Assignment_4

November 2, 2022

[1]:
```
import pandas as pd
import numpy as np
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
from sklearn.preprocessing import LabelEncoder
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models import Word2Vec
from sklearn.model_selection import train_test_split
import gensim
from keras.preprocessing.text import Tokenizer
from keras_preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Activation, Dense, Dropout, Embedding, Flatten,␣
 ↪Conv1D, MaxPooling1D, LSTM
from keras.callbacks import ReduceLROnPlateau, EarlyStopping
from tensorflow.keras.models import load_model
import matplotlib.pyplot as plt
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
```

[1]:
```
True
```

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

[3]:

	v1	v2	Unnamed: 2	\
0	ham	Go until jurong point, crazy.. Available only …	NaN	
1	ham	Ok lar… Joking wif u oni…	NaN	
2	spam	Free entry in 2 a wkly comp to win FA Cup fina…	NaN	
3	ham	U dun say so early hor… U c already then say…	NaN	
4	ham	Nah I don't think he goes to usf, he lives aro…	NaN	

	Unnamed: 3	Unnamed: 4

```
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

[4]:
```
#Dropping NaN values
df=df.iloc[:,[0,1]]
```

[5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

[7]:
```
X = df['v2']
Y = df['v1']
X,Y
```

[7]:
```
(0       Go until jurong point, crazy.. Available only …
 1                           Ok lar… Joking wif u oni…
 2       Free entry in 2 a wkly comp to win FA Cup fina…
 3       U dun say so early hor… U c already then say…
 4       Nah I don't think he goes to usf, he lives aro…
                              …
 5567    This is the 2nd time we have tried 2 contact u…
 5568                 Will Ì_ b going to esplanade fr home?
 5569    Pity, * was in mood for that. So…any other s…
 5570    The guy did some bitching but I acted like i'd…
 5571                           Rofl. Its true to its name
 Name: v2, Length: 5572, dtype: object, 0        ham
 1        ham
 2       spam
 3        ham
 4        ham
         …
 5567    spam
 5568     ham
 5569     ham
 5570     ham
 5571     ham
 Name: v1, Length: 5572, dtype: object)
```

[8]:
```
#Data Pre-processing
#Encoding the labels
le=LabelEncoder()
Y=le.fit_transform(Y)
Y=Y.reshape(-1,1)
Y
```

[8]:
```
array([[0],
       [0],
       [1],
       …,
       [0],
       [0],
       [0]])
```

[9]:
```
#Stemming the text
port_stem=PorterStemmer()
corpus=[]
for i in range(len(df['v2'])):
    text_1=re.sub('[^a-zA-Z]'," ",df['v2'][i])
    text_1=text_1.lower()
    text_1=text_1.split()
    text_1=[port_stem.stem(word) for word in text_1 if word not in stopwords.
 ↪words('english')]
    text_1=' '.join(text_1)
    corpus.append(text_1)
```

[10]:
```
len(corpus), len(Y)
```

[10]:
```
(5572, 5572)
```

[11]:
```
#Splitting data into train and test data
xtrain,xval,ytrain,yval=train_test_split(corpus,Y,test_size=0.2,random_state=42)
```

[12]:
```
documents=[text.split() for text in xtrain]
len(documents)
```

[12]:
```
4457
```

[13]:
```
#Vectorization the words
w2v_model = gensim.models.Word2Vec(size=300,
                                   window=3,
                                   min_count=5,
                                   workers=8)
```

[14]:
```
# Build vocabulary from a dictionary of word frequencies
w2v_model.build_vocab(documents)
```

[15]:
```
words = w2v_model.wv.vocab.keys()
vocab_size = len(words)
print("Vocabulary size", vocab_size)
```

```
Vocabulary size 1335
```

[16]:
```
w2v_model.train(documents,total_examples=len(documents),epochs=32)
```

[16]:
```
(925681, 1279744)
```

[17]:
```
w2v_model.most_similar("answer")
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1:
DeprecationWarning: Call to deprecated `most_similar` (Method will be removed in
4.0.0, use self.wv.most_similar() instead).
  """Entry point for launching an IPython kernel.
```

[17]:
```
[('team', 0.9252808094024658),
 ('burn', 0.9181710481643677),
 ('result', 0.9167132377624512),
 ('avail', 0.9130380153656006),
 ('save', 0.9104349613189697),
 ('recent', 0.9102259874343872),
 ('begin', 0.9045358300209045),
 ('callertun', 0.9027060270309448),
 ('ti', 0.9010258913040161),
 ('user', 0.8993039131164551)]
```

[18]:
```
#Tokenizing the words
tokenizer=Tokenizer()
tokenizer.fit_on_texts(xtrain)
vocab_size = len(tokenizer.word_index) + 1
print("Total words", vocab_size)
```

