COGS 185 Project Report

Syed A. Usmani
Cognitive Science Department
UCSD
susmani@ucsd.edu

Abstract

This project explores the use of character-level recurrent neural networks (Char-RNNs) to model and generate natural language text. Unlike word-level models, Char-RNNs operate at the level of individual characters, allowing them to learn syntax, structure, and stylistic patterns from raw text without the need for tokenization or a fixed vocabulary. I experimented with a multi-layer LSTM-based Char-RNN using PyTorch and train it on a "War and Peace" by Leo Tolstory and a collection of John Stuart Mill's essays, learning to predict the next character in a sequence. After training, the models are capable of generating new text in the style of the original corpus, producing coherent sentences, names, and punctuation structures while following grammar rules. The system is flexible and can be applied to any sufficiently large plain-text dataset, such as literature, code, or poetry. This work demonstrates the surprising capacity of simple sequence models to capture complex linguistic structures when trained at the character level.

Introduction

Character-level language models provide a unique approach to text generation by learning directly from raw sequences of characters. Unlike word-based models, Char-RNNs require no tokenization and can learn spelling, grammar, punctuation, and stylistic elements from scratch. This makes them particularly well-suited to modeling complex, stylistically rich, or historically significant texts.

In this project, I train and evaluate character-level recurrent neural networks (Char-RNNs) on two distinct corpora: "War and Peace" by Leo Tolstoy, and a curated collection of philosophical and political essays by John Stuart Mill, including "On Liberty", "Utilitarianism", "The Subjection of Women", "Socialism", "On Representative Government", and "Principles of Political Economy." These texts offer contrasting linguistic styles (narrative fiction versus formal philosophical discourse) providing an opportunity to explore how Char-RNNs adapt to different textual domains.

Using a multilayer-layer LSTM architecture implemented in PyTorch, I train each model to predict the next character in a sequence. Once trained, the models are used to generate new text in the learned style of their respective corpora. The results illustrate the capacity of simple neural architectures to internalize complex language patterns, and reveal insights into the language and style of two influential literary and philosophical sources.

Method

Data Preprocessing

Two distinct text corpora were obtained from Project Gutenberg in UTF-8 format to train character-level language models: Leo Tolstoy's "War and Peace" (3,201,634 characters, 105 unique characters) and a collection of works by John Stuart Mill (3,156,593 characters, 137 unique characters). Non-content sections such as licensing information and footnotes were manually removed. The texts

Preprint. Under review.

were processed at the character level, with each unique character mapped to a numerical index to create character-to-index (char2idx) and index-to-character (idx2char) dictionaries. A custom CharDataset class was implemented to generate training sequences of fixed length (100 characters) with corresponding target sequences offset by one position, enabling next-character prediction. The dataset was configured to return input-target pairs where the target sequence represents the ground truth for the next character at each position.

Model Architecture

The character-level LSTM model (CharLSTM) consisted of three main components:

- Embedding Layer: Maps character indices to dense vector representations of configurable dimensionality
- LSTM Layers: Multi-layer LSTM with dropout regularization (dropout=0.3) for sequence modeling
- Output Layer: Fully connected layer projecting LSTM hidden states to vocabulary-sized logits, followed by log-softmax activation for probability distribution over characters

The model architecture enables variable-length text generation by maintaining hidden states across sequential character predictions.

Training Procedure

Models were trained using the following configuration:

- Loss Function: Negative Log-Likelihood Loss (NLLLoss)
- Optimizer: Adam optimizer with learning rate of 7×10^{-4}
- Learning Rate Scheduling: ReduceLROnPlateau scheduler (patience=3, factor=0.5)
- Mixed Precision Training: CUDA automatic mixed precision (AMP) with gradient scaling for computational efficiency
- Batch Processing: DataLoader with shuffling and consistent batch sizes

Training was conducted for multiple epochs with progress monitoring via tqdm.

