|  |  |
| --- | --- |
| **Ex No: 9**  **Date: 17th October 2024** | **Recurrent Neural Networks** |

**Objective:** The objective of this lab is to understand and implement a Recurrent Neural Network (RNN) using Python and a suitable deep learning library (e.g., TensorFlow or PyTorch). The focus is on sequence modelling tasks such as time series prediction or text processing.

**Description:**

This lab involves the step-by-step implementation of an RNN. Key processes include data preprocessing, model building, training, and evaluation. The code explanations will cover these steps in detail.

**Building the parts of the algorithm:**

1. **Importing Libraries**

* Import necessary libraries for data manipulation (numpy, pandas), plotting (matplotlib), and deep learning (keras)

1. **Loading and Preprocessing Data**
   * **Purpose:** Load data from a CSV file, select the relevant column, and scale the data to a range of 0 to 1 using Min-Max Scaling. This is crucial for efficient model training.
   * **Expected Output:** The data will be transformed into scaled values, usually between 0 and 1.
2. **Creating Training and Testing Datasets**

* **Purpose:** Split the scaled data into training (80%) and testing (20%) datasets to evaluate the model's performance.
* **Expected Output:** Two datasets: train for model training and test for evaluation.

1. **Converting Data to RNN-Compatible Format**

* **Purpose:** Transform the data into sequences of a specified time step to make it compatible with RNN input requirements.
* **Expected Output:** X\_train, y\_train, X\_test, and y\_test arrays containing input features and targets for training and testing.

1. **Reshaping Input for RNN**

* **Purpose:** Reshape the input data into a three-dimensional format expected by RNNs: [samples, time steps, features].
* **Expected Output:** Reshaped data suitable for feeding into an RNN.

1. **Building the RNN Model**

* **Purpose:** Define an RNN model using a Sequential architecture with one Simple RNN layer of 50 units and a dense layer for output.
* **Expected Output:** Compiled RNN model ready for training.

1. **Training the Model**

* **Purpose:** Train the RNN model on the training data over 50 epochs with a batch size of 32, using validation data to monitor performance.
* **Expected Output:** Training process with loss values per epoch, showing improvement over time.

**8. Model Evaluation and Predictions**

* **Purpose:** Generate predictions on both training and testing data. Predictions are then inverse scaled back to their original range for evaluation.
* **Expected Output:** Predictions for both training and testing data in the original data scale.

**Key Observations:**

* **Data Preparation:** Loading and scaling of data to normalize input values.
* **Sequence Creation:** Transformation of data into time-step sequences for RNN compatibility.
* **Model Building:** SimpleRNN model with one hidden layer and an output layer, optimized using Adam.
* **Training and Evaluation:** Continuous training and validation using Mean Squared Error to minimize prediction errors

**GitHub Link:** [**https://github.com/tulasigr/DeepLearning**](https://github.com/tulasigr/DeepLearning)