# Assignment - 4

**LSTM for Text Classification**

# Download the Dataset

1. **Import required library**

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.utils import pad\_sequences

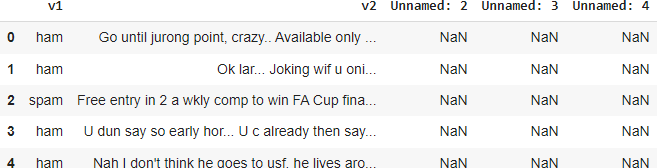
from keras.utils import to\_categorical from keras.callbacks import EarlyStopping

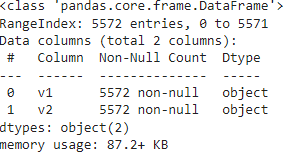
%matplotlib inline

# Read dataset and do pre-processing

Load the data into Pandas dataframe

df = pd.read\_csv('/content/spam.csv',delimiter=',',encoding='latin-1') df.head()

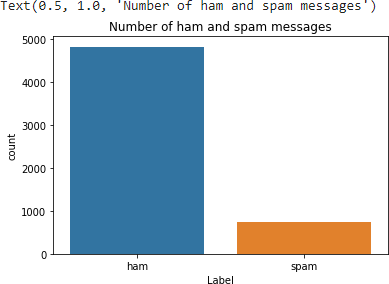


Drop the columns that are not required for the neural network. df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True) df.info()

Understand the distribution better.

sns.countplot(df.v1) plt.xlabel('Label')

plt.title('Number of ham and spam messages')



* Create input and output vectors.
* Process the labels.

X = df.v2 Y = df.v1

le = LabelEncoder()

Y = le.fit\_transform(Y) Y = Y.reshape(-1,1)

Split into training and test data.

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.

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)

# Create Model

* **Add Layers (LSTM, Dense-(Hidden Layers), Output)**

Define the RNN structure. 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)

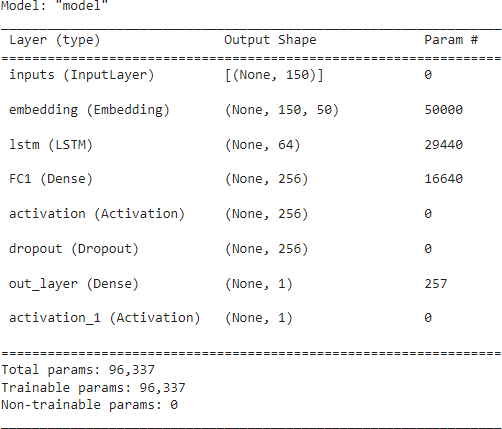
model = Model(inputs=inputs,outputs=layer) return model

Call the function and compile the model. model = RNN()

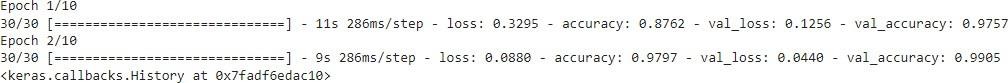
model.summary()

# Compile the Model

model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])



# Fit the Model

model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10, validation\_split=0.2,callbacks=[EarlyStopping(monitor='val\_loss',min\_delta=0.0001)])

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

# Save The Model

lstm\_model.save('text\_model.h5')

# Test The Model

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.

accr = model.evaluate(test\_sequences\_matrix,Y\_test)



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

