Sentiment Analysis in Text with LSTM

The implementation has been done in3phases as follow :

Preparing and Adjusting Dataset :

```
import os

os . chdir ( " aclImdb " )
os . chdir ( "train" )
os . chdir ( " neg " )
print ( os . getcwd ())
directorylist = os . listdir ()
#print( directorylist )
datalist = []
for i in directory list :
    file = open ( i )
    data = file . readlines ()
    file . close ()
    my_str = data [ 0 ]. lower ()
    punctuations = '' '!( )-[] {} ;:'"\,<>./?@#$%^&*_~'''

    no_punct = ""
    for char in my_str :
        if char not in Punctuation :
            no_punct = no_punct + char

    datalist . append ( no_punct )

print ( " \n " , datalist [ 3 ])

wordseperatedataset = []
for i in datalist :
    words = i.split ( " " )
    wordsseparateddataset . append ( words )

print ( " \n " , wordsseparateddataset [ 3 ])


import csv

os . chdir ( "../" )
os . chdir ( "../" )
os . chdir ( "../" )
file = open ( "train_neg.csv" , "w" , newline = "" )
```

```
writer = csv . writer ( file )
writer _ writerows ( wordsseparateddataset )

file . close ()
```

The code above is an example for uniting and creating a list from the data.

In the codes to set the data (sentences), first we remove all of them into lowercase letters and then remove all the symbols from it, then we separate each sentence with the help of split based on the space character, and now the sentence becomes a listof words. will be

This has been done ۵ times because there are5 folders in the aclImdb dataset folder as :follows

1. Train:
   1.1 neg
   1.2 pos
   1.3 unsup
2. tests:
   2.1 neg
   2.2 pos

There are codes for doing this part in the folder " Units and Bases of the List."

The lists created by these codes are saved as a csv file and placed inside the folder " compiled and unified data

csv files wereuploaded ingoogle drive and used in google colab, which call these files asfollows :

```python
# فراخوانی داده‌های آموزش
from google.colab import drive
drive.mount('/content/drive')

train_file_neg= open('/content/drive/MyDrive/HW4_dataset/train_neg.csv', 'r')
train_list_neg = train_file_neg.readlines()
train_file_neg.close()

train_file_pos= open('/content/drive/MyDrive/HW4_dataset/train_pos.csv', 'r')
train_list_pos = train_file_pos.readlines()
train_file_pos.close()

train_file_unsop= open('/content/drive/MyDrive/HW4_dataset/train_unsop.csv', 'r')
train_list_unsop = train_file_unsop.readlines()
train_file_unsop.close()

# فراخوانی داده‌های آزمایش
test_file_neg= open('/content/drive/MyDrive/HW4_dataset/test_neg.csv', 'r')
test_list_neg = test_file_neg.readlines()
test_file_neg.close()

test_file_pos= open('/content/drive/MyDrive/HW4_dataset/test_pos.csv', 'r')
test_list_pos = test_file_pos.readlines()
test_file_pos.close()
```

```
Mounted at /content/drive
```

BERT:

To write the word to vector conversion function, the following libraries have been imported .

```
[4] import tensorflow as tf
    import tensorflow_text as text
    import tensorflow_hub as hub
    print(tf.__version__)

2.11.0
```

```
[5] bert_preprocessor = hub.KerasLayer(
        "https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3")

WARNING:tensorflow:Please fix your imports. Module tensorflow.python.t
```

```
[14] bert_encoder = hub.KerasLayer(
        "https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4")
```

The word-to-vector conversion function is done usingpreprocessor andencoder , BERT :which is as follows ,

```
def get_sentence_embeding(senteces):
  preprocessed_text = bert_preprocessor(senteces)
  return bert_encoder(preprocessed_text)['pooled_output']

get_sentence_embeding(["500$ discount. hurry up",
                       "bhavin, are you up for a game tomorrow?"
                       ])

<tf.Tensor: shape=(2, 768), dtype=float32, numpy=
array([[-0.8435169 , -0.51327264, -0.88845724, ..., -0.74748874,
        -0.75314736,  0.91964483],
       [-0.8765555 , -0.50088423, -0.95878726, ..., -0.8938003 ,
        -0.71859884,  0.8784346 ]], dtype=float32)>
```

