Experiment No 8

Aim:

To design and implement a Long Short-Term Memory (LSTM) model for text classification

Apparatus / Software Requirements:

- Python 3.x
- TensorFlow 2.x (Keras API)
- NumPy
- A computer with internet access for downloading datasets

Theory:

Natural Language Processing (NLP) focuses on enabling computers to understand and interpret human language. Text data is sequential, meaning the meaning of a word often depends on the words around it. To handle this sequential nature, **Recurrent Neural Networks (RNNs)** are commonly used because they can process sequences by maintaining a hidden state that captures information from previous time steps.

However, standard RNNs have limitations due to the **vanishing and exploding gradient problems**, which make it difficult to learn long-range dependencies in text. As a result, they may "forget" important context from earlier in a sequence.

Long Short-Term Memory (LSTM) networks were introduced to overcome these issues. LSTM is a type of RNN that uses special gating mechanisms to control the flow of information:

- **Forget Gate:** Determines which information to discard from the memory cell.
- **Input Gate:** Determines which new information to store in the cell.
- **Output Gate:** Determines what information to output as the hidden state.

This allows LSTMs to retain important context over long sequences, making them well-suited for tasks such as sentiment analysis, text classification, and language modeling.

Before feeding text into an LSTM, words are represented using **embedding vectors**, which map discrete words into dense numerical vectors that capture semantic relationships. For example, words like "good" and "excellent" have similar embeddings, whereas "good" and "bad" are far apart.

For binary sentiment classification (positive vs. negative), the LSTM processes the sequences and passes the output through a **Dense layer with sigmoid activation**, producing a probability

score for the positive class. LSTM models can learn contextual patterns from sequences without extensive manual feature engineering, making them highly effective for text classification tasks.

Procedure:

1. **Data Loading:**
    - Import the IMDB dataset from `tensorflow.keras.datasets.imdb` with a vocabulary size limit of 10,000 words.

2. **Data Preprocessing:**
    - Pad and truncate sequences to a fixed length of 256 to ensure uniform input size.

3. **Model Construction:**
    - Add an **Embedding Layer** to convert words to dense vectors.
    - Add an **LSTM Layer** with 128 units and dropout regularization.
    - Add a **Dense Output Layer** with sigmoid activation for binary classification.

4. **Compilation:**
    - Use binary cross-entropy as the loss function and Adam optimizer with accuracy as a metric.

5. **Training:**
    - Train the model for 10 epochs with a batch size of 128, validating on the test set.

6. **Prediction:**
    - Decode and predict sentiment of unseen test reviews using the trained model.

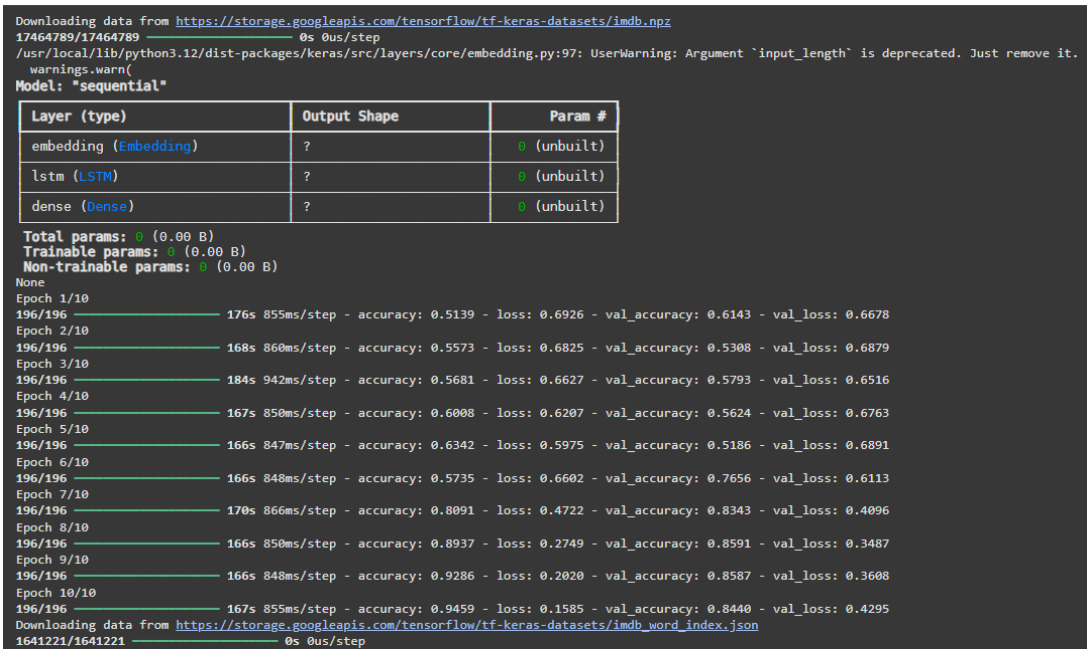
7. **Evaluation:**
    - Compare predicted sentiments with actual labels to compute accuracy.

