In [ ]:
```
#Sentiment Classification Using BERT:
```

In [1]:
```
#Step 1: Import the necessary libraries:

import os
import shutil
import tarfile
import tensorflow as tf
from transformers import BertTokenizer, TFBertForSequenceClassification
import pandas as pd
from bs4 import BeautifulSoup
import re
import matplotlib.pyplot as plt
import plotly.express as px
import plotly.offline as pyo
import plotly.graph_objects as go
from wordcloud import WordCloud, STOPWORDS
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

```
WARNING:tensorflow:From C:\Users\Ahmad\anaconda3\Lib\site-packages\keras\src\losses.p
y:2976: The name tf.losses.sparse_softmax_cross_entropy is deprecated. Please use tf.
compat.v1.losses.sparse_softmax_cross_entropy instead.
```

In [8]:
```
#Step 2: Load the dataset:
```

In [3]:
```
# Get the current working directory
current_folder = os.getcwd()

dataset = tf.keras.utils.get_file(
    fname ="aclImdb.tar.gz",
    origin ="http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz",
    cache_dir= current_folder,
    extract = True)
```

In [4]:
```
#check the dataset folder:

dataset_path = os.path.dirname(dataset)
# Check the dataset
os.listdir(dataset_path)
```

Out[4]:
```
['aclImdb', 'aclImdb.tar.gz']
```

In [7]:
```
#Check the ‘aclImdb’ directory:

# Dataset directory
dataset_dir = os.path.join(dataset_path, 'aclImdb')

# Check the Dataset directory
os.listdir(dataset_dir)
```

Out[7]:
```
['imdb.vocab', 'imdbEr.txt', 'README', 'test', 'train']
```

In [8]:
```
#Check the ‘Train’ dataset folder:
```

```
train_dir = os.path.join(dataset_dir,'train')
os.listdir(train_dir)
```

Out[8]:
```
['labeledBow.feat',
 'neg',
 'pos',
 'unsup',
 'unsupBow.feat',
 'urls_neg.txt',
 'urls_pos.txt',
 'urls_unsup.txt']
```

In [9]:
```
#Read the files of the ‘Train’ directory files:

for file in os.listdir(train_dir):
    file_path = os.path.join(train_dir, file)
    # Check if it's a file (not a directory)
    if os.path.isfile(file_path):
        with open(file_path, 'r', encoding='utf-8') as f:
            first_value = f.readline().strip()
            print(f"{file}: {first_value}")
    else:
        print(f"{file}: {file_path}")
```

```
labeledBow.feat: 9 0:9 1:1 2:4 3:4 4:6 5:4 6:2 7:2 8:4 10:4 12:2 26:1 27:1 28:1 29:2
32:1 41:1 45:1 47:1 50:1 54:2 57:1 59:1 63:2 64:1 66:1 68:2 70:1 72:1 78:1 100:1 106:
1 116:1 122:1 125:1 136:1 140:1 142:1 150:1 167:1 183:1 201:1 207:1 208:1 213:1 217:1
230:1 255:1 321:5 343:1 357:1 370:1 390:2 468:1 514:1 571:1 619:1 671:1 766:1 877:1 1
057:1 1179:1 1192:1 1402:2 1416:1 1477:2 1940:1 1941:1 2096:1 2243:1 2285:1 2379:1 29
34:1 2938:1 3520:1 3647:1 4938:1 5138:4 5715:1 5726:1 5731:1 5812:1 8319:1 8567:1 104
80:1 14239:1 20604:1 22409:4 24551:1 47304:1
neg: C:\Users\Ahmad\ML Lab\ML project 2.0\datasets\aclImdb\train\neg
pos: C:\Users\Ahmad\ML Lab\ML project 2.0\datasets\aclImdb\train\pos
unsup: C:\Users\Ahmad\ML Lab\ML project 2.0\datasets\aclImdb\train\unsup
unsupBow.feat: 0 0:8 1:6 3:5 4:2 5:1 7:1 8:5 9:2 10:1 11:2 13:3 16:1 17:1 18:1 19:1 2
2:3 24:1 26:3 28:1 30:1 31:1 35:2 36:1 39:2 40:1 41:2 46:2 47:1 48:1 52:1 63:1 67:1 6
8:1 74:1 81:1 83:1 87:1 104:1 105:1 112:1 117:1 131:1 151:1 155:1 170:1 198:1 225:1 2
26:1 288:2 291:1 320:1 331:1 342:1 364:1 374:1 384:2 385:1 407:1 437:1 441:1 465:1 46
8:1 470:1 519:1 595:1 615:1 650:1 692:1 851:1 937:1 940:1 1100:1 1264:1 1297:1 1317:1
1514:1 1728:1 1793:1 1948:1 2088:1 2257:1 2358:1 2584:2 2645:1 2735:1 3050:1 4297:1 5
385:1 5858:1 7382:1 7767:1 7773:1 9306:1 10413:1 11881:1 15907:1 18613:1 18877:1 2547
9:1
urls_neg.txt: http://www.imdb.com/title/tt0064354/usercomments
urls_pos.txt: http://www.imdb.com/title/tt0453418/usercomments
urls_unsup.txt: http://www.imdb.com/title/tt0018515/usercomments
```

In [10]:
```
#Load the Movies reviews and convert them into the pandas’ data frame with their respe
#Here 0 means Negative and 1 means Positive

