Practical No:9

Title:

Implement RNN for sequence labelling

Aim:

To implement a **Recurrent Neural Network (RNN)** for sequence labeling tasks, such as part-of-speech tagging or named entity recognition.

Pre-requisites:

1. Basic understanding of **Recurrent Neural Networks (RNNs)** and their applications in sequence data.
2. Familiarity with Python libraries such as TensorFlow or PyTorch for building neural networks.
3. Knowledge of preprocessing techniques, especially for text data (tokenization, padding).
4. Understanding of sequence labeling tasks and evaluation metrics.

Theory:

Recurrent Neural Networks (RNNs):

RNNs are a class of neural networks designed to process sequential data. Unlike traditional feedforward networks, RNNs have connections that allow information to persist across time steps, making them suitable for tasks where the context of previous elements influences the current prediction (e.g., text).

Applications of RNNs:

- Part-of-Speech (POS) tagging
- Named Entity Recognition (NER)
- Machine translation
- Speech recognition

Sequence Labeling:

Sequence labeling involves assigning a label to each element in a sequence. For instance, in POS tagging, each word in a sentence is assigned a grammatical category (e.g., noun, verb). In NER, entities are tagged with categories like PERSON, ORGANIZATION, or LOCATION.

Architecture:

An RNN consists of an input layer, recurrent hidden layers, and an output layer. The hidden layers maintain a hidden state that is updated at each time step based on the current input and the previous hidden state.

Steps for Implementing RNN for Sequence Labeling:

1. **Data Preparation:**
    - Load and preprocess the dataset (tokenization, padding).
    - Create input-output pairs for the RNN (e.g., words and their corresponding labels).
2. **Build the RNN Model:**
    - Construct the RNN architecture using a framework like TensorFlow or PyTorch.
    - Use techniques like **embedding layers** for word representations.

3. **Training the Model:**
    - Compile the model with an appropriate loss function and optimizer.
    - Train the model on the training data.
4. **Model Evaluation:**
    - Evaluate the model on test data and calculate metrics such as accuracy and F1-score.

Conclusion:

In this assignment, we successfully implemented an RNN for a sequence labeling task such as part-of-speech tagging using a sample dataset. The model was trained using the LSTM architecture, which is an effective variant of RNNs for handling sequences. This approach can be adapted for various sequence labeling tasks, including named entity recognition and other NLP applications.