Hyperparameter Configurations

Three distinct hyperparameter configurations were evaluated:

Configuration 1 (High Complexity):

- Sequence length: 100 characters
- Batch size: 1,024
- Hidden size: 512
- Number of LSTM layers: 4
- Embedding size: 256
- Training epochs: 20

Configuration 2 (Low-Medium Complexity):

- Sequence length: 100 characters
- Batch size: 512
- Hidden size: 128
- Number of LSTM layers: 2
- Embedding size: 64
- Training epochs: 20

Configuration 3 (Deep Architecture):

- Sequence length: 100 characters
- Batch size: 256
- Hidden size: 256
- Number of LSTM layers: 8
- Embedding size: 128
- Training epochs: 5

Text Generation and Evaluation

Model performance was assessed through qualitative text generation using temperature-controlled sampling. A custom generation function (`generate_text_during_train`) was implemented to produce text samples during training at the end of every epoch, enabling real-time assessment of model learning progression.

Generation employed multinomial sampling from the softmax probability distribution over the vocabulary, with temperature scaling (temperature=1.0) to control randomness. Models were evaluated using consistent seed phrases ("I am learning to speak!" and "My opinion on cats is") to enable comparative assessment across different configurations and training stages.

Training progress was monitored through both quantitative loss metrics and qualitative assessment of generated text coherence, grammar, and stylistic consistency with the training corpus.

Experiment

Experimental Design

Four distinct experiments were conducted to evaluate the performance of character-level LSTM models across different architectural configurations and text corpora. The experiments were designed to investigate: (1) the impact of model complexity on learning dynamics, (2) the effect of dataset characteristics on model performance, and (3) the trade-offs between model depth and width.

Experiment 1: High-Complexity Model on War and Peace

The first experiment employed the high-complexity configuration (4 layers, 512 hidden units, 256 embedding dimensions) trained on Tolstoy's "War and Peace" for 20 epochs with a batch size of 1,024.

Results: The model demonstrated rapid convergence from an initial loss of 4.65 to approximately 0.87 by the final epoch. Text generation quality improved dramatically over training:

- Initial generation: Produced random character sequences with no linguistic structure
- Mid-training (Epoch 10): Generated coherent sentences with proper punctuation and character names from the source text
- Final generation: Produced stylistically consistent text resembling Tolstoy's prose, including appropriate dialogue formatting and narrative structure

Sample final output: "I am learning to speak! I will tell you, my dear boy—I don't understand that something was in somebody!" said Nicholas.

Experiment 2: Reduced-Complexity Model on War and Peace

A second experiment used a reduced-complexity architecture (2 layers, 128 hidden units, 64 embedding dimensions) on the same dataset to assess the impact of model capacity on performance.

Results: This configuration showed slower convergence, maintaining higher loss values (final average loss: 2.61 vs 0.87 in Experiment 1). Generated text exhibited:

- Less coherent long-range dependencies
- More frequent grammatical errors
- Reduced consistency in character names and dialogue structure
- Faster training time per epoch but inferior text quality

Sample final output: I am learning to speak!”

“Why, I’ve

was in bguoth to them?” And

quite time he did he not be happening in the sound of

in h.

Experiment 3: High-Complexity Model on J.S. Mill Corpus

The third experiment applied the high-complexity configuration to John Stuart Mill’s philosophical writings, training for 5 epochs due to computational constraints.

Results: The model achieved superior convergence compared to the fiction corpus, reaching a final loss of 0.55.

Generated text demonstrated:

- Rapid acquisition of philosophical vocabulary and concepts
- Proper formatting of academic prose including indentation and section breaks
- Coherent argumentation structure typical of Mill’s writing style
- Integration of domain-specific terminology (e.g., "despotism," "liberty," "equality")

Sample output: "My opinion on cats is a true belief that Germany remain under the study of equality. This remarkable condition of society, it is because it is a selfish object..."

Experiment 4: Narrower-But-Deeper Architecture Model on J.S. Mill Corpus

The final experiment employed a deep but narrow architecture (8 layers, 256 hidden units, 128 embedding dimensions) on the Mill corpus to investigate depth vs. width trade-offs.