An example to see the performance of this algorithm along with the output can .be seen in the image above

On one of the examples (comments) of the problem dataset, this function has been :examined and its code and output are as follows

```
print("test_list_pos[1]: ",test_list_pos[1])
a = test_list_pos[1].split(',')
b = get_sentence_embeding(a)
print("bert_vec: ",b)
```

```
test_list_pos[1]:  actor,turned,director,bill,paxton,follows,up,his,

bert_vec:  tf.Tensor(
[[-0.9374246  -0.6122873  -0.82031155 ... -0.52665734 -0.6588041
   0.8915311 ]
 [-0.87391037 -0.14990044  0.3370829  ...  0.25290614 -0.5821726
   0.8747224 ]
 [-0.9359918  -0.4552164  -0.6063657  ... -0.30570996 -0.6558526
   0.87928855]
 ...
 [-0.87888384 -0.20548141  0.47031605 ...  0.4175205  -0.5982299
   0.89078474]
 [-0.8039575  -0.16494808  0.48876008 ...  0.28318304 -0.5206983
   0.83352554]
 [-0.8164539  -0.21256424  0.38060272 ...  0.35062334 -0.5337261
   0.8098593 ]], shape=(344, 768), dtype=float32)
```

LSTM :

we call the necessary libraries to implement ,LSTM :

```
[ ]  from tensorflow import keras
     from keras.layers import Embedding, Dense, LSTM
     from keras.models import Sequential, load_model
     from keras.losses import BinaryCrossentropy
     from keras.optimizers import Adam
     from keras.preprocessing import sequence
```

```
[ ]  from keras.datasets import imdb
     from keras.utils import pad_sequences
```

To improve the speedof training, :the following commands are used

```
# speed up
import os
os.environ['CUDA_VISIBLE_DEVICES'] = '-1'
# Disable eager execution
tf.compat.v1.disable_eager_execution()
```

For the ease of working withgoogle colab and not depending on the data ingoogle drive , the ready IMDB dataset available in thetensorflow library .has been used

```
# Load dataset
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=num_distinct_words)
print(x_train.shape)
print(x_test.shape)

# Pad all sequences
padded_inputs = pad_sequences(x_train, maxlen=max_sequence_length, value = 0.0) # 0.0 because it corresponds with <PAD>
padded_inputs_test = pad_sequences(x_test, maxlen=max_sequence_length, value = 0.0) # 0.0 because it corresponds with <PAD>
```

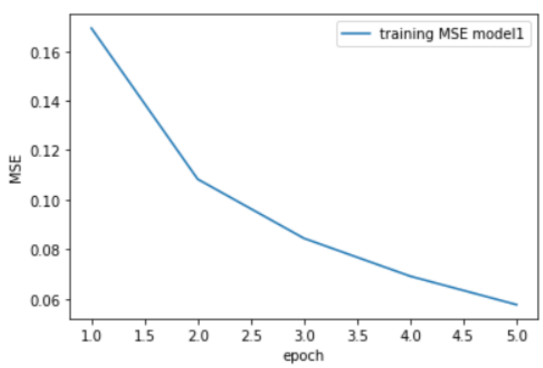
Now it's time toconfigure .the model

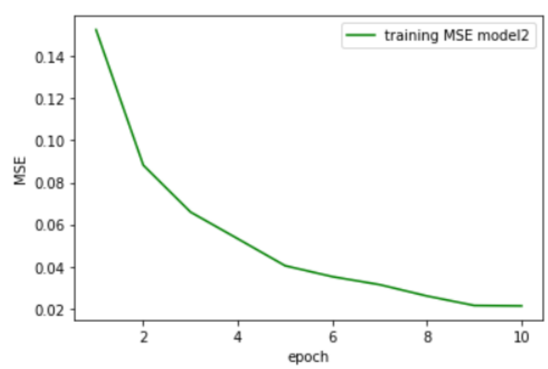
```
# Model configuration
additional_metrics = ['accuracy', 'mean_squared_error']
batch_size = 50
embedding_output_dims = 64
loss_function = BinaryCrossentropy()
max_sequence_length = 300
num_distinct_words = 5000
number_of_epochs = 5
optimizer = Adam()
validation_split = 0.20
verbosity_mode = 1
# Define the Keras model
model = Sequential()
model.add(Embedding(num_distinct_words, embedding_output_dims, input_length=max_sequence_length))
model.add(LSTM(10))
model.add(Dense(1, activation='sigmoid'))
# Compile the model
model.compile(optimizer=optimizer, loss=loss_function, metrics=additional_metrics)
```

:The two basic and simple models namedmodel1 andmodel2 " in theinit_model folder " .contain the code, outputs and results of these models

The only difference between these two models is the number of theirepochs , the first model has epoch=5 and the second model hasepoch=10 .

