Designing a NLP model on Sarcasm detection.

In [1]:

```
import numpy as np
import pandas as pd
import tensorflow as tf
import seaborn as sns
import re
import re,string,unicodedata
from nltk.corpus import stopwords

from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_json("Sarcasm_Headlines_Dataset_v2.json", lines=True)
df.head()
```

Out[2]:

	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-scien...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...
3	1	inclement weather prevents liar from getting t...	https://local.theonion.com/inclement-weather-p...
4	1	mother comes pretty close to using word 'strea...	https://www.theonion.com/mother-comes-pretty-c...

In [3]:

```
df.head()
```

Out[3]:

	is_sarcastic	headline	article_link
0	1	thirtysomething scientists unveil doomsday clo...	https://www.theonion.com/thirtysomething-scien...
1	0	dem rep. totally nails why congress is falling...	https://www.huffingtonpost.com/entry/donna-edw...
2	0	eat your veggies: 9 deliciously different recipes	https://www.huffingtonpost.com/entry/eat-your-...
3	1	inclement weather prevents liar from getting t...	https://local.theonion.com/inclement-weather-p...
4	1	mother comes pretty close to using word 'strea...	https://www.theonion.com/mother-comes-pretty-c...

In [4]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 28619 entries, 0 to 28618
Data columns (total 3 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   is_sarcastic  28619 non-null  int64 
 1   headline      28619 non-null  object
 2   article_link  28619 non-null  object
dtypes: int64(1), object(2)
memory usage: 670.9+ KB
```

In [5]:

```
df.shape
```

Out[5]:

```
(28619, 3)
```

checking for null values in train data

In [6]:

```
df.isnull().sum()
```

Out[6]:

```
is_sarcastic    0
headline        0
article_link    0
dtype: int64
```

In [7]:

```
df.describe(include='object')
```

Out[7]:

	headline	article_link
count	28619	28619
unique	28503	28617
top	'no way to prevent this,' says only nation whe...	https://politics.theonion.com/nation-not-sure-...
freq	12	2

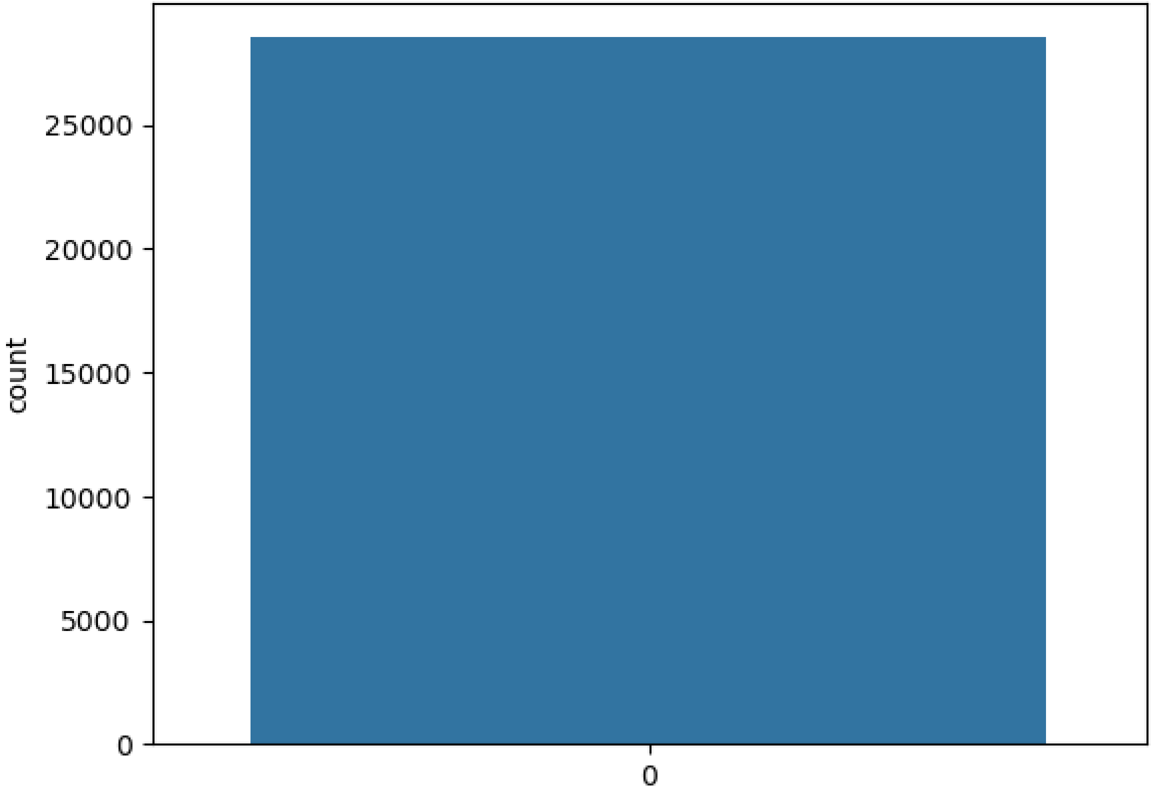
checking for duplicate values

