**Aim:** Write a program to build recurrent neural network for text data using tensorflow

**Objective:**

We'll classify text data (e.g., IMDB movie reviews: positive/negative sentiment) using an RNN with an embedding layer and LSTM (a popular RNN variant).

**Steps Overview:**

1. Import Libraries
2. Load and Preprocess Dataset
3. Build the RNN Model
4. Compile the Model
5. Train the Model
6. Evaluate the Model
7. Visualize Training Progress
8. Test with Custom Input (Optional)

**Implementation:**

**Step 1: Install & Import Libraries**

*# Make sure you have TensorFlow installed*

*# pip install tensorflow*

*import tensorflow as tf*

*from tensorflow.keras.preprocessing.text import Tokenizer*

*from tensorflow.keras.preprocessing.sequence import pad\_sequences*

*from tensorflow.keras.models import Sequential*

*from tensorflow.keras.layers import Embedding, LSTM, Dense*

*import numpy as np*

**Step 2: Load Dataset - For simplicity, we'll use the IMDB movie reviews dataset available in Keras**

*from tensorflow.keras.datasets import imdb*

*# Only keep the top 10,000 most frequent words*

*vocab\_size = 10000*

*(x\_train, y\_train), (x\_test, y\_test) = imdb.load\_data(num\_words=vocab\_size)*

**Step 3: Pad Sequences**

*Neural networks need fixed-length input, so we pad shorter sequences*

*max\_length = 200 # max length of review*

*x\_train = pad\_sequences(x\_train, maxlen=max\_length)*

*x\_test = pad\_sequences(x\_test, maxlen=max\_length)*

**Step 4: Build the RNN Model**

*We'll use an* ***Embedding layer****, an* ***LSTM*** *layer, and a* ***Dense output****.*

*model = Sequential([*

*Embedding(input\_dim=vocab\_size, output\_dim=64),*

*LSTM(units=64),*

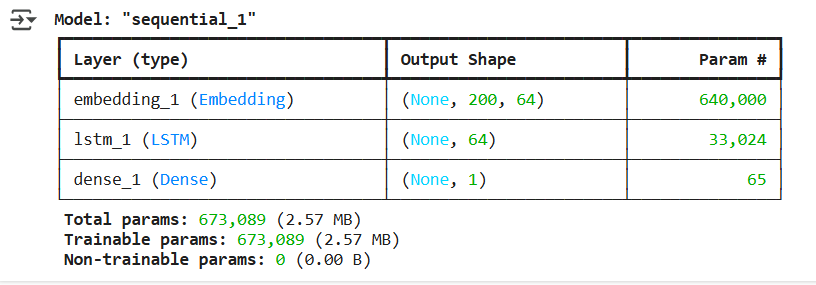
*Dense(1, activation='sigmoid')*

*])*

*# Build the model manually to avoid warnings and show summary properly*

*model.build(input\_shape=(None, max\_length))*

*model.summary()*

**

| ***Layer*** | ***Output Shape*** | ***Description*** |
| --- | --- | --- |
| ***Embedding*** | *(None, 200, 64)* | *Transforms each word (integer) into a 64-dimensional vector. The input sequence has max length 200. Total params = vocab\_size \* embedding\_dim = 10000 \* 64 = 640,000.* |
| ***LSTM*** | *(None, 64)* | *Processes the sequence and outputs a 64-dim summary of the whole review. Parameters depend on the number of LSTM units (here 64).* |
| ***Dense*** | *(None, 1)* | *Final output layer with sigmoid activation for* ***binary classification*** *(positive/negative sentiment).* |

*# Build the model*

*model = Sequential([*

*Embedding(input\_dim=vocab\_size, output\_dim=64),*

*LSTM(units=64),*

*Dense(1, activation='sigmoid')*

*])*

*# Build shape manually*

*model.build(input\_shape=(None, max\_length))*

*# Compile the model*

*model.compile(*

*loss='binary\_crossentropy',*

*optimizer='adam',*

*metrics=['accuracy']*

*)*

*# Now train the model*

*history = model.fit(*

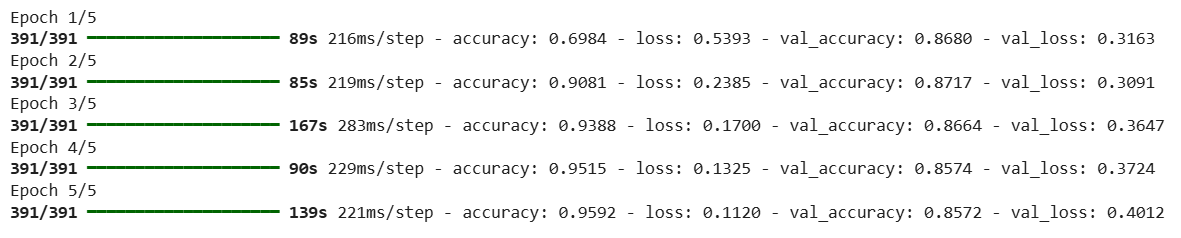
*x\_train, y\_train,*

*epochs=5,*

*batch\_size=64,*

*validation\_data=(x\_test, y\_test)*

*)*

**

*loss, accuracy = model.evaluate(x\_test, y\_test)*

*print(f"Test Accuracy: {accuracy:.4f}")*

**

***Plot Training & Validation Curves:***

*import matplotlib.pyplot as plt*

*# Accuracy plot*

*plt.plot(history.history['accuracy'], label='Train Accuracy')*

*plt.plot(history.history['val\_accuracy'], label='Val Accuracy')*

*plt.title("Accuracy per Epoch")*

*plt.xlabel("Epoch")*

*plt.ylabel("Accuracy")*

*plt.legend()*

*plt.grid(True)*

*plt.show()*

*# Loss plot*

*plt.plot(history.history['loss'], label='Train Loss')*

*plt.plot(history.history['val\_loss'], label='Val Loss')*

*plt.title("Loss per Epoch")*

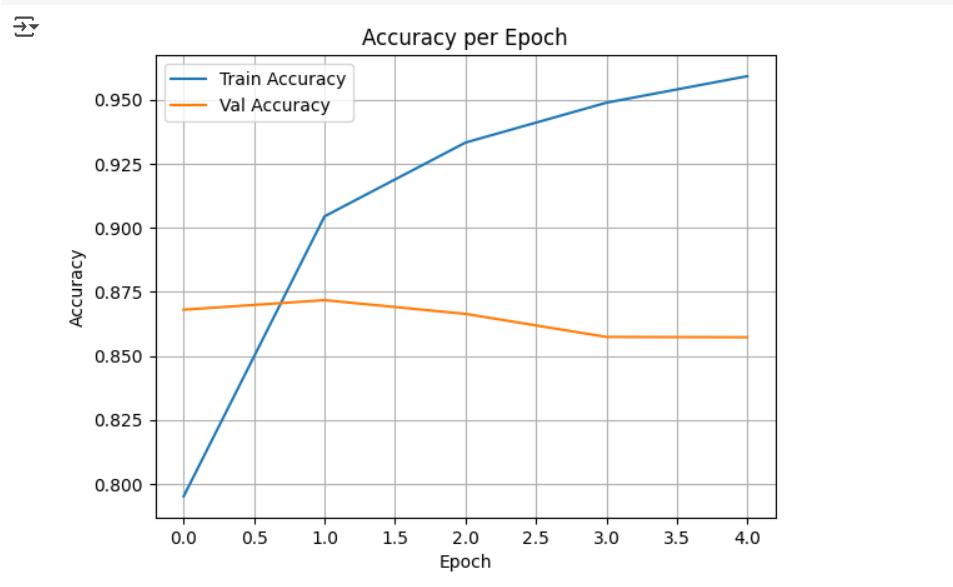
*plt.xlabel("Epoch")*

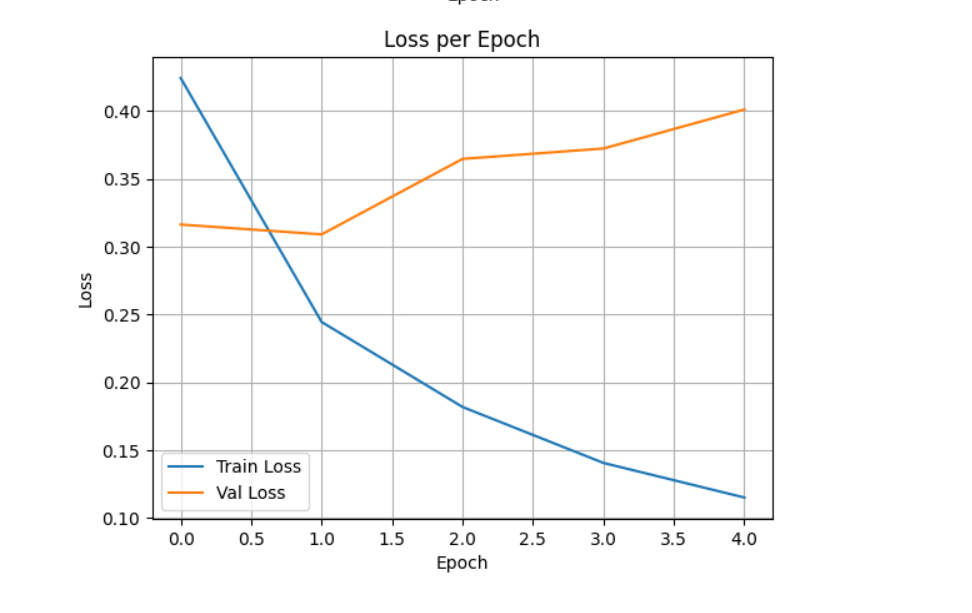
*plt.ylabel("Loss")*

*plt.legend()*

*plt.grid(True)*

*plt.show()*





***Predict on Custom Text***

*from tensorflow.keras.preprocessing.text import Tokenizer*

*from tensorflow.keras.preprocessing.sequence import pad\_sequences*

*# Sample custom text*

*texts = ["The movie was fantastic!", "I did not like the plot."]*

*# Fit tokenizer on your training text if you had custom data*

*tokenizer = Tokenizer(num\_words=vocab\_size)*

*tokenizer.fit\_on\_texts(texts)  # Use your own dataset here*

*# Convert to sequences and pad*

*sequences = tokenizer.texts\_to\_sequences(texts)*

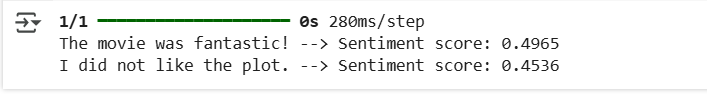
*padded = pad\_sequences(sequences, maxlen=max\_length)*

*# Predict*

*predictions = model.predict(padded)*

*# Show predictions*

*for i, text in enumerate(texts):*

******

**Conclusion:**We successfully built and trained a Recurrent Neural Network using TensorFlow to classify text data. By leveraging an Embedding layer and LSTM, the model learned to capture the sequential structure of movie reviews and achieved effective performance in binary sentiment classification. This implementation serves as a foundational approach to text classification tasks using deep learning.