Assignment-4

SMSSPAMCLASSIFICATION

|  |  |
| --- | --- |
| AssignmentDate | 21October2022 |
| StudentsName | TeamLeader:S.Thanigaivel  TeamMember1:R.Alageshwari  TeamMember2:T.Anbumozhi  TeamMember3:R.Pavithra |
| TeamID | PNT2022TMID48053 |
| MaximumMarks | 2Marks |

# ImporttheDataset

fromgoogle.colabimportfiles uploaded=files.upload() Savingspam.csvtospam.csv Importrequiredlibraries

importcsv importtensorflowastf importpandasaspd importnumpyasnp

importmatplotlib.pyplotasplt

fromtensorflow.keras.preprocessing.textimportTokenizer fromtensorflow.keras.preprocessing.sequenceimportpad\_sequences importnltk nltk.download('stopwords') fromnltk.corpusimportstopwords

STOPWORDS=set(stopwords.words('english'))

[nltk\_data]Downloadingpackagestopwordsto/root/nltk\_data... [nltk\_data]Unzippingcorpora/stopwords.zip.

[nltk\_data]Unzippingcorpora/stopwords.zip. Importdataset

importio

dataset=pd.read\_csv(io.BytesIO(uploaded['spam.csv']),encoding="ISO-8859-1") dataset

v1 v2Unnamed:2\

1. hamGountiljurongpoint,crazy..Availableonly... NaN
2. ham Oklar...Jokingwifuoni... NaN
3. spamFreeentryin2awklycomptowinFACupfina... NaN
4. hamUdunsaysoearlyhor...Ucalreadythensay... NaN
5. hamNahIdon'tthinkhegoestousf,helivesaro... NaN

...... ... ... 5567spamThisisthe2ndtimewehavetried2contactu... NaN

5568ham WillÌ\_bgoingtoesplanadefrhome? NaN

5569hamPity,\*wasinmoodforthat.So...anyothers... NaN

5570hamTheguydidsomebitchingbutIactedlikei'd... NaN

5571ham Rofl.Itstruetoitsname NaN

Unnamed:3Unnamed:4

1. NaN NaN
2. NaN NaN
3. NaN NaN
4. NaN NaN
5. NaN NaN

... ... ...

1. NaN NaN
2. NaN NaN
3. NaN NaN
4. NaN NaN
5. NaN NaN

[5572rowsx5columns]

vocab\_size=5000 embedding\_dim=64 max\_length=200 trunc\_type='post' padding\_type='post' oov\_tok='' training\_portion=.8

Readthedatasetanddopre-processing. Toremovethestopwords.

articles=[] labels=[]

withopen("spam.csv",'r',encoding="ISO-8859-1")asdataset: reader=csv.reader(dataset,delimiter=',') next(reader) forrowinreader:

labels.append(row[0]) article=row[1] forwordinSTOPWORDS: token=''+word+''

article=article.replace(token,'') article=article.replace('','')

articles.append(article)

print(len(labels)) print(len(articles))

5572

5572

# Trainthemodel

train\_size=int(len(articles)\*training\_portion) train\_articles=articles[0:train\_size]

train\_labels=labels[0:train\_size] validation\_articles=articles[train\_size:] validation\_labels=labels[train\_size:] print(train\_size) print(len(train\_articles)) print(len(train\_labels)) print(len(validation\_articles)) print(len(validation\_labels))

4457

4457

4457

1115

1115

tokenizer=Tokenizer(num\_words=vocab\_size,oov\_token=oov\_tok) tokenizer.fit\_on\_texts(train\_articles) word\_index=tokenizer.word\_index dict(list(word\_index.items())[0:10])

{'':1,

'i':2,

'u':3,

'call':4,

'you':5,

'2':6,

'get':7,

"i'm":8,

'ur':9,

'now':10}

# TrainingdatatoSequences

train\_sequences=tokenizer.texts\_to\_sequences(train\_articles) print(train\_sequences[10])

[8,190,37,201,30,260,293,991,222,53,153,3815,423,46] TrainneuralnetworkforNLP

train\_padded=pad\_sequences(train\_sequences,maxlen=max\_length,padding=padding\_type, truncating=trunc\_type) print(len(train\_sequences[0])) print(len(train\_padded[0]))

print(len(train\_sequences[1])) print(len(train\_padded[1]))

print(len(train\_sequences[10])) print(len(train\_padded[10]))

16

200

6

200

14

200

print(train\_padded[10])

[8190372013026029399122253153381542346

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000

00000000000000 0000]

validation\_sequences=tokenizer.texts\_to\_sequences(validation\_articles)

validation\_padded = pad\_sequences(validation\_sequences, maxlen=max\_length, padding=padding\_type,truncating=trunc\_type)

print(len(validation\_sequences)) print(validation\_padded.shape)