def load_dataset(directory):
    data = {"sentence": [], "sentiment": []}
    for file_name in os.listdir(directory):
        print(file_name)
        if file_name == 'pos':
            positive_dir = os.path.join(directory, file_name)
            for text_file in os.listdir(positive_dir):
                text = os.path.join(positive_dir, text_file)
                with open(text, "r", encoding="utf-8") as f:
                    data["sentence"].append(f.read())
                    data["sentiment"].append(1)
        elif file_name == 'neg':
            negative_dir = os.path.join(directory, file_name)
            for text_file in os.listdir(negative_dir):
                text = os.path.join(negative_dir, text_file)
                with open(text, "r", encoding="utf-8") as f:
                    data["sentence"].append(f.read())
                    data["sentiment"].append(0)

    return pd.DataFrame.from_dict(data)
```

In [11]:
```
#Load the training datasets

# Load the dataset from the train_dir
train_df = load_dataset(train_dir)
print(train_df.head())
```

```
labeledBow.feat
neg
pos
unsup
unsupBow.feat
urls_neg.txt
urls_pos.txt
urls_unsup.txt
                                            sentence  sentiment
0  Story of a man who has unnatural feelings for ...          0
1  Airport '77 starts as a brand new luxury 747 p...          0
2  This film lacked something I couldn't put my f...          0
3  Sorry everyone,,, I know this is supposed to b...          0
4  When I was little my parents took me along to ...          0
```

In [12]:
```
#Load the test dataset respectively

test_dir = os.path.join(dataset_dir,'test')

# Load the dataset from the train_dir
test_df = load_dataset(test_dir)
print(test_df.head())
```

```
labeledBow.feat
neg
pos
urls_neg.txt
urls_pos.txt
                                            sentence  sentiment
0  Once again Mr. Costner has dragged out a movie...          0
1  This is an example of why the majority of acti...          0
2  First of all I hate those moronic rappers, who...          0
3  Not even the Beatles could write songs everyon...          0
4  Brass pictures (movies is not a fitting word f...          0
```

In [13]:
```
#Step 3: Preprocessing:
```

In [14]:
```
sentiment_counts = train_df['sentiment'].value_counts()

fig =px.bar(x= {0:'Negative',1:'Positive'},
            y= sentiment_counts.values,
            color=sentiment_counts.index,
            color_discrete_sequence = px.colors.qualitative.Dark24,
            title='<b>Sentiments Counts')
```

```
fig.update_layout(title='Sentiments Counts',
                  xaxis_title='Sentiment',
                  yaxis_title='Counts',
                  template='plotly_dark')

# Show the bar chart
fig.show()
pyo.plot(fig, filename = 'Sentiments Counts.html', auto_open = True)
```

Out[14]:
```
'Sentiments Counts.html'
```

In [16]:
```
#Text Cleaning

def text_cleaning(text):
    soup = BeautifulSoup(text, "html.parser")
    text = re.sub(r'\[[^]]*\]', '', soup.get_text())
    pattern = r"[^a-zA-Z0-9\s,']"
    text = re.sub(pattern, '', text)
    return text
```

In [17]:
```
#Apply text_cleaning

# Train dataset
train_df['Cleaned_sentence'] = train_df['sentence'].apply(text_cleaning).tolist()
# Test dataset
test_df['Cleaned_sentence'] = test_df['sentence'].apply(text_cleaning)
```

```
C:\Users\Ahmad\AppData\Local\Temp\ipykernel_18744\1023263618.py:4: MarkupResemblesLoc
atorWarning:

The input looks more like a filename than markup. You may want to open this file and
pass the filehandle into Beautiful Soup.