```
Total words 5585
```

[19]:
```
x_train = pad_sequences(tokenizer.texts_to_sequences(xtrain), maxlen=300)
x_test = pad_sequences(tokenizer.texts_to_sequences(xval), maxlen=300)
```

[20]:
```
x_train
```

[20]:
```
array([[   0,    0,    0, …,  230,  263, 1584],
       [   0,    0,    0, …, 1585, 1995,  154],
       [   0,    0,    0, …,  264, 2848, 2849],
       …,
       [   0,    0,    0, …,  534,  158, 1638],
       [   0,    0,    0, …,  650,  939,  184],
       [   0,    0,    0, …,   66,    3,   55]], dtype=int32)
```

[21]:
```
#Creating matrix for Embedding layer in model
embedding_matrix = np.zeros((vocab_size, 300))
print(embedding_matrix)
for word, i in tokenizer.word_index.items():
    if word in w2v_model.wv:
        embedding_matrix[i] = w2v_model.wv[word]
print(embedding_matrix.shape)
```

```
[[0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 …
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]
 [0. 0. 0. … 0. 0. 0.]]
(5585, 300)
```

[22]:
```
#Building Model
model = Sequential()
model.add(Embedding(vocab_size, 300, weights=[embedding_matrix],␣
 ↪input_length=300, trainable=False))
model.add(Dropout(0.2))
model.add(LSTM(64, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

[23]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 300, 300)          1675500

 dropout (Dropout)           (None, 300, 300)          0

 lstm (LSTM)                 (None, 64)                93440

 dense (Dense)               (None, 1)                 65

=================================================================
Total params: 1,769,005
Trainable params: 93,505
Non-trainable params: 1,675,500
_________________________________________________________________
```

[24]:
```
model.compile(loss='binary_crossentropy',
              optimizer="adam",
              metrics=['accuracy'])
```

[25]:
```
#Training the model with certain callbacks settings
#Fitting the model
callbacks = [ ReduceLROnPlateau(monitor='val_loss', patience=5, cooldown=0),
              EarlyStopping(monitor='val_accuracy', min_delta=1e-4, patience=5)]
info = model.fit(x_train, ytrain,
                    batch_size=32,
                    epochs=10,
                    validation_split=0.1,
                    verbose=1,
                    callbacks=callbacks)
```

```
Epoch 1/10
126/126 [==============================] - 105s 805ms/step - loss: 0.1060 -
accuracy: 0.9771 - val_loss: 0.1118 - val_accuracy: 0.9686 - lr: 0.0010
Epoch 2/10
126/126 [==============================] - 93s 736ms/step - loss: 0.0561 -
accuracy: 0.9833 - val_loss: 0.1211 - val_accuracy: 0.9709 - lr: 0.0010
Epoch 3/10
126/126 [==============================] - 90s 711ms/step - loss: 0.0511 -
accuracy: 0.9855 - val_loss: 0.1078 - val_accuracy: 0.9686 - lr: 0.0010
Epoch 4/10
126/126 [==============================] - 95s 752ms/step - loss: 0.0474 -
accuracy: 0.9860 - val_loss: 0.1288 - val_accuracy: 0.9686 - lr: 0.0010
Epoch 5/10
126/126 [==============================] - 91s 723ms/step - loss: 0.0451 -
accuracy: 0.9878 - val_loss: 0.1116 - val_accuracy: 0.9664 - lr: 0.0010
Epoch 6/10
126/126 [==============================] - 91s 719ms/step - loss: 0.0428 -
accuracy: 0.9875 - val_loss: 0.1081 - val_accuracy: 0.9664 - lr: 0.0010
Epoch 7/10
126/126 [==============================] - 93s 740ms/step - loss: 0.0408 -
accuracy: 0.9885 - val_loss: 0.1145 - val_accuracy: 0.9686 - lr: 0.0010
```

[28]:
```
score = model.evaluate(x_test,yval,batch_size=32)
print("Accuracy:{1} Loss:{0}".format(score[0],score[1]))
```

```
35/35 [==============================] - 3s 96ms/step - loss: 0.0609 - accuracy:
0.9830
Accuracy:0.9829596281051636 Loss:0.060920845717191696
```

[29]:
```
#Saving the model
model.save('lstm_spam_classifier.h5')
```

[30]:
```
#Testing the model
model1 = load_model('/content/lstm_spam_classifier.h5')
model1.evaluate(x_test,yval)
```

```
35/35 [==============================] - 4s 98ms/step - loss: 0.0609 - accuracy:
0.9830
```

[30]: [0.060920845717191696, 0.9829596281051636]