1115

(1115,200)

label\_tokenizer=Tokenizer() label\_tokenizer.fit\_on\_texts(labels)

training\_label\_seq=np.array(label\_tokenizer.texts\_to\_sequences(train\_labels)) validation\_label\_seq=np.array(label\_tokenizer.texts\_to\_sequences(validation\_labels)) print(training\_label\_seq[0]) print(training\_label\_seq[1]) print(training\_label\_seq[2]) print(training\_label\_seq.shape)

print(validation\_label\_seq[0]) print(validation\_label\_seq[1]) print(validation\_label\_seq[2]) print(validation\_label\_seq.shape)

[1]

[1]

[2]

(4457,1)

[1]

[2]

[1]

(1115,1) reverse\_word\_index=dict([(value,key)for(key,value)inword\_index.items()])

defdecode\_article(text): return''.join([reverse\_word\_index.get(i,'?')foriintext]) print(decode\_article(train\_padded[10]))

print('---') print(train\_articles[10]) i'mgonnahomesoonwanttalkstuffanymoretonightki'vecriedenoughtoday???????????

????????????????????????????????????????????????????????????

???????????????????????????????????????????????????????????? ???????????????????????????????????????????????????????

---

I'mgonnahomesoonwanttalkstuffanymoretonight,k?I'vecriedenoughtoday. ToimplementLSTM

model=tf.keras.Sequential([

tf.keras.layers.Embedding(vocab\_size,embedding\_dim), tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(embedding\_dim)), tf.keras.layers.Dense(embedding\_dim,activation='relu'), tf.keras.layers.Dense(6,activation='softmax')

]) model.summary()

Model:"sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer(type) OutputShape Param#

================================================================= embedding(Embedding) (None,None,64) 320000

bidirectional(Bidirectional(None,128) 66048 l)

dense(Dense) (None,64) 8256 dense\_1(Dense) (None,6) 390

=================================================================

Totalparams:394,694

Trainableparams:394,694

Non-trainableparams:0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ print(set(labels)) {'spam','ham'}

model.compile(loss='sparse\_categorical\_crossentropy',optimizer='adam',metrics=['accuracy']) num\_epochs=10

history = model.fit(train\_padded, training\_label\_seq, epochs=num\_epochs,

validation\_data=(validation\_padded,validation\_label\_seq),verbose=2)

Epoch1/10

140/140-37s-loss:0.3177-accuracy:0.9251-val\_loss:0.0387-val\_accuracy:0.9830-

37s/epoch-265ms/step

Epoch2/10

140/140-35s-loss:0.0310-accuracy:0.9915-val\_loss:0.0318-val\_accuracy:0.9901-

35s/epoch-252ms/step

Epoch3/10

140/140-32s-loss:0.0130-accuracy:0.9975-val\_loss:0.0627-val\_accuracy:0.985732s/epoch-230ms/step

Epoch4/10

140/140-31s-loss:0.0060-accuracy:0.9987-val\_loss:0.0478-val\_accuracy:0.9901-

31s/epoch-220ms/step

Epoch5/10

140/140-30s-loss:0.0042-accuracy:0.9989-val\_loss:0.0613-val\_accuracy:0.9883-

30s/epoch-215ms/step

Epoch6/10

140/140-29s-loss:0.0033-accuracy:0.9991-val\_loss:0.0728-val\_accuracy:0.9883-

29s/epoch-210ms/step

Epoch7/10

140/140-29s-loss:0.0020-accuracy:0.9996-val\_loss:0.0540-val\_accuracy:0.9865-

29s/epoch-208ms/step

Epoch8/10

140/140-31s-loss:7.6466e-04-accuracy:0.9998-val\_loss:0.0644-val\_accuracy:0.990131s/epoch-219ms/step

Epoch9/10

140/140-30s-loss:3.9159e-04-accuracy:1.0000-val\_loss:0.0678-val\_accuracy:0.988330s/epoch-211ms/step

Epoch10/10

140/140-29s-loss:1.7514e-04-accuracy:1.0000-val\_loss:0.0726-val\_accuracy:0.9883-

29s/epoch-208ms/step

defplot\_graphs(history,string): plt.plot(history.history[string]) plt.plot(history.history['val\_'+string]) plt.xlabel("Epochs") plt.ylabel(string)

plt.legend([string,'val\_'+string]) plt.show()

plot\_graphs

(

history

,

"

accuracy

"

)

plot\_graphs

(

history

,

"

loss

"

)