C:\Users\Ahmad\AppData\Local\Temp\ipykernel_18744\1023263618.py:4: MarkupResemblesLoc
atorWarning:

The input looks more like a filename than markup. You may want to open this file and
pass the filehandle into Beautiful Soup.
```

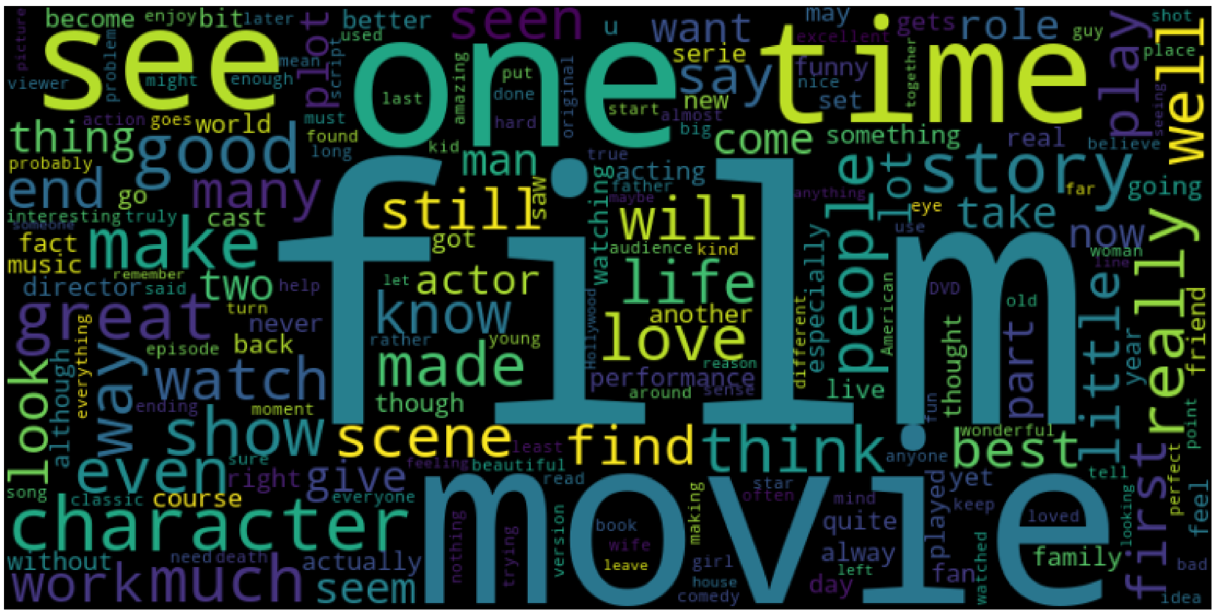
In [18]:
```python
#Plot reviews on WordCLouds

# Function to generate word cloud
def generate_wordcloud(text,Title):
    all_text = " ".join(text)
    wordcloud = WordCloud(width=800,
                          height=400,
                          stopwords=set(STOPWORDS),
                          background_color='black').generate(all_text)
    plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis("off")
    plt.title(Title)
    plt.show()
```

In [19]:
```python
#Positive Reviews

positive = train_df[train_df['sentiment']==1]['Cleaned_sentence'].tolist()
generate_wordcloud(positive,'Positive Review')
```

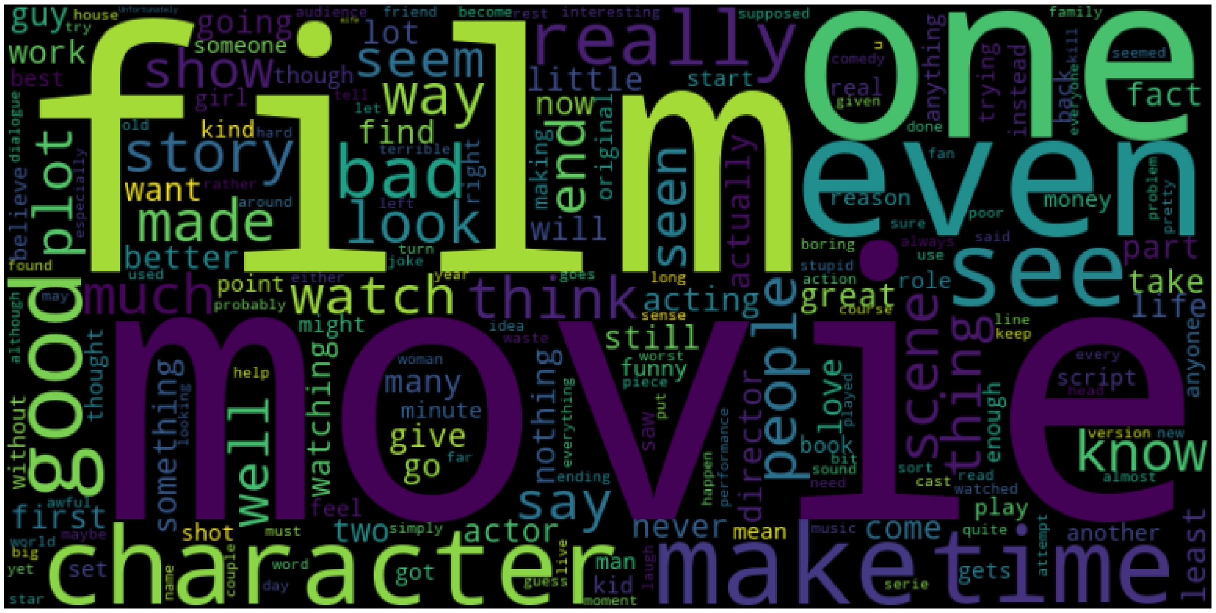
Positive Review



In [20]:
```python
#Negative Reviews

