Text Generator Using PyTorch and Tkinter with Recurrent Neural Network (RNN) Model.

This project demonstrates a character-level text generation model trained on a dataset using a Long Short-Term Memory (LSTM) network. It also provides a graphical user interface (GUI) built with Tkinter for generating new text based on the trained model.

Features

- Character-level text prediction based on input sequences.
- PyTorch-based LSTM model for text generation.
- GUI for user interaction with text length and temperature options.
- Sampling-based text generation.

Code Overview

Libraries

- `torch`, `torch.nn`, `torch.optim`: For building and training the neural network.
- `numpy`: For numerical computations.
- `random`: For sampling random sequences from the text.
- `tkinter`: For creating the graphical user interface (GUI).

Data Preparation

1. **Loading the Dataset**:
   - The text dataset is downloaded from a URL and read as a string.
   - The text is truncated to a specific range to limit size and complexity.

2. **Preprocessing**:
   - Characters in the text are indexed to create mappings between characters and integers.
   - Features (input sequences) and targets (next characters) are prepared with a fixed sequence length (`SEQ_LENGTH`) and step size (`STEP_SIZE`).

3. **One-Hot Encoding**:

  - Each input sequence is converted into a one-hot encoded tensor for training.

  - Targets are similarly converted for computing loss.

Model Definition

- **Architecture**:

  - An `nn.LSTM` layer processes the input sequences.

  - A fully connected (`nn.Linear`) layer maps the LSTM output to the character space.

  - A `Softmax` activation function computes probabilities for each character.

- **Parameters**:

  - `input_size`: The number of unique characters in the dataset.

  - `hidden_size`: The number of hidden units in the LSTM.

  - `output_size`: The number of unique characters (same as `input_size`).

Training the Model

- The training loop iterates over multiple epochs and uses a batch size of 256.

- The training loop iterates over multiple number of hidden size of 128.

- Loss is calculated using `nn.CrossEntropyLoss`.

- The `RMSprop` optimizer is used for parameter updates.

- At each step, the model predicts the next character based on the current input sequence.

Text Generation

1. **Sampling Function**:

- Converts the model's output probabilities into a single character index based on a given `temperature`.

- Higher temperatures result in more random predictions, while lower temperatures make predictions more deterministic.

2. **Generation Logic**:

- A random sequence is chosen as the starting point.

- Characters are generated iteratively based on the model's predictions.

- The generated text is appended to the initial sequence.

Graphical User Interface (GUI)

- Built using `Tkinter`.

- Features:

- Input fields for specifying text length and temperature.

- A "Generate Text" button to trigger text generation.

- A scrollable output area to display the generated text.

- Error handling ensures valid inputs.

How to Use

1. **Run the Script**:

Execute the Python script. The Tkinter GUI will open.

2. **Input Parameters**:

- Enter the desired text length (positive integer).

- Enter the temperature (float between 0 and 2).

3. **Generate Text**:

- Click the "Generate Text" button to generate new text based on the model.

- The generated text will appear in the scrollable output area.

Code Customization

- **Dataset**:

  - Change the `url` to use a different text dataset.
  - Modify `SEQ_LENGTH` and `STEP_SIZE` to adjust sequence preprocessing.

- **Model**:
  - Adjust `hidden_size` for a larger or smaller LSTM model.
  - Tune hyperparameters like `epochs` or learning rate for better performance.

- **GUI**:
  - Add more features, such as saving generated text to a file.

Example Usage

1. Enter `Text Length: 100` and `Temperature: 1.0`.
2. Click "Generate Text".
3. The output might look something like this:

```
and the morrow my soul did see thee
oh the song of the past shall bear it,
yet the heart will still be free.
```

Future Improvements

- Add support for GPU acceleration.
- Enable training on larger datasets.
- Improve the GUI with additional customization options (e.g., saving/loading models).
- Implement beam search or other advanced sampling techniques for text generation.

Dependencies

Ensure the following Python packages are installed:

- torch
- numpy
- tkinter
- random

Install them using:

```
pip install torch
pip install numpy
pip install tkinter
```