Results: This configuration achieved comparable loss values (final: 1.14) but with different learning characteristics:

- More stable training dynamics with less variation between epochs
- Generated text showed good local coherence but occasional long-range inconsistencies
- Faster convergence in early epochs but plateau effect in later training
- Computational efficiency improved due to reduced hidden dimensions

Sample output: "My opinion on cats is an equality to their spirits, and silver, except both on the amount, not so that no notions of the produce of banks, and he has avrigured to persons..."

Cross-Experiment Analysis

Dataset Effects: The philosophical corpus (Mill) demonstrated faster convergence and better final performance compared to the narrative corpus (War and Peace), likely due to:

- More consistent vocabulary and sentence structure
- Repetitive argumentative patterns
- Formal academic writing style with less stylistic variation

Architectural Trade-offs:

- High-capacity models (Experiment 1) produced the most coherent long-form text generation
- Reduced-capacity models (Experiment 2) showed acceptable performance for shorter sequences but degraded quality in extended generation
- Deep architectures (Experiment 4) balanced computational efficiency with reasonable performance, but still failed to produce text as coherent as what was found in Experiment 1

Conclusion

This study systematically investigated the performance of character-level LSTM models across different architectural configurations and text corpora, revealing several key insights into neural language modeling at the character level.

Key Findings

Model Capacity and Performance: My experiments demonstrate a clear relationship between model complexity and text generation quality. The high-complexity configuration (4 layers, 512 hidden units) consistently outperformed reduced-capacity alternatives, achieving significantly lower loss values (0.87 vs 2.61) and generating more coherent, stylistically consistent text. This suggests that character-level language modeling benefits substantially from increased model capacity, particularly for capturing long-range dependencies and complex linguistic patterns.

Corpus Characteristics Impact Learning: A particularly striking finding was the superior performance achieved on the philosophical corpus (J.S. Mill) compared to the narrative text (War and Peace). The Mill corpus enabled faster convergence (0.55 final loss in 5 epochs vs 0.87 in 20 epochs) and produced more coherent generated text. This performance differential likely stems from the more structured, repetitive nature of academic prose compared to the varied narrative styles, dialogue, and character interactions present in literary fiction.

Architectural Trade-offs: The comparison between high-capacity wide models and deep narrow architectures revealed nuanced trade-offs. While the 8-layer deep model achieved reasonable performance with improved computational efficiency, it exhibited plateau effects and occasional long-range coherence issues. This suggests that for character-level modeling, width (hidden dimension size) may be more critical than depth for optimal performance.

Implications

These findings have important implications for practical applications of character-level language models. For applications requiring high-quality text generation, my results suggest prioritizing model capacity over computational efficiency. The superior performance on philosophical text suggests that character-level models may be particularly well-suited for domain-specific applications in academic writing, technical documentation, or other structured text types where consistency and formal language patterns are prevalent.

Limitations and Future Work

The computational constraints limited the exploration of even larger model architectures that might yield further improvements. To expand on this in the future I would: (1) conduct systematic investigation of optimal sequence lengths for different text types, (2) compare performance with transformer-based character-level models, (3) explore hybrid approaches combining character and word-level representations, and (4) develop of more sophisticated sampling strategies for text generation.

References

Karpathy, A. (2015, May 21). The Unreasonable Effectiveness of Recurrent Neural Networks. Andrej Karpathy blog. Retrieved from http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Karpathy, A. (2015). char-rnn: Multi-layer Recurrent Neural Networks (LSTM, GRU, RNN) for character-level language models in Torch [Computer software]. GitHub. https://github.com/karpathy/char-rnn

Project Gutenberg. (n.d.). Free eBooks - Project Gutenberg. Retrieved from https://www.gutenberg.org/

Tolstoy, L. (1869). War and Peace. Project Gutenberg. Retrieved from https://www.gutenberg.org/

Mill, J. S. The Collected Works of John Stuart Mill. Project Gutenberg. Retrieved from https://www.gutenberg.org/