According tothe accuracy and MSE changes chart with increasingepoch , the trend of accuracy is increasing and the trendof MSE (as you can see in the chart below) is decreasing, butaccuracy in the test data decreased with increasing epoch . It seems that increasingthe epoch ,from 5 to 10 leads to overfitas a result, the accuracyof lstm .on the test samples has decreased





Accuracy " result in model ١ and ٢ is ininit_model folder , this result for "model1 with) 5epochs (Accuracy: 85.291999578847595% And For model2 with ١٠)epochs (Accuracy: 80.03600239753723%_ is

Bidirectional LSTM

To bidirectionalizeLSTM, it is enough to call theBidirectional library as follows

```
from keras.layers Import Bidirectional
```

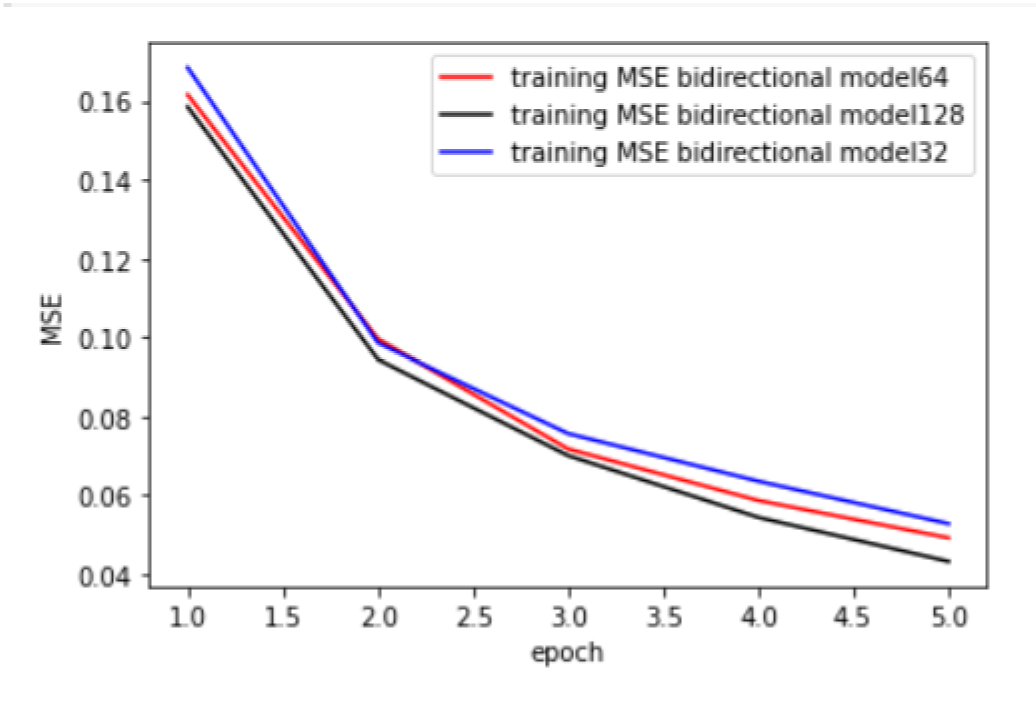
.And then the only change we need to make is to convert line(1) to line)2(

```
model.add ( LSTM( 10 )) :line)1(

model.add (Bidirectional( LSTM( 10 ))) :line)2(
```

) We implemented this algorithmBidirectional LSTM) with three different numbers of memory units of this algorithm and all parts of these 3 codes and their results and implementations " are located in thebidirectional_model .folder "

A comparison on theMSE chart .of these three cases can be seen in the figure below



MSE decreases faster, and as a result, the accuracy of the training process increases faster.