RNN-based Poem Generation Code

1. Model Definitions

RNN, LSTM, GRU

What:

- **Recurrent Neural Networks (RNNs)**, **Long Short-Term Memory (LSTM)**, and **Gated Recurrent Units (GRU)** are architectures used for sequence modeling.

Why:

- **RNN:** Basic sequential processing but suffers from the **vanishing gradient problem**, making it difficult to learn long-term dependencies.
- **LSTM/GRU:** Designed to handle long-term dependencies through **gating mechanisms**.
    - **LSTM:** Uses forget, input, and output gates to manage memory.
    - **GRU:** Uses reset and update gates, making it computationally lighter than LSTM.

Advantages:

- LSTM/GRU can handle longer sequences better than vanilla RNN.
- GRU is computationally more efficient than LSTM while performing comparably in many tasks.

Limitations:

- RNNs struggle with **long sequences** despite LSTM/GRU improvements.
- All models are computationally **intensive** for large datasets.

One-Hot vs. Embedding Layers

One-Hot Encoding:

- Converts tokens to sparse vectors of size **vocab_size**.
- **Pros:** Simple implementation for small vocabularies.
- **Cons:** High memory usage; no semantic relationships captured between words.

Embedding Layers:

- Maps tokens to dense vectors of size **embed_size**.
- **Pros:** Captures semantic relationships; reduces dimensionality.

- **Cons:** Requires learning embeddings but is more efficient for large vocabularies.

---

2. Data Processing

Preprocessing:

- **Steps:** Lowercasing, preserving line breaks, separating punctuation, removing special characters, tokenization.
- **Why:** Standardizes text, ensuring consistency and retaining poetic structure (e.g., line breaks).

Vocabulary Creation:

- **Special Tokens:** `<pad>`, `<unk>`, `<sos>`, `<eos>` for **padding**, unknown words, start/end of sequence.
- **Minimum Frequency (min_freq):** Filters rare words to reduce noise and improve training efficiency.

Sequence Creation:

- **Sliding Window:** Generates **input-target pairs** by shifting a fixed-length window across the text.
- **Why:** Ensures sufficient context for training while maximizing data usage.

---

3. Training and Evaluation

Loss Function:

- **CrossEntropyLoss** with `ignore_index=pad_idx` to exclude padding tokens during loss computation.

Gradient Clipping:

- **clip_grad=5.0** prevents exploding gradients, which can occur in deep RNNs.

Early Stopping:

- **Patience = 5** (stops training if validation loss doesn't improve for 5 epochs).
- **Why:** Prevents overfitting and saves computation time.

Perplexity:

- **Formula:** `exp(loss)`
- **Measures model uncertainty** (lower = better).

Learning Rate Scheduler:

- **ReduceLROnPlateau** adjusts learning rate dynamically based on validation loss.

---

4. Text Generation

Sampling Techniques:

- **Temperature Sampling:** Controls randomness in text generation.
  - **Higher temperature (>1.0):** More randomness.
  - **Lower temperature (<1.0):** More deterministic.
- **Top-K Sampling:** Limits token selection to the **top k** highest probability words.
- **Nucleus (Top-P) Sampling:** Chooses tokens from the smallest subset that sums to **top_p probability**.

Why:

- Balances creativity and coherence in generated poems.

Post-Processing:

- Ensures correct spacing around punctuation and line breaks for **better readability**.

---

5. Model Comparison

Metrics:

- **Test Loss / Perplexity:** Measure model generalization ability.
- **Accuracy:** Percentage of correct predictions, excluding padding.
- **Visualization:** Bar plots compare loss, perplexity, and accuracy across different models.

Why:

- Helps **identify the best-performing architecture** (e.g., LSTM with embeddings often outperforms RNN).

---

6. Main Function

Dataset:

- Loads poems from `poems-100.csv` and processes them.

Hyperparameters:

- `embed_size = 256`, `hidden_size = 512`, `num_layers = 2`, `dropout = 0.3`.
- **Why:**
  - **Larger hidden size** helps capture complex patterns.
  - **Dropout** prevents overfitting.

Train/Val/Test Split:

- **80% Train, 10% Validation, 10% Test** ensures fair evaluation.

Key Advantages and Limitations

Advantages:

- **Modular design** (easy to swap models and layers).
- **Supports multiple sampling strategies** for text generation.
- **Early stopping and gradient clipping** stabilize training.

Limitations:

- **One-hot encoding** is memory-heavy for large vocabularies.
- **RNN variants** still struggle with very long sequences.
- **Training time increases** with model complexity (LSTM/GRU slower than RNN).

Example Workflow

Data Flow:

- **Poems → Tokenization → Vocabulary → Sequences → Batches**.

Training:

- All models trained for **100 epochs** with early stopping.

Evaluation:

- Best models saved and compared on the **test set**.

Generation:

- Sample poems generated using **temperature sampling**.