Semester	B.E. Semester VII
Subject	Deep Learning
Subject Professor In-charge	Dr. Nayana Mahajan
Laboratory	M201B

Student Name	Harsh Jain	Division	B
Roll Number	22108B0054	Batch	4
Grade and Subject Teacher's Signature			

Experiment Number	7
Experiment Title	Implement RNN for sentiment analysis on movie reviews.
Resources / Apparatus Required	Software: Google Colab

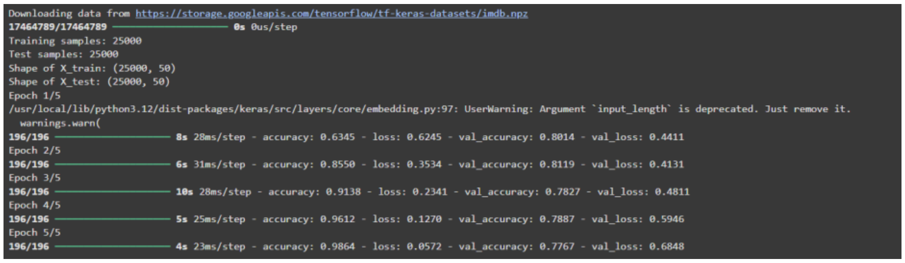
Algorithm

- **Start**
- **Import Required Libraries**
- **Load IMDB Movie Review Dataset**
- **Pad Sequences to Equal Length**
- **Define Sequential Model**
- **Add Embedding and SimpleRNN Layers**
- **Add Dense Output Layer with Sigmoid Activation**
- **Compile the Model with Optimizer and Loss Function**
- **Train the Model with Training and Validation Data**
- **Evaluate Model Performance and Display Results**

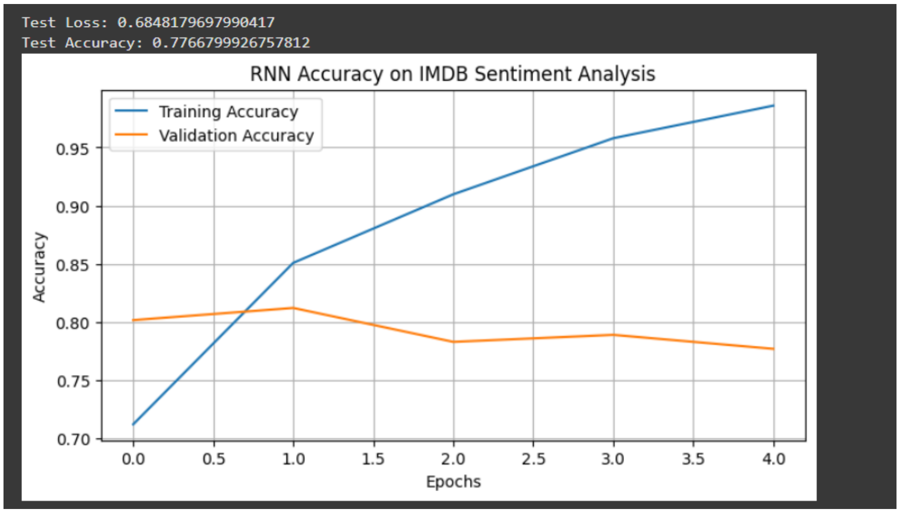
Program code

```
#      RNN Sentiment Analysis on IMDB Movie Reviews (Google Colab Compatible)

import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, SimpleRNN, Embedding

# ✅ 1. Load IMDB dataset (top 10,000 most frequent words)
num_words = 10000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=num_words)

print("Training samples:", len(X_train))
print("Test samples:", len(X_test))
```

```
# ✅ 2. Pad sequences to have equal length (50 words per review)
maxlen = 50
X_train = pad_sequences(X_train, maxlen=maxlen, padding='post')
X_test = pad_sequences(X_test, maxlen=maxlen, padding='post')

print("Shape of X_train:", X_train.shape)
print("Shape of X_test:", X_test.shape)

# ✅ 3. Build RNN model
model = Sequential()
model.add(Embedding(input_dim=num_words, output_dim=32,
input_length=maxlen))
model.add(SimpleRNN(32, return_sequences=False))
model.add(Dense(1, activation='sigmoid'))

# ✅ 4. Compile model
model.compile(optimizer='adam', loss='binary_crossentropy',
metrics=['accuracy'])

# ✅ 5. Train the model
history = model.fit(X_train, y_train,
                    epochs=5,
                    batch_size=128,
                    validation_data=(X_test, y_test))

# ✅ 6. Evaluate model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=0)
print("\nTest Loss:", test_loss)
print("Test Accuracy:", test_acc)

# ✅ 7. Plot accuracy vs epochs
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))
plt.plot(history.history['accuracy'], label="Training Accuracy")
plt.plot(history.history['val_accuracy'], label="Validation Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.title("RNN Accuracy on IMDB Sentiment Analysis")
plt.legend()
plt.grid()
plt.show()
```

<table>
<tr>
<td>Output</td>
<td>


```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Training samples: 25000
Test samples: 25000
Shape of X_train: (25000, 50)
Shape of X_test: (25000, 50)
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
196/196 ━━━━━━━━━━━━━━━━━━━━ 8s 28ms/step - accuracy: 0.6345 - loss: 0.6245 - val_accuracy: 0.8014 - val_loss: 0.4411
Epoch 2/5
196/196 ━━━━━━━━━━━━━━━━━━━━ 6s 31ms/step - accuracy: 0.8550 - loss: 0.3534 - val_accuracy: 0.8119 - val_loss: 0.4131
Epoch 3/5
196/196 ━━━━━━━━━━━━━━━━━━━━ 10s 28ms/step - accuracy: 0.9138 - loss: 0.2341 - val_accuracy: 0.7827 - val_loss: 0.4811
Epoch 4/5
196/196 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.9612 - loss: 0.1270 - val_accuracy: 0.7887 - val_loss: 0.5946
Epoch 5/5
196/196 ━━━━━━━━━━━━━━━━━━━━ 4s 23ms/step - accuracy: 0.9864 - loss: 0.0572 - val_accuracy: 0.7767 - val_loss: 0.6848
```



```
Test Loss: 0.6848179697990417
Test Accuracy: 0.7766799926757812
```

</td>
</tr>
<tr>
<td>Conclusion</td>
<td>The RNN model successfully performed sentiment analysis on IMDB movie reviews and achieved good accuracy. The results show that RNNs can effectively capture sequential patterns in text data, making them suitable for natural language processing tasks like sentiment classification.</td>
</tr>
</table>