2306 AML128
BHASKARBABU ALAM

Importing Data

```
import pandas as pd
import os

ds = pd.read_json('Sarcasm_Headlines_Dataset.json', lines = True)
```

Viewing the data

```
ds
```

```
                                                article_link  \
0      https://www.huffingtonpost.com/entry/versace-b...
1      https://www.huffingtonpost.com/entry/roseanne-...
2      https://local.theonion.com/mom-starting-to-fea...
3      https://politics.theonion.com/boehner-just-wan...
4      https://www.huffingtonpost.com/entry/jk-rowlin...
...                                                  ...
26704  https://www.huffingtonpost.com/entry/american-...
26705  https://www.huffingtonpost.com/entry/americas-...
26706  https://www.huffingtonpost.com/entry/reparatio...
26707  https://www.huffingtonpost.com/entry/israeli-b...
26708  https://www.huffingtonpost.com/entry/gourmet-g...

                                                headline  is_sarcastic
0      former versace store clerk sues over secret 'b...             0
1      the 'roseanne' revival catches up to our thorn...             0
2      mom starting to fear son's web series closest ...             1
3      boehner just wants wife to listen, not come up...             1
4      j.k. rowling wishes snape happy birthday in th...             0
...                                                  ...           ...
26704               american politics in moral free-fall             0
26705                            america's best 20 hikes             0
26706                              reparations and obama             0
26707  israeli ban targeting boycott supporters raise...             0
26708                  gourmet gifts for the foodie 2014             0

[26709 rows x 3 columns]
```

Reading the first few headlines to understand the mix

```
ds['headline'][0 : 20]
```

```
0     former versace store clerk sues over secret 'b...
1     the 'roseanne' revival catches up to our thorn...
2     mom starting to fear son's web series closest ...
3     boehner just wants wife to listen, not come up...
4     j.k. rowling wishes snape happy birthday in th...
5                           advancing the world's women
6        the fascinating case for eating lab-grown meat
7     this ceo will send your kids to school, if you...
8     top snake handler leaves sinking huckabee camp...
9     friday's morning email: inside trump's presser...
10    airline passengers tackle man who rushes cockp...
11    facebook reportedly working on healthcare feat...
12    north korea praises trump and urges us voters ...
13    actually, cnn's jeffrey lord has been 'indefen...
14    barcelona holds huge protest in support of ref...
15    nuclear bomb detonates during rehearsal for 's...
16    cosby lawyer asks why accusers didn't come for...
17    stock analysts confused, frightened by boar ma...
18    bloomberg's program to build better cities jus...
19                                 craig hicks indicted
Name: headline, dtype: object
```

Cleaning the strings in headlines to remove special characters, numbers etc

```
import re
from nltk.corpus import stopwords
import nltk
import string
nltk.download('stopwords')
stopwords = set(stopwords.words('english'))
def clean(text):
  text = re.sub(r'\d+', '', text)
  text = "".join([char for char in text if char not in
string.punctuation])
  return text

ds['headline'] = ds['headline'].apply(clean)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/sravya/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Checking the data after cleaning

```
ds['headline'][0 : 20]
```

```
0     former versace store clerk sues over secret bl...
1     the roseanne revival catches up to our thorny ...
2     mom starting to fear sons web series closest t...
3     boehner just wants wife to listen not come up ...
4     jk rowling wishes snape happy birthday in the ...
5                             advancing the worlds women
6         the fascinating case for eating labgrown meat
7     this ceo will send your kids to school if you ...
8     top snake handler leaves sinking huckabee camp...
9     fridays morning email inside trumps presser fo...
10    airline passengers tackle man who rushes cockp...
11    facebook reportedly working on healthcare feat...
12    north korea praises trump and urges us voters ...
13    actually cnns jeffrey lord has been indefensib...
14    barcelona holds huge protest in support of ref...
15    nuclear bomb detonates during rehearsal for sp...
16    cosby lawyer asks why accusers didnt come forw...
17    stock analysts confused frightened by boar market
18    bloombergs program to build better cities just...
19                                  craig hicks indicted
Name: headline, dtype: object
```

As we can see we now have data to work with

Removing unnecesary columns

```
ds.drop('article_link', inplace = True, axis = 1)
```

Finding the maximum length for padding

```
maxlen = max([len(text) for text in ds['headline']])
```

Making all necessary imports

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.layers import Dense, Input, LSTM, Embedding,
Dropout, Activation, Flatten, Bidirectional, GlobalMaxPool1D
from tensorflow.keras.models import Model, Sequential
```

