Name: Sai Ram Gunturu
ID: C00313478

Note: This is a completely new notebook created by me

LSTM Notebook Project Log

Phases	Techniques Used	Reason for the Technique Used	Duration	Difficulty level (1-10)
Dataset selection	Used the same "MTG Card Names dataset"	RNN created some structured output but a lot of gibberish words. To improve the prediction, I have used the same dataset	5 min	5
Algorithm Version	Single LSTM Layer, Two Layer LSTM (Stacked),	One layer can focus on structure and other can focus on spellings, punctuations and to get more meaningful data.	3 hours	8
Further Enhancements	Trained with Baseline LSTM Model	To test how the dataset work with the existing model	6 hours	9
	Used 2 Layer stacked LSTM without iterative feedback approach	The results were completely gibberish. So I moved to "2 Layer LSTM without iterative feedback approach" to focus on first pattern of the data i.e., [ ]		
	Used 2 Layer stacked LSTM with iterative feedback approach	"Without iterative feedback approach" failed drastically. I used "With iterative feedback approach" for longer sequences and for better training of the model		

	Added Validation step and early stopping criteria with increased training steps	Improved structure of the results, but took a lot of time, to reduce this I added a validation step and early stopping criteria with increased steps		
	Hyperparameter Search and training the model with best hyperparameters	Although using early stopping criteria saved a lot of time and improved the performance of the model, to fine tune the model I tried hyperparameter search		
Conclusions		With these model refinements I reduced the validation loss from 28% (In RNN) to 22% in LSTM. Through this I have learned that fine tuning the model and training for longer steps will play an important role that leads to a better prediction	1 Hour	6

1. Business Understanding

The goal of this notebook is to generate high-quality Magic: The Gathering (MTG) card lines which follow a certain format (e.g., "[mana cost] Creature, Name: Effect"). In my earlier RNN experiments I was able to pick up the overall structure, bracketed mana costs and card types but it still suffered from gibberish and truncated tokens. To address these shortcomings, I leaned towards an LSTM-based approach as its gating mechanisms are much better at retaining long-range dependencies

2. Data Understanding & Preparation

I continue to use the same dataset (mtg_card_names.txt) with approximately 16,000 MTG entries. Each entry contains bracketed mana costs, card types, and card names, with a fixed maximum sequence length of 169 characters and a vocabulary of 79 unique tokens.

- Preprocessing involves adding a start token (a space) and a pad token ("#") such that every sequence is exactly 169 characters long.
- I then converted the text lines into padded numeric matrices using a custom to_matrix function, ensuring that all inputs have a consistent shape for training.

3. Modeling & Training the LSTM

3.1 Approach Overview: Switching from Vanilla RNN to LSTM

In my earlier experiments with vanilla RNNs, I observed that the model can generate bracketed mana costs and card types, but its outputs were often ruined by gibberish and incomplete tokens. To solve these issues, I switched over to an LSTM model. LSTM cells with their gating mechanisms, are much more effective at capturing long-range dependencies. To avoid confusion with the previous RNN code, I created a new computational graph and ran a new session just for the LSTM model.

3.2 Stacked 2-Layer LSTM: Implementation and Results

For the LSTM model, I implemented a two-layer (stacked) architecture using modern Keras layers in TF1.x mode. Here's what I did:

- I created a placeholder with [None, 169] shape for passing input token IDs and reused my embedding layer to convert these tokens into dense vectors. Through my initial experiments, I later tuned the embedding size to 32.
- Next, I stacked 2 LSTM layers, each with 256 units, with return_sequences=True. The first LSTM layer was responsible for capturing local patterns (punctuation and bracket placement) and the second layer learned higher-level dependencies (like card type transitions and naming conventions).
- A last Dense layer mapped those from LSTM outputs to the vocabulary size (n_tokens), resulting in logits for next-token prediction.
- For training, I shifted the target sequence by one token and calculated the sparse softmax cross-entropy loss over all time steps. I used the Adam optimizer with gradient clipping to optimize the loss.
- To generate text, I implemented an iterative, token-by-token generation method. Instead of using a single-pass generation approach (that produced truncated outputs), I maintained a fixed input, which is an array of shape (1, 169) and iteratively fed each newly generated token back into the model until I reached a desired sequence length (e.g., 60 tokens).

Hyperparameter Rationale:

The structure of MTG card lines is so complex. My initial experiments suggested that using a larger embedding size and more LSTM units improved the model's capacity to capture the complex structure.

Based on these findings, I decided on an embedding size of 32 and 256 LSTM units per layer. With these hyperparameter choices, the loss was reduced and produced more coherent outputs compared to smaller configurations. As a result, these values were used for the later training and adjustments.

3.3 Finding Best Hyperparameters

I conducted a random hyperparameter search by varying the embedding size and LSTM unit count while keeping dropout and learning rate constant. For each trial, I trained the model for fewer steps (around 1,000) and monitored the validation loss. Through this process, I discovered that the combination of an embedding size of 32 and 256 LSTM units created a significantly lower validation loss, which suggested a better ability to capture complex patterns of the dataset.

3.4 Training with Best Hyperparameters

After identifying the optimal configuration (embedding_size=32, lstm_units=256), I then trained the final stacked LSTM model for 8,000 steps with early stopping based on validation loss. This extended training allowed the model to make its predictions better, resulting in a final training loss of approximately 0.18 and a validation loss around 0.22. Next, to sample full-length MTG card lines, I used an iterative generation method.

3.5 Final Results Analysis

The final model now produces more coherent MTG card lines. For example:

- **No Prefix Samples:**
  - "[None] Creature, Horsy Skeleton: Suntreast of the Nedrel"
  - "[2] Artifact Creature, Construct: Sbillew Welder"
  - "[5G] Creature, Efred: Skull Calamiy"
  - "[3U] Creature, Dragon: Torc Wizar"
  - "[5R] Creature, Beast: Axob Couzery"

- **With [2U] Prefix Samples:**
  - "[2U] Instant: E."
  - "[2U] Artalt, Eldrazi :riftskrouk Mawler"

- “[2U] Instant: Aetherinds”
- “[2U] Instant: Hinders”
- “[2U] Instant: Pulson of Kit”

These outputs consistently feature bracketed mana costs and recognizable card types, although some tokens still appear partially nonsensical. This means that while the LSTM has greatly improved the overall structure, further training and fine-tuning could still enhance output coherence.

4. Evaluation

The stacked LSTM model is evaluated which shows a steady decline in both training and validation losses over 8,000 steps, with final losses around 0.18 (training) and 0.22 (validation). The generated samples shows that the model is now consistently producing bracketed mana costs and card types. Although sometimes there are nonsensical tokens, the improvements in structure and overall coherence are significant compared to previous RNN models.

5. Conclusion

By changing to a stacked LSTM with an iterative generation approach the structural coherence of the generated MTG card lines has greatly improved. The hyperparameter tuning discovered an embedding size of 32 and LSTM units of 256 per layer as the optimal configuration. After further training, the final model had a training loss of ~0.18 and a validation loss of ~0.22. While the outputs now consistently include proper bracketed mana costs and card types, further refinements such as extended training and additional fine-tuning are required to fully eliminate residual gibberish. This log reflects my methodology, results, and the rationale for the next steps in this project.