LSTM Modeling for Book Review Text Prediction: A Char-RNN approach on Goodreads review samples

Ella Bartt

Abstract

By using a Long Short-Term Memory model to improve upon a Character-level Recurrent Neural Networks (Char RNN), one can emulate Goodreads book reviews of romance books. By varying the hidden layer size and number of hidden layers, we can change the complexity and effectiveness of the model. By further subsetting to one- and five-star book reviews, we see a difference in the model's effectiveness.

1. Introduction

Many readily available examples of projects using character-level text prediction using RNNs use text samples from one or a few authors. However, with online book reviews, that is not the case. Book reviewers on Goodreads come from many backgrounds, all with different expectations and opinions of often similar books. The goal of this project is to train multiple LSTM models using Goodreads book reviews and analyze how they perform given text samples from different authors to detect text patterns.

Given two subsets of the same dataset of Goodreads romance reviews, 1-star and 5-star reviews, comparing the training losses and text samples gives a good idea of the difference in text patterns based on the reviewer's opinions. The model will predict character sequences better from five star reviews than one star reviews, as the loss will be lower and the model will converge faster. I predict that overall, 5-star reviews will have a more predictable, uniform structure given the clear expectations of a good romance book from most readers.

2. Related Works

Wan et al. The dataset for this work comes from Wan et al. in UC San Diego's Computer Science and Engineering Department (Wan & McAuley, 2018; Wan et al., 2019). The data was scraped from Goodreads and includes information for over 1,378,033 English book reviews, across 25,475 books and 8 genres. Some books overlap across genres. In "Fine-Grained Spoiler Detection from Large-Scale Review Corpora," Wan et al. propose a new neural network, SpoilerNet, to categorize whether certain book reviews include "spoiler" info for the book using word popularity and uniqueness (Wan et al., 2019). This is based on CNN and RNN approaches, yet expanded to better predict unconventional character sequences. In another paper by the same lab, "Item Recommendation on Monotonic Behavior Chains," they propose an algorithm to score user preferences and recommendation algorithms based on book reviews and ratings (Wan & McAuley, 2018).

Karpathy. The proposed use of Char-RNNs in this project is based on Andrej Karpathy's work and the Tiny Shakespeare Dataset (Karpathy). In this project, Karpathy uses a basic Char RNN to generate blocks of Shakespeare-like text. Adapted from Andrej Karpathy's work, the blog from Kaggle utilizes an Long Short-term memory framework to better detect text patterns in a large dataset (noa, a). The Kaggle dataset analyzes the effectiveness of an LSTM for Leo Tolstoy's Anna Karenina. While incredibly effective, the text data used to train the data differs greatly from that of the Goodreads dataset, as it follows one story and author throughout. Regardless, the latter model better fits the data and converges to a much lower loss in the following experiments.

3. Methodology

3.1. Design

Recurrent Neural Networks are a robust and flexible structure for character-level text prediction, allowing for recurrent connections and dependencies between characters. However, given the vanishing gradient problem of traditional Recurrent Neural Networks, a Long Short-term Memory approach becomes necessary for more robust character-level text prediction. Therefore, I will train multiple LSTM models to see which one performs best.

First, the dataset chosen provides two unique challenges:

1. Different authors for each review leads to slightly different writing styles and, therefore, different possible text predictions, and

2. Variability in genre and length.

In order to address some of these challenges, I first subset to the romance genre out of all Goodreads reviews. I did this because each genre audience may have different expectations, and romance specifically has certain words, terms, and structures that readers may use to review each book. For example, romance books usually have two main protagonists and a happy ending. Readers look for original characters and chemistry. Next, since the romance subset was already quite large, I subset further to English-language reviews between 100 and 500 characters (see attached code).

As further investigation into the differences between character LSTMs, I used the same methods to subset 1-star and 5-star romance reviews lists of the same size (39479 reviews, randomly sampled). The goal is to then compare the models between one and five star reviews with varying hidden layer sizes and quantities to get the best model. These models will be compared by looking at their loss functions and randomly generated sample text.

Originally, when using a regular Character-level Recurrent Neural Network (as provided in Homework 3 Bonus Task 1) on the sample dataset without batching or a dropout rate, the model converged very poorly. The loss stagnated around 2 for all text reviews (using cross entropy loss) after 20000 iterations. The LSTM model, adapted from a basic Character RNN and LSTM codes on Github (Andrej 2026) and Kaggle (Char-RNN n.d.), utilized short-term memory and a dropout probability of 0.5, and the model converged at a much lower loss for each test. Additionally, by using batch testing as outlined in the code for such a large dataset, the model trains much faster and more effectively.