```
In [8]: df['headline'].duplicated().sum()
```

```
Out[8]: 116
```

```
In [9]: df = df.drop(df[df['headline'].duplicated()].index,axis=0)
```

```
In [10]: sns.countplot(df['is_sarcastic']);
```



In [11]:

```
import nltk
nltk.download('stopwords')
stop = set(stopwords.words('english'))
punctuation = list(string.punctuation)
stop.update(punctuation)
```

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

Removing the stopwords from text

In [12]:

```
def split_into_words(text):
    # split into words by white space
    words = text.split()
    return words
```

```python
def to_lower_case(words):
    # convert to lower case
    words = [word.lower() for word in words]
    return words

def remove_punctuation(words):
    # prepare regex for char filtering
    re_punc = re.compile('[%s]' % re.escape(string.punctuation))
    # remove punctuation from each word
    stripped = [re_punc.sub('', w) for w in words]
    return stripped

def keep_alphabetic(words):
    # remove remaining tokens that are not alphabetic
    words = [word for word in words if word.isalpha()]
    return words

def remove_stopwords(words):
    # filter out stop words
    stop_words = set(stopwords.words('english'))
    words = [w for w in words if not w in stop_words]
    return words

def to_sentence(words):
    # join words to a sentence
    return ' '.join(words)
```

Removing the noisy text

In [13]:

```python
def denoise_text(text):
    words = split_into_words(text)
    words = to_lower_case(words)
    words = remove_punctuation(words)
    words = keep_alphabetic(words)
    words = remove_stopwords(words)
    return to_sentence(words)
```

In [14]:

```python
df['headline']=df['headline'].apply(denoise_text)
```

Apply function on review column

In [15]:

```
labels = (df['is_sarcastic'])
data = (df['headline'])
```

In [16]:

```
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 [17]:

```
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)
```

```
25662
```

In [18]:

```
maxlen=max([len(i) for i in train_sequences])
train_padded = pad_sequences(train_sequences, maxlen=maxlen,  padding='post')
test_padded = pad_sequences(test_sequences, maxlen=maxlen,  padding='post')
```

Print a sample headline

In [19]:

```
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: [1972, 2572, 315, 3022, 943, 7]
padded sequence: [1972 2572  315 3022  943    7    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    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:
 ['lesbian considered father indiana amazing one']

shape of padded sequences: (22802, 106)
```

Model Building:

In [20]:

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size+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"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding (Embedding)       (None, 106, 100)          2566300   
                                                                 
 bidirectional (Bidirection  (None, 256)               234496    
 al)                                                             
                                                                 
 flatten (Flatten)           (None, 256)               0         
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 dense (Dense)               (None, 64)                16448     
                                                                 
 dense_1 (Dense)             (None, 1)                 65        
                                                                 
=================================================================
Total params: 2817309 (10.75 MB)
Trainable params: 2817309 (10.75 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [21]:

```
history=model.fit(train_padded, np.array(train_labels),validation_data = (test_padded,np.array(test_labels)) , epochs = 5 , verb
```

