Assignment-4

SMSSPAMCLASSIFICATION

AssignmentDate	07November2022
Teamleader	B.Babu
Teammembers	T.Siva,R.Imran,V.Jeeva
TeamID	PNT2022TMID41556
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\
0    hamGountiljurongpoint,crazy..Availableonly... NaN
1    ham           Oklar...Jokingwifuoni... NaN
2spamFreeentryin2awklycomptowinFACupfina... NaN
3    hamUdunsaysoearlyhor...Ucalreadythensay... NaN
4    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
0 NaN NaN 1 NaN NaN
2 NaN NaN 3 NaN NaN
4 NaN NaN
... ... ...
5567 NaN NaN 5568 NaN
NaN 5569 NaN NaN 5570
NaN NaN
5571 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_portio
n) 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_label
s))
validation_label_seq=np.array(label_tokenizer.texts_to_sequences(validatio
n_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'mgonnahomesoonwa
nttalkstuffanymoreton
ight,k?I'vecriedenough
today.
```

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_padd
ed, 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:0.9857-32s/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:0.9901-31s/epoch-219ms/step Epoch9/10

140/140-30s-loss:3.9159e-04-accuracy:1.0000-val_loss:0.0678-val_accuracy:0.9883-30s/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")
```