

## Assignment - 4 LSTM for Text Classification

Assignment submission	7 November 2022
Student Name	Uthra v
Student Roll Number	951919CS108
Maximum Marks	2 Marks

### 1. Download the Dataset 2. 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.np_utils import to_categorical
from keras.callbacks import EarlyStopping

%matplotlib inline
```

### 3. 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()
```

	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 does to usf. he lives aro...	NaN	NaN	NaN

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()
```

```

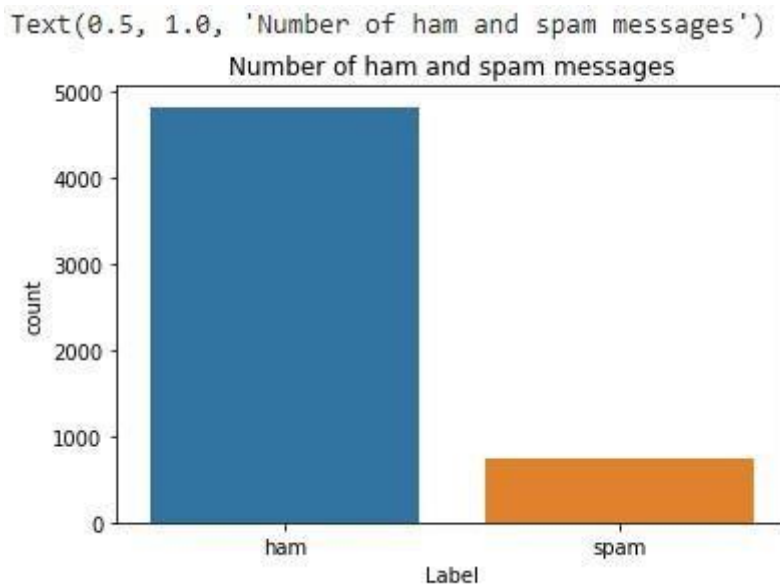
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  ---
0    v1      5572 non-null    object
1    v2      5572 non-null    object
dtypes: object(2)
memory usage: 87.2+ KB

```

Understand the distribution better.

```
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)
```

## 5. 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()
```

## 6. Compile the Model

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

Model: "model"

Layer (type)	Output Shape	Param #
inputs (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_1 (Activation)	(None, 1)	0

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

## 7. Fit the Model

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

```
Epoch 1/10  
30/30 [=====] - 11s 286ms/step - loss: 0.3295 - accuracy: 0.8762 - val_loss: 0.1256 - val_accuracy: 0.9757  
Epoch 2/10  
30/30 [=====] - 9s 286ms/step - loss: 0.0880 - accuracy: 0.9797 - val_loss: 0.0440 - val_accuracy: 0.9905  
<keras.callbacks.History at 0x7fadf6edac10>
```

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

## 8. Save The Model lstm\_model.save('text\_model.h5')

## 9. 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)
```

```
27/27 [=====] - 1s 23ms/step - loss: 0.0606 - accuracy: 0.9833
```

```
print("Test set\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(accur[0],accur[1]))
```

```
Test set
```

```
Loss: 0.061
```

```
Accuracy: 0.983
```