Setting parameters/attributes

```
max_features = 10000
maxlen = max([len(text) for text in ds['headline']])
embedding_size = 200
```

Tokenizer

```
tokenizer = Tokenizer(num_words = max_features, filters = '!"#$
%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n', lower = True, split = ' ',
char_level = False)
tokenizer.fit_on_texts(ds['headline'])

X = tokenizer.texts_to_sequences(ds['headline'])
X = pad_sequences(X, maxlen = maxlen)
y = np.asarray(ds['is_sarcastic'])

print("No of Samples - ", len(X))
print(X[0])
print("No of Labels - ", len(y))
print(y[0])
```

```
No of Samples -  26709
[   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    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    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    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    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    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0  287  780 3505 2213   47  353   92 2111    5
 2476 8139]
No of Labels -  26709
0
```

Volume of vocabulary

```
num_words = len(tokenizer.word_index)
print(num_words)
```

```
27667
```

GloVe Embedding

```
glove_file = "glove.6B.zip"
```

```
from zipfile import ZipFile
with ZipFile(glove_file, 'r') as z:
  z.extractall()

EMBEDDING_FILE = 'glove.6B.200d.txt'

embeddings = {}
for o in open(EMBEDDING_FILE):
    word = o.split(" ")[0]
    embd = o.split(" ")[1 :]
    embd = np.asarray(embd, dtype = 'float32')
    embeddings[word] = embd
```

Creating Weight Matrix

```
embedding_matrix = np.zeros((num_words, 200))

for word, i in tokenizer.word_index.items():
    embedding_vector = embeddings.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

len(embeddings.values())
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call
last)
Cell In[42], line 6
      4     embedding_vector = embeddings.get(word)
      5     if embedding_vector is not None:
----> 6         embedding_matrix[i] = embedding_vector
      8 len(embeddings.values())

IndexError: index 27667 is out of bounds for axis 0 with size 27667
```

Compiling the model after creation

```
import tensorflow as tf

input_layer = Input(shape = (maxlen, ), dtype = tf.int64)
embed = Embedding(embedding_matrix.shape[0], output_dim = 200, weights
= [embedding_matrix], input_length = maxlen, trainable = True)
(input_layer)
lstm = Bidirectional(LSTM(128))(embed)
drop = Dropout(0.3)(lstm)
dense = Dense(100,activation = 'relu')(drop)
out = Dense(2,activation = 'softmax')(dense)
```

Fit your model with a batch size of 100 and validation_split = 0.2. and state the validation accuracy

```
batch_size = 100
epochs = 5

model = Model(input_layer,out)
model.compile(loss = 'sparse_categorical_crossentropy', optimizer =
"adam", metrics = ['accuracy'])
model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 240)]             0

 embedding (Embedding)       (None, 240, 200)          5533400

 bidirectional (Bidirection  (None, 256)               336896
 al)

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 100)               25700

 dense_1 (Dense)             (None, 2)                 202

=================================================================
Total params: 5896198 (22.49 MB)
Trainable params: 5896198 (22.49 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size =
0.2, random_state = 10)

model.fit(X_train, y_train, batch_size = batch_size, epochs = epochs,
verbose = 1)
```

```
Epoch 1/5
214/214 [==============================] - 164s 751ms/step - loss:
0.4483 - accuracy: 0.7822
Epoch 2/5
214/214 [==============================] - 163s 763ms/step - loss:
0.2628 - accuracy: 0.8937
Epoch 3/5
214/214 [==============================] - 168s 783ms/step - loss:
0.1826 - accuracy: 0.9277
```

```
Epoch 4/5
214/214 [==============================] - 175s 819ms/step - loss:
0.1157 - accuracy: 0.9571
Epoch 5/5
214/214 [==============================] - 183s 853ms/step - loss:
0.0738 - accuracy: 0.9734

<keras.src.callbacks.History at 0x7ff135001b10>

test_pred = model.predict(np.array(X_test), verbose = 1)

167/167 [==============================] - 17s 99ms/step

test_pred = [1 if j > i else 0 for i, j in test_pred]

from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, test_pred)

array([[2686,  345],
       [ 376, 1935]])

from sklearn.metrics import classification_report

print(classification_report(y_test, test_pred))

              precision    recall  f1-score   support

           0       0.88      0.89      0.88      3031
           1       0.85      0.84      0.84      2311

    accuracy                           0.87      5342
   macro avg       0.86      0.86      0.86      5342
weighted avg       0.86      0.87      0.86      5342
```