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 pad\_sequences

from keras.utils import to\_categorical

from keras.callbacks import EarlyStopping READING DATASET

df = pd.read\_csv('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 goes to usf, he lives aro... | NaN | NaN | NaN |

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

1. v1 5572 non-null object
2. v2 5572 non-null object dtypes: object(2)

memory usage: 87.2+ KB df.groupby(['v1']).size() v1

ham 4825

spam 747 dtype: int64

df.groupby(['v2']).size() v2

<#> in mca. But not conform. 1

<#> mins but i had to stop somewhere first. 1

<DECIMAL> m but its not a common car here so its better to buy from china or asia. Or if i find it less expensive. I.ll holla 1

and picking them up from various points 1

came to look at the flat, seems ok, in his 50s? \* Is away alot wiv work. Got woman coming at

6.30 too. 1

..

ÌÏ still got lessons? ÌÏ in sch? 1

ÌÏ takin linear algebra today? 1

ÌÏ thk of wat to eat tonight. 1

ÌÏ v ma fan... 1

ÌÏ wait 4 me in sch i finish ard 5.. 1

Length: 5169, dtype: int64 X = df.v2

Y = df.v1

le = LabelEncoder()

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

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 = pad\_sequences(sequences,maxlen=max\_len) CREATE MODEL AND ADD LAYERS

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) COMPILE AND FIT THE MODEL

model.summary()

model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy']) model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,

validation\_split=0.2) 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

Epoch 1/10

30/30 [==============================] - 11s 286ms/step - loss: 0.3204 - accuracy:

0.8820 - val\_loss: 0.1487 - val\_accuracy: 0.9726 Epoch 2/10

30/30 [==============================] - 8s 260ms/step - loss: 0.0889 - accuracy:

0.9791 - val\_loss: 0.0641 - val\_accuracy: 0.9831 Epoch 3/10

30/30 [==============================] - 8s 263ms/step - loss: 0.0482 - accuracy:

0.9863 - val\_loss: 0.0461 - val\_accuracy: 0.9895 Epoch 4/10

30/30 [==============================] - 8s 261ms/step - loss: 0.0361 - accuracy:

0.9894 - val\_loss: 0.0363 - val\_accuracy: 0.9895 Epoch 5/10

30/30 [==============================] - 8s 258ms/step - loss: 0.0312 - accuracy:

0.9897 - val\_loss: 0.0365 - val\_accuracy: 0.9895 Epoch 6/10

30/30 [==============================] - 10s 328ms/step - loss: 0.0223 - accuracy:

0.9923 - val\_loss: 0.0418 - val\_accuracy:

0.9863Epoch 7/10

30/30 [==============================] - 9s 284ms/step - loss: 0.0179 - accuracy:

0.9945 - val\_loss: 0.0473 - val\_accuracy:

0.9852Epoch 8/10

30/30 [==============================] - 10s 344ms/step - loss: 0.0123 - accuracy:

0.9950 - val\_loss: 0.0599 - val\_accuracy:

0.9895Epoch 9/10

30/30 [==============================] - 8s 271ms/step - loss: 0.0087 - accuracy:

0.9974 - val\_loss: 0.0592 - val\_accuracy:

0.9905Epoch 10/10

30/30 [==============================] - 8s 262ms/step - loss: 0.0082 - accuracy:

0.9971 - val\_loss: 0.0490 - val\_accuracy:

0.9884SAVING THE MODEL

model.save('sms\_classifie r.h5')TEST THE MODEL

test\_sequences = tok.texts\_to\_sequences(X\_test) test\_sequences\_matrix = pad\_sequences(test\_sequences,maxlen=max\_len)accr = model.evaluate(test\_sequences\_matrix,Y\_test)

27/27 [==============================] - 1s 24ms/step - loss: 0.0512 - accuracy:

0.9856

print('Test set\n Loss: {:0.3f}\n Accuracy:

{:0.3f}'.format(accr[0],accr[1]))Test set Loss: 0.051

Accuracy: 0.986

TEAM ID:PNT2022TMID24307