Code:

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from tensorflow.keras.preprocessing.sequence import pad_sequences
import numpy as np

# --- 1. Configuration and Data Loading (IMDB) ---
MAX_WORDS = 10000
MAX_SEQUENCE_LENGTH = 256
EMBEDDING_DIM = 100
LSTM_UNITS = 128
```

```
EPOCHS = 10
BATCH_SIZE = 128

(X_train, y_train), (X_test, y_test) = tf.keras.datasets.imdb.load_data(num_words=MAX_WORDS)

# --- 2. Text Preprocessing and Padding ---
X_train_padded = pad_sequences(X_train, maxlen=MAX_SEQUENCE_LENGTH, padding='post', truncating='post')
X_test_padded = pad_sequences(X_test, maxlen=MAX_SEQUENCE_LENGTH, padding='post', truncating='post')
VOCAB_SIZE = MAX_WORDS

# --- 3. Define the LSTM Model for Binary Classification ---
model = Sequential()
model.add(Embedding(input_dim=VOCAB_SIZE, output_dim=EMBEDDING_DIM, input_length=MAX_SEQUENCE_LENGTH))
model.add(LSTM(units=LSTM_UNITS, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(units=1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())

# --- 4. Train the Model ---
history = model.fit(X_train_padded, y_train, epochs=EPOCHS, batch_size=BATCH_SIZE, validation_data=(X_test_padded, y_test), verbose=1)

# --- 5. Sentiment Prediction Function ---
def predict_sentiment_from_test_data(test_index):
    word_index = tf.keras.datasets.imdb.get_word_index()
    reverse_word_index = dict([(value, key) for (key, value) in word_index.items()])
    raw_sequence = X_test[test_index]
    decoded_review = ' '.join([reverse_word_index.get(i - 3, '?') for i in raw_sequence])
    input_sequence = X_test_padded[test_index].reshape(1, MAX_SEQUENCE_LENGTH)
    probability = model.predict(input_sequence, verbose=0)[0][0]
    sentiment = "Positive (1)" if probability >= 0.5 else "Negative (0)"
    confidence = f"{probability*100:.2f}%"
    true_label = "Positive (1)" if y_test[test_index] == 1 else "Negative (0)"
      return {'text': decoded_review, 'true_label': true_label, 'prediction': sentiment, 'confidence': confidence}
# --- 6. Demonstration ---
result_pos = predict_sentiment_from_test_data(0)
print(result_pos)
result_neg = predict_sentiment_from_test_data(5)
print(result_neg)
```

Result:



{'text': "? please give this one a miss br br ? ? and the rest of the cast rendered terrible performances the show is flat flat flat br br i don't know how michael madison could have allowed this one on his plate he almost seemed to know this wasn't going to work out and his performance was quite ? so all you madison fans give this a miss", 'true_label': 'Negative (0)', 'prediction': 'Negative (0)', 'confidence': '4.09%'}

{'text': "? i'm absolutely disgusted this movie isn't being sold all who love this movie should email disney and increase the demand for it they'd eventually have to sell it then i'd buy copies for everybody i know everything and everybody in this movie did a good job and i haven't figured out why disney hasn't put this movie on dvd or on vhs in rental stores at least i haven't seen any copies this is a wicked good movie and should be seen by all the kids in the new generation don't get to see it and i think they should it should at least be put back on the channel this movie doesn't deserve a cheap ? it deserves the real thing i'm them now this movie will be on dvd", 'true_label': 'Positive (1)', 'prediction': 'Negative (0)', 'confidence': '38.72%'}

Conclusion:

The LSTM-based model effectively learned to classify movie reviews as positive or negative, demonstrating the power of recurrent architectures for text classification tasks. The use of word embeddings and LSTM layers enabled the model to capture contextual information and improve prediction accuracy.