

# Assignment 2 Report: Neural Language Model Training (PyTorch)

## 1. Introduction

The objective of this assignment was to implement a neural language model from scratch using PyTorch and to evaluate how different model capacities affect generalization. The task involved preprocessing the provided dataset, building a character-level vocabulary, training an LSTM model, and comparing underfitting, best-fit, and overfitting behaviors using training/validation loss and perplexity.

Language modeling is a core task in Natural Language Processing. Character-level models predict one character at a time based on previous context, making them suitable for sequence learning and understanding long-term dependencies.

## 2. Dataset & Preprocessing

The dataset provided for the assignment was *Pride and Prejudice* by Jane Austen. The raw text contained metadata and license information from Project Gutenberg, which was removed during cleaning.

### Preprocessing steps:

- Converted text to lowercase
- Removed Gutenberg headers/footers
- Extracted only the main story (beginning with “it is a truth universally acknowledged”)
- Saved cleaned text to `data/Pride_and_Prejudice_CLEANED.txt`

A character-level vocabulary was created by extracting all unique characters. The text was then encoded into integers using `char_to_idx` and `idx_to_char` mappings.

To create training samples, sliding windows of length 100 characters were used:

- **Input (X):** 100-character sequence
- **Target (Y):** the next character

A 90/10 train-validation split was used to evaluate generalization.

## 3. Model Architecture

The model implemented was a simple character-level LSTM, built entirely from scratch without any high-level language modeling libraries.

### Architecture:

- **Embedding Layer:** converts character IDs to vector representations
- **LSTM Layer:** processes sequences and learns dependencies
- **Fully-Connected Layer:** outputs probability distribution over next characters

Embedding → LSTM → Linear → Softmax

The model predicts the next character using cross-entropy loss. Perplexity (PPL) was used as the main evaluation metric.

## 4. Experiments

To demonstrate understanding of model capacity and generalization, three separate experiments were conducted:

### 4.1 Underfitting Model

#### Configuration:

- Embedding: 16
- Hidden size: 32
- Epochs: 3

#### Behavior:

The model was too small to learn meaningful patterns. Both training and validation losses remained high, and the perplexity stayed high as well. This is expected from an under-parameterized model.

### 4.2 Best-Fit Model

#### Configuration:

- Embedding: 64
- Hidden size: 128
- Epochs: 5

#### Behavior:

This model offered the best balance between learning capacity and generalization. Training and validation losses decreased smoothly, and both curves remained close. The final validation perplexity (~3.7) was the best among all models.

This configuration is selected as the **best-fit** model for the assignment.

## 4.3 Overfitting Model

### Configuration:

- Embedding: 128
- Hidden size: 256
- Tiny subset (1000 sequences) to force overfitting
- Learning rate: 0.01
- Epochs: 7

### Behavior:

The model quickly memorized the small training subset. Training loss decreased sharply, but validation loss started increasing after a few epochs. Perplexity worsened, indicating poor generalization—classic overfitting behavior.

## 5. Results

### Perplexity Scores

#### Model    Perplexity

Underfit   Very high (~50+)

Best-Fit   ~3.7

Overfit   ~15–18

### Loss Curves

All plots were saved as PNG images under `plots/`:

- `underfit_loss.png`
- `bestfit_loss.png`
- `overfit_loss.png`

These plots visually confirm the expected behavior of each model type.

## 6. Observations

1. **Underfitting** occurs when the model is too small to capture the data's complexity.
2. **Overfitting** happens when model capacity is too high relative to dataset size.
3. **Best-fit** models show stable validation loss and lowest perplexity.

4. Character-level LSTMs learn slowly but can effectively model stylistic patterns over time.
5. Perplexity is a useful metric for comparing generalization across models.

## 7. Conclusion

This assignment demonstrated how model architecture and dataset size influence learning dynamics in language modeling. By implementing an LSTM from scratch and running controlled experiments, I gained practical understanding of:

- Sequence modeling
- Training vs. validation behavior
- Underfitting vs. overfitting
- Perplexity as a quantitative measure
- The importance of model capacity

The best-fit configuration achieved the lowest perplexity, while underfit and overfit models behaved exactly as expected. All assignment deliverables plots, models, cleaned dataset, and code—are included in the repository.