negative = train_df[train_df['sentiment']==0]['Cleaned_sentence'].tolist()
generate_wordcloud(negative,'Negative Review')
```

Negative Review



In [21]:
```python
#Separate input text and target sentiment of both train and test

# Training data
#Reviews = "[CLS] " +train_df['Cleaned_sentence'] + "[SEP]"
Reviews = train_df['Cleaned_sentence']
Target = train_df['sentiment']

# Test data
#test_reviews = "[CLS] " +test_df['Cleaned_sentence'] + "[SEP]"
test_reviews = test_df['Cleaned_sentence']
test_targets = test_df['sentiment']
```

In [22]:
```python
#Split TEST data into test and validation

x_val, x_test, y_val, y_test = train_test_split(test_reviews,
                                                test_targets,
                                                test_size=0.5,
                                                stratify = test_targets)
```

In [23]:
```python
#Step 4: Tokenization & Encoding
```

In [24]:
```python
#Load the pre-trained BERT tokenizer

#Tokenize and encode the data using the BERT tokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased', do_lower_case=True)
```

In [25]:
```python
#Apply the BERT tokenization in training, testing and validation dataset

max_len= 128
# Tokenize and encode the sentences
X_train_encoded = tokenizer.batch_encode_plus(Reviews.tolist(),
                                              padding=True,
                                              truncation=True,
                                              max_length = max_len,
                                              return_tensors='tf')

X_val_encoded = tokenizer.batch_encode_plus(x_val.tolist(),
                                              padding=True,
                                              truncation=True,
                                              max_length = max_len,
                                              return_tensors='tf')

X_test_encoded = tokenizer.batch_encode_plus(x_test.tolist(),
                                              padding=True,
                                              truncation=True,
                                              max_length = max_len,
                                              return_tensors='tf')
```

In [26]:
```
#Check the encoded dataset

k = 0
print('Training Comments -->>',Reviews[k])
print('\nInput Ids -->>\n',X_train_encoded['input_ids'][k])
print('\nDecoded Ids -->>\n',tokenizer.decode(X_train_encoded['input_ids'][k]))
print('\nAttention Mask -->>\n',X_train_encoded['attention_mask'][k])
print('\nLabels -->>',Target[k])
```

```
Training Comments -->> Story of a man who has unnatural feelings for a pig Starts out
with a opening scene that is a terrific example of absurd comedy A formal orchestra a
udience is turned into an insane, violent mob by the crazy chantings of it's singers
Unfortunately it stays absurd the WHOLE time with no general narrative eventually mak
ing it just too off putting Even those from the era should be turned off The cryptic
dialogue would make Shakespeare seem easy to a third grader On a technical level it's
better than you might think with some good cinematography by future great Vilmos Zsig
mond Future stars Sally Kirkland and Frederic Forrest can be seen briefly

Input Ids -->>
 tf.Tensor(
[  101  2466  1997  1037  2158  2040  2038 21242  5346  2005  1037 10369
  4627  2041  2007  1037  3098  3496  2008  2003  1037 27547  2742  1997
 18691  4038  1037  5337  4032  4378  2003  2357  2046  2019  9577  1010
  6355 11240  2011  1996  4689 22417  2015  1997  2009  1005  1055  8453
  6854  2009 12237 18691  1996  2878  2051  2007  2053  2236  7984  2776
  2437  2009  2074  2205  2125  5128  2130  2216  2013  1996  3690  2323
  2022  2357  2125  1996 26483  7982  2052  2191  8101  4025  3733  2000
  1037  2353  3694  2099  2006  1037  4087  2504  2009  1005  1055  2488
  2084  2017  2453  2228  2007  2070  2204 16434  2011  2925  2307  6819
 13728  2891  1062  5332 21693 15422  2925  3340  8836 11332  3122  1998
 15296 16319  2064  2022  2464  4780   102     0], shape=(128,), dtype=int32)

Decoded Ids -->>
 [CLS] story of a man who has unnatural feelings for a pig starts out with a opening
scene that is a terrific example of absurd comedy a formal orchestra audience is turn
ed into an insane, violent mob by the crazy chantings of it's singers unfortunately i
t stays absurd the whole time with no general narrative eventually making it just too
off putting even those from the era should be turned off the cryptic dialogue would m
ake shakespeare seem easy to a third grader on a technical level it's better than you
might think with some good cinematography by future great vilmos zsigmond future star
s sally kirkland and frederic forrest can be seen briefly [SEP] [PAD]

Attention Mask -->>
 tf.Tensor(
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0], shape=(128,), dtype=int32)

Labels -->> 0
```

In [27]:
```python
#Step 5: Build the classification model
```

In [28]:
```python
#Load the model

# Intialize the model
model = TFBertForSequenceClassification.from_pretrained('bert-base-uncased', num_label
```

```
model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]
```

```
C:\Users\Ahmad\anaconda3\Lib\site-packages\huggingface_hub\file_download.py:147: User
Warning:

`huggingface_hub` cache-system uses symlinks by default to efficiently store duplicat
ed files but your machine does not support them in C:\Users\Ahmad\.cache\huggingface
\hub. Caching files will still work but in a degraded version that might require more
space on your disk. This warning can be disabled by setting the `HF_HUB_DISABLE_SYMLI
NKS_WARNING` environment variable. For more details, see https://huggingface.co/docs/
huggingface_hub/how-to-cache#limitations.
To support symlinks on Windows, you either need to activate Developer Mode or to run
Python as an administrator. In order to see activate developer mode, see this articl
e: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-d
evelopment
```

```
WARNING:tensorflow:From C:\Users\Ahmad\anaconda3\Lib\site-packages\keras\src\backend.
py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_defa
ult_graph instead.
```

```
All PyTorch model weights were used when initializing TFBertForSequenceClassificatio
n.

Some weights or buffers of the TF 2.0 model TFBertForSequenceClassification were not
initialized from the PyTorch model and are newly initialized: ['classifier.weight',
'classifier.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for p
redictions and inference.
```

In [29]:

```
#Compile the model

# Compile the model with an appropriate optimizer, loss function, and metrics
optimizer = tf.keras.optimizers.Adam(learning_rate=2e-5)
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
model.compile(optimizer=optimizer, loss=loss, metrics=[metric])
```

In [30]:

```
# Step 5: Train the model
history = model.fit(
    [X_train_encoded['input_ids'], X_train_encoded['token_type_ids'], X_train_encoded[
    Target,
    validation_data=(
    [X_val_encoded['input_ids'], X_val_encoded['token_type_ids'], X_val_encoded['atter
    batch_size=32,
    epochs=3
)
```

```
Epoch 1/3
WARNING:tensorflow:From C:\Users\Ahmad\anaconda3\Lib\site-packages\keras\src\utils\tf
_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.comp
at.v1.ragged.RaggedTensorValue instead.

782/782 [==============================] - 9930s 13s/step - loss: 0.3477 - accuracy:
0.8410 - val_loss: 0.3082 - val_accuracy: 0.8734
Epoch 2/3
782/782 [==============================] - 8050s 10s/step - loss: 0.2033 - accuracy:
0.9200 - val_loss: 0.2930 - val_accuracy: 0.8843
Epoch 3/3
782/782 [==============================] - 8748s 11s/step - loss: 0.1078 - accuracy:
0.9606 - val_loss: 0.3566 - val_accuracy: 0.8864
```

In [31]:
```
#Step 6:Evaluate the model:
```

In [32]:
```
#Evaluate the model on the test data
test_loss, test_accuracy = model.evaluate(
    [X_test_encoded['input_ids'], X_test_encoded['token_type_ids'], X_test_encoded['at
    y_test
)
print(f'Test loss: {test_loss}, Test accuracy: {test_accuracy}')
```

