EXPERIMENT 4

SENTIMENT ANALYSIS ON IMDB DATASET USING BI-DIRECTIONAL LSTM

AIM:

To implement and evaluate a Bi-directional LSTM model for sentiment classification using the IMDB movie review dataset. The model will learn to classify reviews as positive or negative.

PRE-REQUISITES:

1. Basics of Machine Learning
2. Python Programming
3. Knowledge on Numpy, Pandas, Matplotlib, TensorFlow/ Keras
4. Jupyter Notebook
5. Data Pre-Processing Techniques
6. Knowledge on Neural Networks

IMDB Dataset

- IMDB Dataset consists of 50,000 movie reviews along with sentiment labels (positive or negative).
- Popular for natural language processing (NLP) tasks like sentiment analysis, text classification.
- It is a balanced Dataset with 25000 Positive & 25000 Negative Reviews.
- Download the Dataset - https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

1. Importing the Basic Libraries

In [33]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the Dataset

In [34]:
```
df = pd.read_csv('IMDB Dataset.csv')
df.head()
```

Out[34]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

In [35]:
```
# Clean the HTML tags in the Dataset
import re

df['review'] = df['review'].apply(lambda x: re.sub(r'<.*?>', '', x))

# Strip extra spaces also
df['review'] = df['review'].str.strip()
```

In [36]:
```
df.head()
```

Out[36]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The filming tec...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

3. Pre-Process the Dataset

In [37]:
```
# Import the Required Libraries

from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

a) Convert labels to binary (positive=1, negative=0)

In [38]:
```
df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})
df.head()
```

Out[38]:

	review	sentiment
0	One of the other reviewers has mentioned that ...	1
1	A wonderful little production. The filming tec...	1
2	I thought this was a wonderful way to spend ti...	1
3	Basically there's a family where a little boy ...	0
4	Petter Mattei's "Love in the Time of Money" is...	1

b) Split the Dataset into Training & Testing Sets

In [39]:
```
X_train, X_test, y_train, y_test = train_test_split(df['review'], df['sentiment'], test_size=0.2, random_state=101)

X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[39]:
```
((40000,), (10000,), (40000,), (10000,))
```

c) Apply Tokenization & Generate Numerical Sequences

In [41]:
```
tokenizer = Tokenizer(num_words=10000, oov_token="<OOV>")
tokenizer.fit_on_texts(X_train)
```

In [43]:
```
# Convert text to sequences
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
```

d) Apply Padding

In [44]:
```
maxlen = 200
X_train_pad = pad_sequences(X_train_seq, maxlen=maxlen, padding='post')
X_test_pad = pad_sequences(X_test_seq, maxlen=maxlen, padding='post')
```

4. Build the Bi-directional LSTM Model

In [50]:
```
# Import the required Libraries
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dense, Dropout
```

In [51]:
```
# Creating the Model Instance
model = Sequential()

# Add the Layers
model.add(Embedding(input_dim=10000, output_dim=128, input_length=maxlen))
model.add(Bidirectional(LSTM(64, return_sequences=False)))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

In [53]:
```
# Compile the Model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

5. Train the Model

In [56]:
```
# Import and apply Early Stopping
from tensorflow.keras.callbacks import EarlyStopping

early_stop = EarlyStopping(monitor='val_loss', patience=2, restore_best_weights=True)
```

In [57]:
```
# Train the Model

model.fit(X_train_pad, y_train, validation_data=(X_test_pad, y_test),
          epochs=5, batch_size=64, callbacks=[early_stop],verbose=1)
```

```
Epoch 1/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 111s 176ms/step - accuracy: 0.6797 - loss: 0.5863 - val_accuracy: 0.8305 - val_loss:
0.4138
Epoch 2/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 86s 138ms/step - accuracy: 0.8535 - loss: 0.3622 - val_accuracy: 0.8397 - val_loss: 0.
4573
Epoch 3/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 90s 143ms/step - accuracy: 0.9101 - loss: 0.2403 - val_accuracy: 0.8837 - val_loss: 0.
2747
Epoch 4/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 87s 139ms/step - accuracy: 0.9328 - loss: 0.1850 - val_accuracy: 0.8861 - val_loss: 0.
2835
Epoch 5/5
625/625 ━━━━━━━━━━━━━━━━━━━━ 89s 142ms/step - accuracy: 0.9516 - loss: 0.1383 - val_accuracy: 0.8873 - val_loss: 0.
3058
```

Out[57]: <keras.src.callbacks.history.History at 0x35ef38910>

6. Evaluate the Model Performance

In [58]:
```
loss, accuracy = model.evaluate(X_test_pad, y_test, verbose=1)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 8s 25ms/step - accuracy: 0.8852 - loss: 0.2705
```

In [59]:
```
loss, accuracy
```

Out[59]: (0.2747051417827606, 0.8837000131607056)

In [63]:
```
# Generate y_predictions for calculating metrics
y_pred = (model.predict(X_test_pad) > 0.5).astype("int32")
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 8s 25ms/step
```

In [64]:
```
# Print the Classification Report
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.90      0.86      0.88      4959
           1       0.87      0.91      0.89      5041

    accuracy                           0.88     10000
   macro avg       0.88      0.88      0.88     10000
weighted avg       0.88      0.88      0.88     10000
```

7. Generate New Predictions for unseen data

In [67]:
```
review = ["This movie was a complete waste of time and money"]
review_seq = tokenizer.texts_to_sequences(review)
review_pad = pad_sequences(review_seq, maxlen=maxlen, padding='post')
```

In [71]:
```
prediction = model.predict(review_pad)
print("Predicted Sentiment:", "Positive" if prediction > 0.5 else "Negative")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step
Predicted Sentiment: Negative
```

In [76]:
```
review = ["This movie is really motive"]
review_seq = tokenizer.texts_to_sequences(review)
review_pad = pad_sequences(review_seq, maxlen=maxlen, padding='post')
```

In [77]:
```
prediction = model.predict(review_pad)
print("Predicted Sentiment:", "Positive" if prediction > 0.5 else "Negative")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 28ms/step
Predicted Sentiment: Positive
```

RESULT:

The Bi-Directional LSTM model was successfully implemented on the IMDB dataset for sentiment analysis. The model achieved 88% accuracy on the test set, demonstrating its ability to predict the new reviews for unseen data.