3.2. Definitions

In order to define the LSTM network mathematically, we first define:

- x_t: input at time t (letter, represented by the one-hot encoder)
- h_t: hidden state at step t (r_out)
- C_t: Cell state, a vector of possible character outcomes (in long-term memory)
- σ: sigmoid activation function
- W, b: learned weights and bases at each step

Given the above values, we run the training steps for the LSTM as follows:

The dropout gate: To train the model, we define

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_i)$$

to determine which information of C_{t-1} we want to "forget." The dropout probability used in these models is 0.5.

Input information: We define the input information (one-hot encoded text vectors) as i_t and the new candidate values (letters) $\tilde{C}_t$ for our cell state at t (before forgetting):

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i),$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Updated Cell state: We can then update the next cell state (the vector of predicted next values / letters) given f_t, i_t, and $\tilde{C}_t$:

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Hidden layer and output: The output, o_t is the version of C_t after forgetting, which we then put through the hidden layer:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t).$$

Finally, the output and hidden state are used to map back to the size of our possible character outputs using $W_{fc} \cdot h_t + b_{fc}$. This allows us to predict the next character effectively.

4. Experiment

First, I tested the model by batching the reviews in batches of 128 samples of length 100 characters. This batch size allowed the model to train much faster and to run on multiple different reviews at a time. The dropout rate was set to 0.5 to avoid overfitting to training data, which led to relatively equal training and validation losses across all models. All models ran for 20 epochs.

The first model I ran, Model 0, had 2 layers and a hidden layer of size 512. The second model, Model 1, had 2 layers and a hidden layer of size 1024. The third model, Model 2, had 4 layers and a hidden layer of size 512, and the last model Model 3, had 4 layers and a hidden layer of size 1024.

By looking at the loss every tenth iteration on the validation batch, and then the final 100 values (the converged loss), we can see the average loss that each model converged to in table Table [1] above.

As we can see, in general, the 5-star reviews converged faster and had a lower loss. The models with larger hidden layers (1024) had very similar losses, 0.964 and 0.975 for 2 and 4 layers, respectively. The difference between these two models for the 1-star dataset was negligible. Therefore, the model with only two layers but the largest hidden layer performed the best. For the sake of sampling, we choose this model to analyze going forward.

When we choose the best model, with 2 hidden layers and a hidden size of 1024, we can see how the model performs

Table 1. Average Loss after 20 epochs across all tested models, between 1- and 5-star datasets. *(N_layers = number of hidden layers, N_hidden = size of hidden layers.)*

N_LAYERS	N_HIDDEN	1-STAR	5-STAR
2	512	1.056	1.018
	1024	1.014	0.964
4	512	1.104	1.049
	1024	1.013	0.975

differently between 1- and 5-star book reviews in Figure [1]. Here, one can clearly observe the lower loss for the 5-star dataset. This pattern holds across models, as shown in the table, and the difference between training steps for 5-star reviews is clearly shown in Figure [2]. Here, one can additionally see that the model we have selected for sampling not only has the lowest loss, but converged the fastest out of all of them.

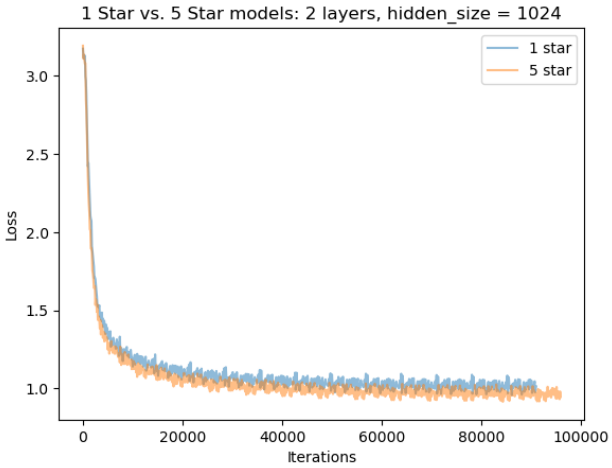

Figure 1. Model training loss for 1- vs. 5-star book reviews. 2 hidden layers with a hidden layer size of 1024. Cross Entropy Loss.