```
391/391 [==============================] - 1083s 3s/step - loss: 0.3610 - accuracy:
0.8860
Test loss: 0.360990047454834, Test accuracy: 0.8859999775886536
```

In [33]:
```
#Save the model and tokenizer to the local folder

path = 'path-to-save'
# Save tokenizer
tokenizer.save_pretrained(path +'/Tokenizer')

# Save model
model.save_pretrained(path +'/Model')
```

In [34]:
```
#Load the model and tokenizer from the local folder

# Load tokenizer
bert_tokenizer = BertTokenizer.from_pretrained(path +'/Tokenizer')

# Load model
bert_model = TFBertForSequenceClassification.from_pretrained(path +'/Model')
```

```
Some layers from the model checkpoint at path-to-save/Model were not used when initia
lizing TFBertForSequenceClassification: ['dropout_37']
- This IS expected if you are initializing TFBertForSequenceClassification from the c
heckpoint of a model trained on another task or with another architecture (e.g. initi
alizing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFBertForSequenceClassification from t
he checkpoint of a model that you expect to be exactly identical (initializing a Bert
ForSequenceClassification model from a BertForSequenceClassification model).
All the layers of TFBertForSequenceClassification were initialized from the model che
ckpoint at path-to-save/Model.
If your task is similar to the task the model of the checkpoint was trained on, you c
an already use TFBertForSequenceClassification for predictions without further traini
ng.
```

In [35]:
```
#Predict the sentiment of the test dataset

pred = bert_model.predict(
    [X_test_encoded['input_ids'], X_test_encoded['token_type_ids'], X_test_encoded['at

# pred is of type TFSequenceClassifierOutput
logits = pred.logits

# Use argmax along the appropriate axis to get the predicted labels
pred_labels = tf.argmax(logits, axis=1)

# Convert the predicted labels to a NumPy array
pred_labels = pred_labels.numpy()

label = {
    1: 'positive',
    0: 'Negative'
}

# Map the predicted labels to their corresponding strings using the label dictionary
pred_labels = [label[i] for i in pred_labels]
Actual = [label[i] for i in y_test]

print('Predicted Label :', pred_labels[:10])
print('Actual Label :', Actual[:10])
```

```
391/391 [==============================] - 1129s 3s/step
Predicted Label : ['Negative', 'positive', 'positive', 'Negative', 'Negative', 'Negat
ive', 'Negative', 'positive', 'Negative', 'positive']
Actual Label : ['Negative', 'positive', 'positive', 'Negative', 'Negative', 'Negativ
e', 'Negative', 'positive', 'Negative', 'Negative']
```

In [36]:
```
#Classification Report

print("Classification Report: \n", classification_report(Actual, pred_labels))
```

```
Classification Report:
               precision    recall  f1-score   support

    Negative       0.89      0.88      0.88      6250
    positive       0.88      0.89      0.89      6250

    accuracy                           0.89     12500
   macro avg       0.89      0.89      0.89     12500
weighted avg       0.89      0.89      0.89     12500
```

In [37]:
```
#Step 7: Prediction with user inputs
```

In [38]:
```
def Get_sentiment(Review, Tokenizer=bert_tokenizer, Model=bert_model):
    # Convert Review to a list if it's not already a list
    if not isinstance(Review, list):
        Review = [Review]

    Input_ids, Token_type_ids, Attention_mask = Tokenizer.batch_encode_plus(Review,
                                                                             padding=Tr
                                                                             truncatior
                                                                             max_length
                                                                             return_ter
    prediction = Model.predict([Input_ids, Token_type_ids, Attention_mask])

    # Use argmax along the appropriate axis to get the predicted labels
    pred_labels = tf.argmax(prediction.logits, axis=1)

    # Convert the TensorFlow tensor to a NumPy array and then to a list to get the pre
    pred_labels = [label[i] for i in pred_labels.numpy().tolist()]
    return pred_labels
```

In [39]:
```
#Let’s predict with our own review

Review ='''Bahubali is a blockbuster Indian movie that was released in 2015.
It is the first part of a two-part epic saga that tells the story of a legendary hero
The movie has received rave reviews from critics and audiences alike for its stunning
spectacular action scenes, and captivating storyline.'''
Get_sentiment(Review)
```

```
1/1 [==============================] - 4s 4s/step
['positive']
```

Out[39]:

In [ ]: