In [1]:
```
import pandas as pd
import os
import numpy as np

import matplotlib.pyplot as plt

data = pd.read_json(os.path.join('Sarcasm_Headlines_Dataset.json'),lines=True)
```

In [2]:
```
data
```

Out[2]:

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

26709 rows × 3 columns

In [3]:
```
print (data.shape)
data.describe()
```

```
(26709, 3)
```

Out[3]:

	is_sarcastic
count	26709.000000
mean	0.438953
std	0.496269
min	0.000000
25%	0.000000
50%	0.000000
75%	1.000000
max	1.000000

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26709 entries, 0 to 26708
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   article_link  26709 non-null  object
 1   headline      26709 non-null  object
 2   is_sarcastic  26709 non-null  int64
dtypes: int64(1), object(2)
memory usage: 626.1+ KB
```

In [5]:
```
data['headline'][3]
```

Out[5]:
```
'boehner just wants wife to listen, not come up with alternative debt-reduction ideas'
```

In [6]:
```
##The column headline needs to be cleaned up as we have special characters and numbers in the column

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

data['headline']=data['headline'].apply(cleanData)
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\sys\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [7]:
```
data['headline'][3]
```

Out[7]:
```
'boehner just wants wife to listen not come up with alternative debtreduction ideas'
```

In [8]:
```
#checking for null values in train data
data.isnull().sum()
```

Out[8]:
```
article_link    0
headline        0
is_sarcastic    0
dtype: int64
```

In [9]:
```
#checking for duplicate values
data['headline'].duplicated().sum()
```

Out[9]:
```
132
```

In [10]:
```
data=data.drop(data[data['headline'].duplicated()].index,axis=0)
```

In [11]:
```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
```

Drop article_link from dataset.

As we only need headline text data and is_sarcastic column for this project. We can drop artical link column here.

In [13]:
```
data.drop('article_link',inplace=True,axis=1)
```

Get the Length of each line and find the maximum length.

As different lines are of different length. We need to pad the our sequences using the max length

In [14]:
```
maxlen = max([len(text) for text in data['headline']])
```

Modelling

Import required modules required for modelling.¶

In [15]:
```
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
```

Set Different Parameters for the model.

In [16]:
```
max_features = 10000
maxlen = max([len(text) for text in data['headline']])
embedding_size = 200
```

Apply Keras Tokenizer of headline column of your data.

Hint - First create a tokenizer instance using Tokenizer(num_words=max_features) And then fit this tokenizer instance on your data column df['headline'] using .fit_on_texts()

In [17]:
```
tokenizer = Tokenizer(num_words=max_features,filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n',lower=True,split=' ', char_level=False)
tokenizer.fit_on_texts(data['headline'])
```

In [22]:
```
labels = (data['is_sarcastic'])
data = (data['headline'])
```

In [23]:
```
train_ratio = 0.80

train_size = int(len(labels)*train_ratio)

train_data = data[:train_size]
train_labels= labels[:train_size]

test_data = data[train_size:]
test_labels = labels[train_size:]
```

In [24]:
```
tokenizer = Tokenizer(oov_token='<OOV>')
tokenizer.fit_on_texts(train_data)

vocab_size = len(tokenizer.word_index)
print(vocab_size)

train_sequences = tokenizer.texts_to_sequences(train_data)
test_sequences = tokenizer.texts_to_sequences(test_data)
```

```
24774
```

In [25]:
```
train_padded = pad_sequences(train_sequences, maxlen=maxlen,  padding='post')
test_padded = pad_sequences(test_sequences, maxlen=maxlen,  padding='post')
```

Define X and y for your model.

In [27]:
```
# Print a sample headline
index = 10
print(f'sample headline: {train_sequences[index]}')
print(f'padded sequence: {train_padded[index]} \n')

print(f'Original Sentence:  \n {tokenizer.sequences_to_texts(train_sequences[index:index+1])} \n')

# Print dimensions of padded sequences
print(f'shape of padded sequences: {train_padded.shape}')
```

```
sample headline: [3167, 1906, 4307, 14, 35, 4900, 6930, 5, 1907, 932]
padded sequence: [3167 1906 4307   14   35 4900 6930    5 1907  932    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]

Original Sentence:
 ['airline passengers tackle man who rushes cockpit in bomb threat']

shape of padded sequences: (21261, 240)
```

Get the Vocabulary size

You can use tokenizer.word_index.

In [28]:
```
num_words=len(tokenizer.word_index)
print (num_words)
```

```
24774
```

In [29]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(num_words+1,100,input_length=maxlen),
    tf.keras.layers.Bidirectional( tf.keras.layers.LSTM(128)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dropout(0.50),
    tf.keras.layers.Dense(64,activation='relu'),
    tf.keras.layers.Dense(1,activation='sigmoid')
])
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 240, 100)          2477500

 bidirectional_1 (Bidirecti  (None, 256)               234496
 onal)

 flatten_1 (Flatten)         (None, 256)               0

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 64)                16448

 dense_3 (Dense)             (None, 1)                 65

=================================================================
Total params: 2728509 (10.41 MB)
Trainable params: 2728509 (10.41 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [30]:
```
predu=model.fit(train_padded, np.array(train_labels),validation_data = (test_padded,np.array(test_labels)) , epochs = 5 , verbose=2)
```

```
Epoch 1/5
665/665 - 1091s - loss: 0.3992 - accuracy: 0.8119 - val_loss: 0.3276 - val_accuracy: 0.8585 - 1091s/epoch - 2s/step
Epoch 2/5
665/665 - 1134s - loss: 0.1662 - accuracy: 0.9373 - val_loss: 0.3646 - val_accuracy: 0.8582 - 1134s/epoch - 2s/step
Epoch 3/5
665/665 - 1118s - loss: 0.0714 - accuracy: 0.9757 - val_loss: 0.4985 - val_accuracy: 0.8499 - 1118s/epoch - 2s/step
Epoch 4/5
665/665 - 1139s - loss: 0.0311 - accuracy: 0.9898 - val_loss: 0.6469 - val_accuracy: 0.8403 - 1139s/epoch - 2s/step
Epoch 5/5
665/665 - 1133s - loss: 0.0168 - accuracy: 0.9945 - val_loss: 0.7155 - val_accuracy: 0.8416 - 1133s/epoch - 2s/step
```

In [ ]:

In [ ]: