Sentiment analysis

In [1]:

```
# Import library
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
```

In [2]:

```
# Loading Dataset
df = pd.read_csv(r"C:\Users\Aparn\OneDrive\Desktop\DATA SCIENCES\py folder\New folder
```

In [3]:

```
# Head of dataset
df.head()
```

	text	label
0	I grew up (b. 1965) watching and loving the Th...	0
1	When I put this movie in my DVD player, and sa...	0
2	Why do people who do not know what a particula...	0
3	Even though I have great interest in Biblical ...	0
4	Im a die hard Dads Army fan and nothing will e...	1

In [4]:

```
# Define feature variable and target variable
text = df['text'].values
label = df['label'].values
```

In [5]:

```
#split the data in to train and test
train_text, test_text, train_label, test_label = train_test_split(text, label, test_s
```

In [6]:

```
# split the paragraph into tokens
tokenizer = Tokenizer(num_words=10000, oov_token='<00V>')
tokenizer.fit_on_texts(train_text)
train_sequences = tokenizer.texts_to_sequences(train_text)
test_sequences = tokenizer.texts_to_sequences(test_text)
```

In [7]:

```
maxlen=200
train_padded = pad_sequences(train_sequences, maxlen=maxlen, truncating='post', paddi
test_padded = pad_sequences(test_sequences, maxlen=maxlen, truncating='post', padding
```

In [8]:

```
# Create the input layer, hidden layer and output layer
# We are also deleting the neurons which is  not required
# We are also copiling the model
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(input_dim=10000, output_dim=16, input_length=maxlen),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(32, activation='relu'),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [9]:

```
#Training the model
model.fit(train_padded, train_label, epochs=20, validation_data=(test_padded, test_la
```

```
Epoch 1/20
1000/1000 [==============================] - 9s 8ms/step - loss: 0.4357 - accuracy: 0.7858 - val_loss: 0.3286
Epoch 2/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.2129 - accuracy: 0.9205 - val_loss: 0.3426
Epoch 3/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0866 - accuracy: 0.9726 - val_loss: 0.4716
Epoch 4/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0273 - accuracy: 0.9922 - val_loss: 0.6391
Epoch 5/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0125 - accuracy: 0.9973 - val_loss: 0.7703
Epoch 6/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0088 - accuracy: 0.9979 - val_loss: 0.8659
Epoch 7/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0100 - accuracy: 0.9968 - val_loss: 0.9116
Epoch 8/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0087 - accuracy: 0.9973 - val_loss: 0.9991
Epoch 9/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0063 - accuracy: 0.9982 - val_loss: 1.1547
Epoch 10/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0095 - accuracy: 0.9972 - val_loss: 1.1803
Epoch 11/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0067 - accuracy: 0.9980 - val_loss: 1.2296
Epoch 12/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0074 - accuracy: 0.9977 - val_loss: 1.2853
Epoch 13/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0067 - accuracy: 0.9978 - val_loss: 1.3744
Epoch 14/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0063 - accuracy: 0.9982 - val_loss: 1.3576
Epoch 15/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0047 - accuracy: 0.9984 - val_loss: 1.4476
Epoch 16/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0047 - accuracy: 0.9983 - val_loss: 1.5183
Epoch 17/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0037 - accuracy: 0.9988 - val_loss: 1.5665
Epoch 18/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0032 - accuracy: 0.9989 - val_loss: 1.6406
Epoch 19/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0050 - accuracy: 0.9982 - val_loss: 1.6438
Epoch 20/20
1000/1000 [==============================] - 7s 7ms/step - loss: 0.0058 - accuracy: 0.9983 - val_loss: 1.6855
```

```
<keras.callbacks.History at 0x14031f04c48>
```

In [10]:

```
predictions = prediction = model.predict(test_padded)
```

```
250/250 [==============================] - 1s 3ms/step
```

In [11]:

```
predictions = (predictions > 0.5).astype(int)
```

In [12]:

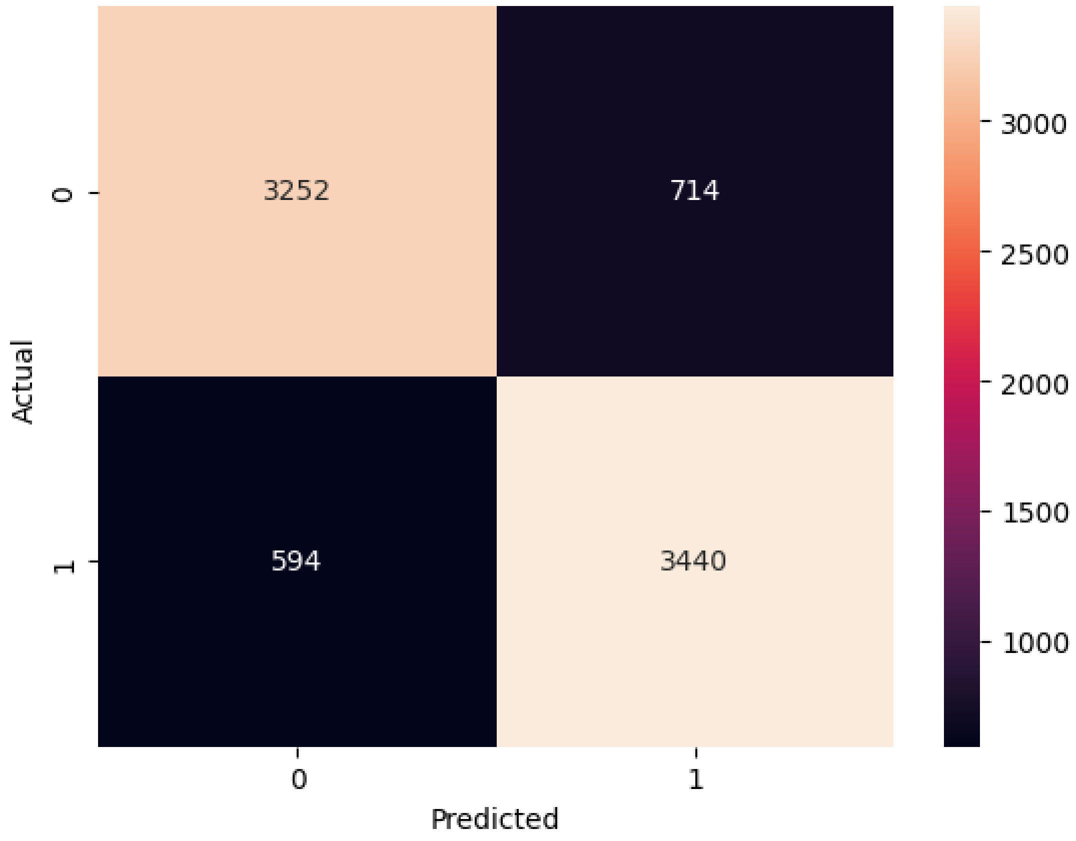
```
#Confusion matrics
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(test_label, predictions)
print(cm)
```

```
[[3252  714]
 [ 594 3440]]
```

In [13]:

```
#show heatmap
sns.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```

```
Text(50.722222222222214, 0.5, 'Actual')
```



In [14]:

```
#Import classification_report and show that we can get precision, recall, f1-score an
# this is also a performance measure
from sklearn.metrics import classification_report
print(classification_report(test_label, predictions))
```

```
              precision    recall  f1-score   support

           0       0.85      0.82      0.83      3966
           1       0.83      0.85      0.84      4034

    accuracy                           0.84      8000
   macro avg       0.84      0.84      0.84      8000
weighted avg       0.84      0.84      0.84      8000
```

In [ ]: