Ex.No: 5 Roll.No: 210701702

PYTHON PROGRAM TO BUILD AN RNN WITH KERAS

Aim:

To build a RNN(Recurrent Neural Networks) with keras in python.

Procedure:

1. Import NumPy, TensorFlow, IMDB dataset utilities, RNN layers, and Matplotlib for plotting.
2. Set a random seed for reproducibility in NumPy and TensorFlow.
3. Load the IMDB dataset, restricting to the top 10,000 words and specifying a maximum sequence length of 200.
4. Pad sequences to ensure uniform input length across all samples.
5. Build a Sequential model with an Embedding layer, a SimpleRNN layer, and a Dense output layer for binary classification.
6. Compile the model using Adam optimizer and binary cross-entropy loss, with accuracy as the evaluation metric.
7. Print the model summary to check the architecture and layer details.
8. Train the model for 5 epochs with a batch size of 64, including validation data.
9. Evaluate the model on the test data and print the accuracy.
10. Plot training and validation accuracy and loss over epochs using Matplotlib.

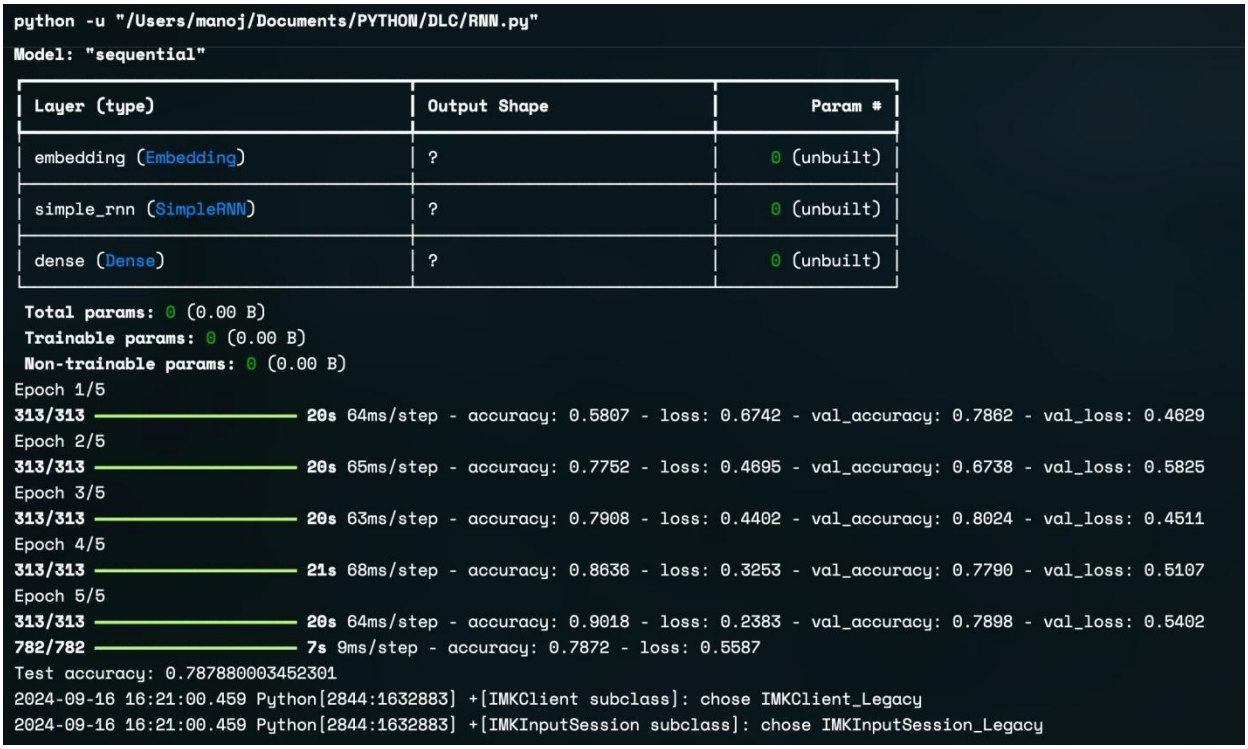
Code:

```
# Import necessary libraries
import numpy as np import
tensorflow as tf
```

```
from tensorflow.keras.datasets import imdb from
tensorflow.keras.preprocessing import sequence from
tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense import
matplotlib.pyplot as plt

# Set random seed for reproducibility
np.random.seed(42) tf.random.set_seed(42)

# Load the IMDB dataset
max_features = 10000 # Number of words to consider as features maxlen = 200
# Maximum length of input sequences

(train_data, train_labels), (test_data, test_labels) =
imdb.load_data(num_words=max_features)

# Pad sequences to ensure consistent input size
train_data = sequence.pad_sequences(train_data, maxlen=maxlen) test_data =
sequence.pad_sequences(test_data, maxlen=maxlen)

# Build the RNN model model =
Sequential()
model.add(Embedding(input_dim=max_features, output_dim=128,
input_length=maxlen))
model.add(SimpleRNN(128, activation='relu')) model.add(Dense(1,
activation='sigmoid'))

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

```
# Print model summary model.summary()

# Train the model
history = model.fit(train_data, train_labels,
                    epochs=5, batch_size=64,
                    validation_split=0.2,
                    verbose=1)

# Evaluate the model
test_loss, test_acc = model.evaluate(test_data, test_labels) print(f'Test
accuracy: {test_acc}')

# Plot training and validation accuracy and loss plt.figure(figsize=(12, 4))

# Accuracy plot plt.subplot(1,
2, 1)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epochs') plt.ylabel('Accuracy') plt.legend()
# Loss plot plt.subplot(1,
2, 2)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs') plt.ylabel('Loss') plt.legend()

plt.show()
```

Output:



Result:

Thus, to build a RNN using Keras in python has been completed successfully.