USN NUMBER:1RVU22BSC097
NAME: SHREYA SINHA

Ex No: 8 **Date: 25-09-24**	**Dinosaur Name Generation Using Recurrent Neural Networks (RNN)**

Objective:

The objective of this experiment is to generate new dinosaur names by training a Recurrent Neural Network (RNN) on a text corpus containing names of various dinosaurs. The model will learn character sequences and produce new names based on the patterns observed in the training data.

Introduction:

Recurrent Neural Networks (RNNs) are a type of neural network designed to handle sequential data, such as text or time series. RNNs maintain a hidden state that captures information about previous elements in the sequence, making them ideal for tasks such as text generation, where the next character depends on prior characters. In this lab, we will use an RNN to generate names of fictional dinosaurs after training on a list of real dinosaur names.

Materials and Methods:

 Software:

- Python 3.9
- NumPy for mathematical operations
- utils.py for text preprocessing
- Jupyter Notebook for coding and execution

 Dataset:

- dinos.txt: A text file containing a list of dinosaur names, used as the training data.

Methods:

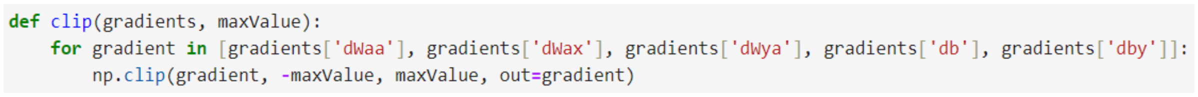
Step 1: Loading and Preprocessing the Data:

- The dataset is read from a text file (dinos.txt) and converted to lowercase.
- The list of unique characters (vocabulary) is extracted from the data, and character-to-index (char_to_ix) and index-to-character (ix_to_char) mappings are created.
- The data is analyzed to understand the number of total and unique characters.

USN NUMBER:1RVU22BSC097
NAME: SHREYA SINHA

Clipping Function:

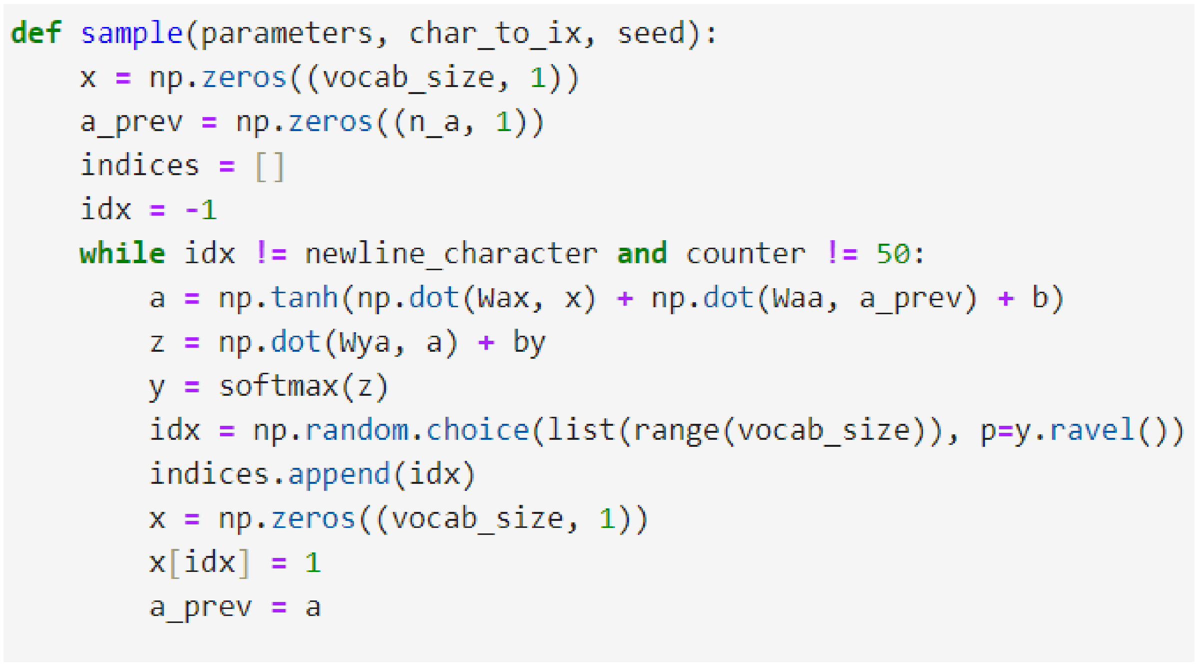
- The function clip (gradients, maxValue) is implemented to clip the gradients between -maxValue and maxValue to avoid the exploding gradient problem.
- This function loops over the gradient values and uses np.clip() to limit their values.



```python
def clip(gradients, maxValue):
    for gradient in [gradients['dWaa'], gradients['dWax'], gradients['dWya'], gradients['db'], gradients['dby']]:
        np.clip(gradient, -maxValue, maxValue, out=gradient)
```

Sampling Function:

- The function sample (parameters, char_to_ix, seed) is implemented to sample a sequence of characters from the output probability distribution of the RNN.
- It generates characters until either the newline character is reached or a maximum of 50 characters are sampled.
- Forward propagation is used at each time step, and the index of the next character is selected based on the output probabilities.



```python
def sample(parameters, char_to_ix, seed):
    x = np.zeros((vocab_size, 1))
    a_prev = np.zeros((n_a, 1))
    indices = []
    idx = -1
    while idx != newline_character and counter != 50:
        a = np.tanh(np.dot(Wax, x) + np.dot(Waa, a_prev) + b)
        z = np.dot(Wya, a) + by
        y = softmax(z)
        idx = np.random.choice(list(range(vocab_size)), p=y.ravel())
        indices.append(idx)
        x = np.zeros((vocab_size, 1))
        x[idx] = 1
        a_prev = a
```

Optimization Function:

- The function optimize(X, Y, a_prev, parameters, learning_rate) performs one step of optimization:
    - Forward propagation computes the loss.
    - Backward propagation computes the gradients.
    - Gradients are clipped and the parameters are updated.

USN NUMBER:1RVU22BSC097
NAME: SHREYA SINHA

Training the Model:

- The function model(data, ix_to_char, char_to_ix, num_iterations, n_a, dino_names, vocab_size, verbose) is the main loop to train the RNN model:
    - It initializes the RNN parameters and hidden state.
    - It loops through a shuffled list of dinosaur names and applies the optimization step.
    - Every 2000 iterations, the model samples new dinosaur names and prints them to monitor progress.

Results:

After training the model for 35,000 iterations, the RNN was able to generate plausible dinosaur names based on the training data.

Initial Training Phase (Iteration 0):

- Loss: 23.087336
- Sampled names were short and lacked coherence, e.g., "on," "nia," and "ay."

Mid Training Phase (Iteration 20,000):

- Loss: 22.786
- The model began generating more coherent names, such as "Sauroster," "Stego," and "Pteranos."

Final Training Phase (Iteration 35,000):

- Loss: 21.519
- By this point, the generated names were significantly more natural, resembling real dinosaur names. Example names:
    - "Dargon"
    - "Branidon"
    - "Pterasaurus"

These results show how the RNN improved its ability to generate names over time as it learned from the training data.

GitHub Link: https://github.com/Shreyasinha7/Lab-8-RNN.git