Name: Sai Ram Gunturu
ID: C00313478

Detailed Log of Work: Magic: The Gathering RNN Name Generator

Original Notebook Link - Notebook

Phases	Changes Made	Reason for the change	Duration	Difficulty level (1-10)
Dataset selection	Changed the dataset from "Earthling Names" to "MTG Card Names dataset"	Names dataset sequences and sizes are less, so I thought to go for a dataset that suits a broader size in terms of data and sequence.	30 min	6
Algorithm Version	Used a Baseline RNN and a Two Layer RNN	One layer can focus on structure and other can focus on spellings, punctuations and to get more meaningful data.	2 hours	8
Further Enhancements	Trained with Baseline model from the original Notebook	To test how new dataset work with the existing model	4 hours	9
	Hyper parameters changed	Because of the below par results from the baseline model, I have changed the Rnn_num_units, embedding_size and dropout to improve the model performance		
	Added a Second RNN Layer	Improved results, but still having issues with gibberish words, So I added one more layer. One layer will take care of patterns, and one will take care of the words		

Conclusions		With the added layer, the model performs better than the previous single layer model but still struggling with gibberish words. Moving forward I will try to improve the model with the LSTM Algorithm which is specifically good for long sequence sentences.	40 min	6

1. Business Understanding

Objective: To build a character-level RNN that generates Magic: The Gathering (MTG) card lines (e.g., "[mana cost] Creature, Name: Some Effect").

Success Criteria:

- The model should output mana costs in brackets, followed by a reasonable type (e.g., "Creature," "Enchantment," "Instant") and a name.
- Lower training/validation loss and more coherent text indicate success.

Key Changes/Reasoning:

- **Dataset Replacement**:
  - From: An original dataset of ~8k "names."
  - To: mtg_card_names.txt, with ~16k lines of MTG entries.
  - Reason: Closely aligned with personal interest and provide a more complex structure for the RNN to learn.

2. Data Understanding

Dataset:

- Every line contains bracketed mana cost(s) (e.g., "[1G]", "[2U]"), card type(s), and card names.
- Max sequence length of 169 characters, requiring significant padding.
- 79 unique tokens - including special characters and brackets.

Key Changes/Reasoning:

- **Token Range**:

    - Expanded the allowable token check from 50–60 to up to 300 because MTG lines have more symbols.
- **Practical Impact**:
    - Larger token set → The model needs to learn a broader vocabulary.
- **Analysis**:
    - The resulting dataset is broader inducing more complexity in training but yields richer generation possibilities.

3. Data Preparation

Preprocessing:

- Added a start token " " to each line.
- Used a pad token "#" to ensure uniform length (169).
- Created a to_matrix(...) function that converts lines to a numeric matrix.

Key Changes/Reasoning:

- **Start Token**: Helps the model to identify the beginning of a sequence.
- **Pad Token**: Maintains consistent batch shapes.
- **Analysis**:
    - Proper padding ensures each sequence is of the same length for training, which is crucial for RNN unrolling.

4. Modeling

4.1 Single-Layer Rnn

- **Architecture**:
    - Embedding → RNN (64 hidden units) → Dense (softmax).
    - Loss: Cross-entropy on next-character predictions.
- **Training**:
    - Initially 1,000 steps, batch size 32.
    - Observed rapid drop in loss (~7 down to ~2), then plateau around ~1–0.5.

Practical Impact & Analysis:

- Outputs:
    - Learned bracket usage and some structure but generated gibberish words.
    - 64 hidden units may be insufficient for the complexity of MTG lines.
- Improvements Considered:
    - More training steps, larger hidden state, dropout, and a second layer.

4.2 Larger Hidden State & Dropout

- **Architecture Changes**:
    - Hidden units: from 64 → 128.
    - Dropout: ~0.3 added after the hidden layer.
- **Training Steps**: 5,000–10,000 to allow deeper learning.

Practical Impact & Analysis:

- Loss: Decreased further (e.g., ~0.3 range).
- Outputs: Showed more coherent bracketed costs and card types, though still with invented words.
- Reasoning: Dropout prevents overfitting, while a larger hidden state captures more nuanced patterns.

4.3 Two-Layer Rnn

- **Architecture**:
    - Two Dense-based RNN layers, each with 128 units, plus dropout at each layer.
    - Stacked to capture local patterns in the first layer and higher-level dependencies in the second.

Practical Impact & Analysis:

- Loss: Could reach ~0.28–0.3 after 5,000 steps.
- Outputs: More consistent "[mana cost] type: name" lines, fewer random tokens, but still some nonsense words.
- Conclusion: Stacking improved structure, but the model still benefits from further refinements or a more advanced RNN cell (e.g., LSTM).

5. Evaluation

1. **Loss Curves**:
    - Monitored training loss, which started high (~7) and gradually dropped to below 0.5 for the single-layer and ~0.28 for the two-layer model.
2. **Sample Outputs**:
    - Without Prefix: Bracketed costs like "[3WBB]", "[4G]", "[1GG]" and types such as "Creature" or "Enchantment," plus a random name (e.g., "Favumane Glame").
    - With Prefix "[2U]": The model continues from "[2U]", generating lines like "[2U] Instant: Chizinifr" or "[2U] Creature, Kof Bhe: Magat Ries".

Analysis:

- Improved Format: The network reliably includes bracketed mana costs and a card type.
- Invented Words: Names are partially coherent but often contain nonsense syllables.
- Next Steps: Additional training or a more powerful architecture (LSTM) may result in more natural names.

6. Deployment

- **Minimal Deployment**:
  - A function generate_sample enables text generation with an optional prefix (e.g., "[2U]").
  - The user can enter a seed phrase at runtime to prompt the model in the same session.

Further Plans:

- Serialization: Potentially save model checkpoints with tf.train.Saver (TF1.x) for reuse.
- Web/App: Could integrate a simple web or CLI tool to generate lines on demand.

7. Conclusion & Future Work

- **Model Progress**:
  - Transitioning from a single-layer RNN to a two-layer RNN with dropout and a larger hidden state improved structural consistency of generated lines.
- **Limitations**:
  - Invented or nonsensical words persist, meaning the model needs either more capacity, more training steps, or an advanced RNN cell.
- **Next Notebook**: Implement LSTM
  - Why: LSTM gating mechanisms handle long-range dependencies better than vanilla RNNs, typically reducing nonsense tokens and improving coherence.
  - Goal: Compare training loss, validation metrics, and generation quality against the two-layer vanilla RNN to further refine MTG card line generation.

In summary, these logs record how changes in the dataset, architecture, and hyperparameters affect performance of the model. Overall, the two-layer RNN shows promising structural gains, and LSTM is expected to address the remaining gibberish text in future work.