
```

Out[16]:

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



Create input and output vectors.

Process the labels.

In [17]:

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

Split into training and test data.

In [18]:

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

Process the data

- **Tokenize the data and convert the text to sequences.**
- **Add padding to ensure that all the sequences have the same shape.**
- **There are many ways of taking the max_len and here an arbitrary length of 150 is chosen.**

In [69]:

```
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)
```

RNN

Define the RNN structure.

In [59]:

```
def RNN():
    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)
    return model
```

Call the function and compile the model.

In [70]:

```
model = RNN()
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
_________________________________________________________________
```

Fit on the training data.

In [60]:

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
          validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=0.0001)])
```

```
Epoch 1/10
30/30 [==============================] - 1s 17ms/step - loss: 0.0059 - accuracy: 0.9976 - val_loss: 0.1091 - val_accuracy: 0.9842
Epoch 2/10
30/30 [==============================] - 0s 13ms/step - loss: 0.0029 - accuracy: 0.9995 - val_loss: 0.1160 - val_accuracy: 0.9852
```

Out[60]:

```
<keras.callbacks.History at 0x7fea9c3548d0>
```

The model performs well on the validation set and this configuration is chosen as the final model.

Process the test set data.

In [61]:

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

Evaluate the model on the test set.

In [66]:

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

```
27/27 [==============================] - 0s 6ms/step - loss: 0.0996 - accuracy: 0.9880
```

In [71]:

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

```
Test set
 Loss: 0.100
 Accuracy: 0.988
```