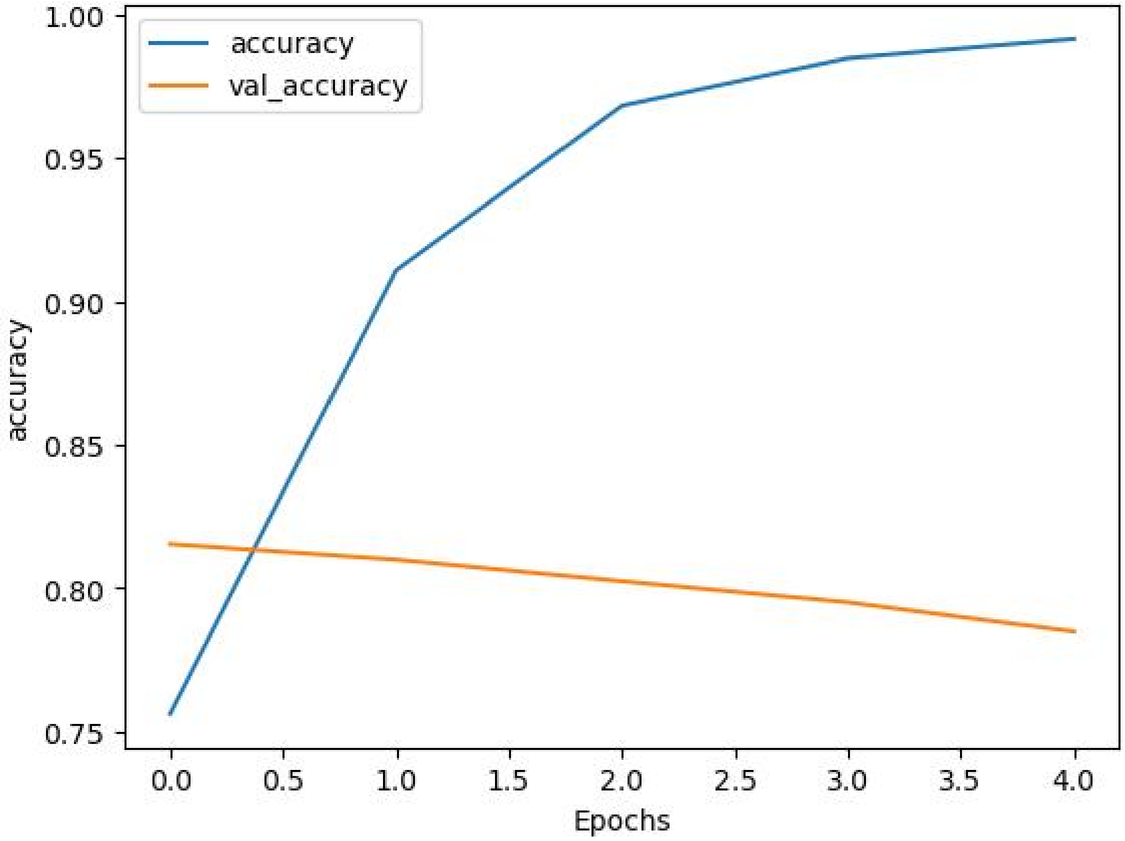
```
Epoch 1/5
713/713 - 119s - loss: 0.4809 - accuracy: 0.7563 - val_loss: 0.4040 - val_accuracy: 0.8155 - 119s/epoch - 167ms/step
Epoch 2/5
713/713 - 108s - loss: 0.2264 - accuracy: 0.9109 - val_loss: 0.4603 - val_accuracy: 0.8100 - 108s/epoch - 152ms/step
Epoch 3/5
713/713 - 106s - loss: 0.0890 - accuracy: 0.9683 - val_loss: 0.6091 - val_accuracy: 0.8025 - 106s/epoch - 149ms/step
Epoch 4/5
713/713 - 105s - loss: 0.0449 - accuracy: 0.9849 - val_loss: 0.7368 - val_accuracy: 0.7951 - 105s/epoch - 147ms/step
Epoch 5/5
713/713 - 100s - loss: 0.0263 - accuracy: 0.9915 - val_loss: 1.0178 - val_accuracy: 0.7850 - 100s/epoch - 140ms/step
```

