Program 4

1. **Objective:-** To explore text generation using Recurrent Neural Network [RNN] and understand the impact on different word representation.

2. **Dataset:-** Kaggle dataset of 100 poems in csv format.

3. **Theory:-** Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed for sequential data. Unlike traditional feedforward networks, RNNs have a built-in mechanism to retain information from previous inputs, making them suitable for tasks like text generation, speech recognition, time series prediction, and machine translation.
   An RNN processes an input sequence one step at a time, passing information from one step to the next through a hidden state. The mathematical formulation is as follow:-

$$\mathbf{h_t = f\,(W_h h_{t-1} + W_x x_t + b)}$$
$$\mathbf{y_t = g(W_y h_t + c)}$$

where,

- h_t = hidden state in time t
- x_t = input at time t
- W_h, W_x, W_y = weight matrices
- b, c = bias terms
- f = activation function (typically tanh or ReLU)
- y_t = output at time t
- g = output activation (like softmax for classification)

In this program we have to explore text generation using Recurrent Neural Networks (RNNs) and understand the impact of different word representations:

1. One-Hot Encoding:- One-Hot Encoding (OHE) is a technique used to convert categorical data (such as words or labels) into a numerical format that machine learning models can understand. It represents each unique category as a binary vector, where only one position is "1" (hot) and the rest are "0" (cold).w

2. Trainable Word Embeddings:- Trainable word encoding refers to learning word representations dynamically during model training instead of using static encodings like One-Hot Encoding. In deep learning, this is commonly done using word embeddings, where words are represented as dense vectors in a continuous vector space.Instead of manually assigning word representations, the model learns meaningful representations based on context during training.

4. **Steps to Implement:-**
   Step 1: Preprocessing the Dataset

- Read the dataset of 100 poems.
- Tokenize the text.
- Prepare inputs for both One-Hot Encoding and Trainable Word Embeddings.

Step 2: Model Implementation

- Define an RNN or LSTM model.
- Train separate models using One-Hot Encoding and Word Embedding.

Step 3: Training and Evaluation

- Train both models.
- Compare loss and training time.
- Generate Texts.

5. **Comparison:-** The comparison between both the models can be summarised in the below table:-

Aspect	**One Hot Encoder**	**Trainable Word Embedding**
Loss	Loss decreases at very slow rate.	Loss decreases at faster rate in comparison to OHE.
Efficiency	High dimensionality makes it inefficient.	Model learns the relationship between words making it more efficient.
Overall	Model struggles with long term dependences.	Models is well structured and coherent.

6. **Conclusion:-** We have successfully implemented the RNN model using both the One-Hot Encoding as well as Trainable Word Embedding. **Hence, we conclude that the Trianable Word Embedding is more efficient.**

Github repository