After choosing the best models (two separate ones, each with the same parameters, but trained on two separate datasets), one can initialize a sample with the characters 'This' and of length 300 characters and obtain the following texts:

- **1-star**: *This is a romance but this book just isn't for me. I couldn't get past the first chapters. I feel like I was reading the next book of the worst reading. But nothing more than one of them, this isn't a book I can't read because it is a bunch of a crap from a run or page of a book and I didn't enjoy it,*
- **5-star**: *This book is amazing. I love this series, and this one did not disappoint. This story has a small thing*

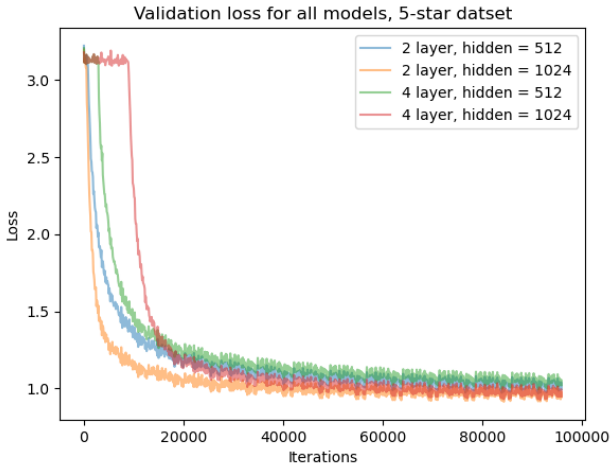

Figure 2. Model training loss for 5-star book reviews, showing convergence rates.

from the first page to the very last sentence. I love this author and this story has some suspense and texps. A story and a great story which I am so impressed. Let me start this series, the book's so

As we can see, the above texts are robust, relatively understandable, and follow the structure of a basic book review. The samples contain genre-specific word patterns, such as "romance." The sentiment is also clearly different in each sample, showing longer text patterns and differences. The 1-star sample has more negative contractions ("isn't", "couldn't", "didn't"), while the 5-star sample has fewer contractions overall. This shows that most 5-star reviews, while they may contain "not" ("did not disappoint"), separate out contractions to further emphasize positive sentiment, a pattern which the model picked up on.

5. Conclusion

For character-level text prediction, the Long Short-term Memory approach is an effective and robust machine learning algorithm that can be applied to diverse types of textual data. In this analysis, I built upon previous Char-RNN and LSTM work for author texts and analyzed how the LSTM performs for Goodreads romance book reviews. The book reviews provide a unique challenge for the ML algorithm given diverse authorship and book themes. When subset to only English-language book reviews for romance books and divided between 1- and 5-star reviews, the model converged each time to a low loss and effective prediction method. Each model performed slightly worse for 1-star reviews than 5-star reviews, showing more uniform and predictable text and word patterns for positive reviews. The model with

2 layers and a larger hidden layer converged quickest and, when sampled, generated legible and relevant text to emulate an actual book review. While the method could be expanded to a Transformer-based character-level text predictor, the robustness of the non-transformer based models tested prove an adequate and less computationally expensive alternative.

References

Char-RNN, a. URL https://kaggle.com/code/ashukr/char-rnn.

Goodreads Datasets, b. URL https://cseweb.ucsd.edu/~jmcauley/datasets/goodreads.html.

What is LSTM - Long Short Term Memory?, c. URL https://www.geeksforgeeks.org/deep-learning/deep-learning-introduction-to-long-short-term-memory/. Section: Deep Learning.

Long short-term memory, March 2026a. URL https://en.wikipedia.org/w/index.php?title=Long_short-term_memory&oldid=1342441618. Page Version ID: 1342441618.

Recurrent neural network, March 2026b. URL https://en.wikipedia.org/w/index.php?title=Recurrent_neural_network&oldid=1342698264. Page Version ID: 1342698264.

Brownlee, J. Text Generation With LSTM Recurrent Neural Networks in Python with Keras, August 2016. URL https://www.machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/.

Karpathy, A. The Unreasonable Effectiveness of Recurrent Neural Networks. URL https://karpathy.github.io/2015/05/21/rnn-effectiveness/.

Robertson, S. spro/char-rnn.pytorch, March 2026. URL https://github.com/spro/char-rnn.pytorch. original-date: 2017-03-19T04:48:52Z.

Wan, M. and McAuley, J. Item recommendation on monotonic behavior chains. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pp. 86–94, Vancouver British Columbia Canada, September 2018. ACM. ISBN 978-1-4503-5901-6. doi: 10.1145/3240323.3240369. URL https://dl.acm.org/doi/10.1145/3240323.3240369.

Wan, M., Misra, R., Nakashole, N., and McAuley, J. Fine-Grained Spoiler Detection from Large-Scale Review Corpora. In Korhonen, A., Traum, D., and Màrquez, L. (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 2605–2610, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1248. URL https://aclanthology.org/P19-1248/.