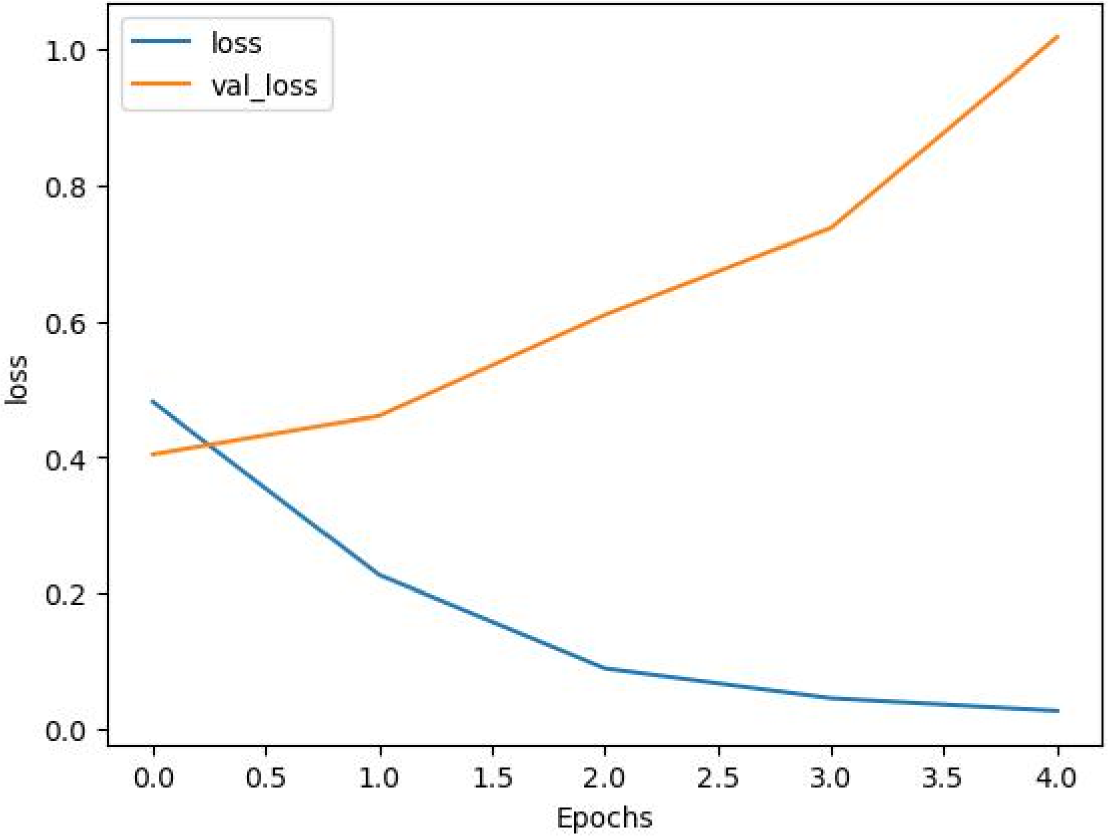
In [22]:

```
import matplotlib.pyplot as plt

# Plot utility
def plot_graphs(model, string):
  plt.plot(model.history[string])
  plt.plot(model.history['val_'+string])
  plt.xlabel("Epochs")
  plt.ylabel(string)
  plt.legend([string, 'val_'+string])
  plt.show()

# Plot the accuracy and loss
plot_graphs(history, "accuracy")
plot_graphs(history, "loss")
```




loss
val_loss
1.0
0.8
0.6
0.4
0.2
0.0
loss
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs