Assignment -4
LSTM for Text Classification

DATE	31 October 2022
TEAM ID	PNT2022TMID26356
PROJECT NAME	Natural Disasters Intensity Analysis and Classification

#Import necessary libraries import numpy as np import

pandas as pd import matplotlib.pyplot as plt import

seaborn as sns %matplotlib inline from

sklearn.model_selection import train_test_split

from keras.layers import Dense , LSTM , Embedding , Dropout , Activation , Flatten

from sklearn.preprocessing import LabelEncoder from keras.preprocessing.text

import Tokenizer from keras.models import Sequential from

tensorflow.keras.preprocessing import sequence from tensorflow.keras.utils import

to_categorical from keras.callbacks import EarlyStopping from

tensorflow.keras.optimizers import RMSprop from keras_preprocessing.sequence

import pad_sequences

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [2]:
```
from sklearn.model_selection import train_test_split
from keras.layers import Dense , LSTM , Embedding , Dropout , Activation , Flatten
from sklearn.preprocessing import LabelEncoder
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import RMSprop
from keras_preprocessing.sequence import pad_sequences
```

#Read dataset and do pre-processing

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

#Information about dataset

data.describe().T data.shape

#Check if there is any missing values data.isnull().sum()

data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)

#Visualize the dataset sns.countplot(data.v1)

#Preprocess using Label Encoding

X = data.v2 Y = data.v1 le = LabelEncoder()

Y = le.fit_transform(Y)

Y = Y.reshape(-1,1)

```
In [3]: data = pd.read_csv('/content/spam.csv',delimiter=',',encoding='latin-1')

In [4]: data
```

Out[4]:

	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
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will İ_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

5572 rows × 5 columns

In [5]:
```
data.describe().T
```

Out[5]:

	count	unique	top	freq
v1	5572	2	ham	4825
v2	5572	5169	Sorry, I'll call later	30
Unnamed: 2	50	43	bt not his girlfrnd... G o o d n i g h t . . .@"	3
Unnamed: 3	12	10	MK17 92H. 450Ppw 16"	2
Unnamed: 4	6	5	GNT:-)"	2

In [6]:
```
data.shape
```

Out[6]:
```
(5572, 5)
```

In [7]:
```
data.isnull().sum()
```

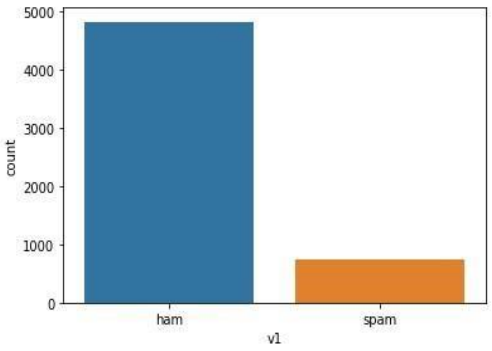
Out[7]:
```
v1              0
v2              0
Unnamed: 2    5522
Unnamed: 3    5560
Unnamed: 4    5566
dtype: int64
```

In [8]:
```
data.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
```

In [9]:
```
sns.countplot(data.v1)
```

```
/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 argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  FutureWarning
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1735223150>
```



In [10]:
```
X = data.v2
Y = data.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
```

In [11]:
```
Y = Y.reshape(-1,1)
```

#Create Model and Add Layers (LSTM, Dense-(Hidden Layers), Output) #Splitting into training and testing data

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

max_word = 1000 max_len = 250 token =

Tokenizer(num_words = max_word)

token.fit_on_texts(X_train) sequences =

token.texts_to_sequences(X_train) seq_matrix = sequence.pad_sequences(sequences

, maxlen = max_len)

#Creating the model model = Sequential()

model.add(Embedding(max_word , 32 , input_length = max_len))

model.add(LSTM(64))

model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dropout(0.5))

model.add(Dense(120, activation='relu')) model.add(Dense(1,

activation='sigmoid'))

```
In [12]: X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

In [13]: max_word = 1000
         max_len = 250
         token = Tokenizer(num_words = max_word)
         token.fit_on_texts(X_train)
         sequences = token.texts_to_sequences(X_train)
         seq_matrix = sequence.pad_sequences(sequences , maxlen = max_len)

In [28]: model = Sequential()
         model.add(Embedding(max_word , 32 , input_length = max_len))
         model.add(LSTM(64))
         model.add(Flatten())
         model.add(Dense(250, activation='relu'))
         model.add(Dropout(0.5))
         model.add(Dense(120, activation='relu'))
         model.add(Dense(1, activation='sigmoid'))
```

#compile the model

model.compile(loss = 'binary_crossentropy' , optimizer = 'RMSprop' , metrics = 'accuracy') model.summary()

```
In [15]: model.compile(loss = 'binary_crossentropy' , optimizer = 'RMSprop' , metrics = 'accuracy')
         model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 250, 32)           32000

 lstm (LSTM)                 (None, 64)                24832

 flatten (Flatten)           (None, 64)                0

 dense (Dense)               (None, 250)               16250

 dropout (Dropout)           (None, 250)               0

 dense_1 (Dense)             (None, 120)               30120

 dense_2 (Dense)             (None, 1)                 121

=================================================================
Total params: 103,323
Trainable params: 103,323
Non-trainable params: 0
_________________________________________________________________
```

#Fit the model

model.fit(seq_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlySt

opping(monitor='val_loss',min_delta=0.0001)]) test_seq = token.texts_to_sequences(X_test)

test_seq_matrix = sequence.pad_sequences(test_seq,maxlen=max_len)

```
In [18]: model.fit(seq_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta
```

```
Epoch 1/10
28/28 [==============================] - 14s 489ms/step - loss: 0.0256 - accuracy: 0.9910 - val_loss: 0.0566 - val_accuracy: 0.
9843
Epoch 2/10
28/28 [==============================] - 11s 385ms/step - loss: 0.0180 - accuracy: 0.9952 - val_loss: 0.0582 - val_accuracy: 0.
9843
```

```
Out[18]: <keras.callbacks.History at 0x7f172ca32b90>
```

```
In [19]: test_seq = token.texts_to_sequences(X_test)
         test_seq_matrix = sequence.pad_sequences(test_seq,maxlen=max_len)
```

#Save the model model.save(r'lstm_model.h5')

```
In [24]: model.save(r'lstm_model.h5')
```

#Test the model: from tensorflow.keras.models import

load_model new_model=load_model(r'lstm_model.h5')

new_model.evaluate(test_seq_matrix,Y_test) scores =

model.evaluate(test_seq_matrix, Y_test, verbose=0) scores

print("Accuracy: %.2f%%" % (scores[1]*100))

```
In [25]: from tensorflow.keras.models import load_model
         new_model=load_model(r'lstm_model.h5')

In [27]: new_model.evaluate(test_seq_matrix,Y_test)
         35/35 [==============================] - 2s 36ms/step - loss: 0.0655 - accuracy: 0.9821
Out[27]: [0.06549865007400513, 0.9820627570152283]
```

```
In [20]: scores = model.evaluate(test_seq_matrix, Y_test, verbose=0)
         scores
Out[20]: [0.06549865007400513, 0.9820627570152283]

In [21]: print("Accuracy: %.2f%%" % (scores[1]*100))
         Accuracy: 98.21%
```