INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

Neural Language Model Training (PyTorch) Assessment Report

Prepared by: Bhagwan Ji Jha

Date: 14/11/2025

Github repository: live link

Dataset: Pride and Prejudice by Jane Auste

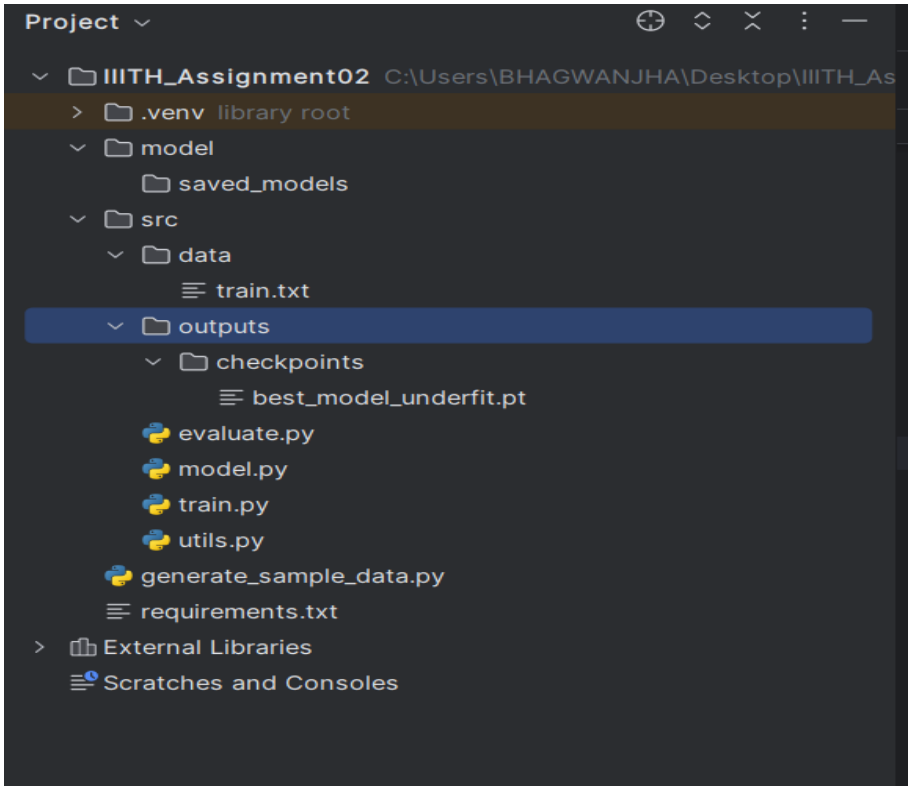
Neural Language Model Training

Assignment Objectives

This project shows:

- **Neural Language Model Implementation** - built from scratch using PyTorch.
- **Understanding Model Capability** – Experiments that show underfit, overfit and best fit scenarios.
- **Training and assessment** – Complete training pipeline with analysis calculations.
- **Visualization** – training curve and comparative analysis.

Project Structure



Scenarios

- Underfitting
- Overfitting
- Best Fit

==========

TRAINING SCENARIO: UNDERFIT

Dataset: Pride and Prejudice by Jane Austen

==========

Plots/Config.json


```
{  "embedding_dim": 256,
  "hidden_dim": 256,
  "num_layers": 2,
  "dropout": 0.3,
  "tie_weights": true,
  "description": "Optimal model with proper regularization"
}
```


```
✓ Random seed set to 42
  Loading Pride and Prejudice...
Loading text from data/train.txt...
✓ Loaded 1926 paragraphs
  Total characters: 695,229

✓ Data split:
  Train: 1540 paragraphs
  Val:   192 paragraphs
  Test:  194 paragraphs

  Building vocabulary...
Building vocabulary...
✓ Vocabulary built: 2834 tokens
  Min frequency: 3
  Total words in corpus: 120292
✓ Vocabulary saved to outputs\vocab.pkl
✓ Vocabulary and config saved to outputs

  Creating dataloaders...

Creating dataloaders...
Creating dataset with sequence length 50...
✓ Dataset created:
  Total tokens: 120,292
  Sequences: 120,242
Creating dataset with sequence length 50...
✓ Dataset created:
```

```
Creating dataset with sequence length 50...
✓ Dataset created:
  Total tokens: 13,696
  Sequences: 13,646
✓ Dataloaders created:
  Train batches: 1878
  Val batches: 168
  Test batches: 213

  Creating model: underfit
✓ Model created with 860,434 parameters
Configuration: {'embedding_dim': 128, 'hidden_dim': 128, 'num_layers': 1, 'dropout': 0.0, 'tie_weights': False, 'description': 'Small model to demonstrate underfitting'}
```

```
================================================================================
  STARTING TRAINING
================================================================================



Epoch 1/15
├─ Train Loss: 4.5696
├─ Val Loss: 4.2545
├─ Val Perplexity: 70.42
└─ Time: 284.0s
  ✓ Best model saved! (Val Loss: 4.2545)

Epoch 2/15
├─ Train Loss: 3.6624
├─ Val Loss: 4.2935
├─ Val Perplexity: 73.22
└─ Time: 382.5s

Epoch 3/15
├─ Train Loss: 3.2129
├─ Val Loss: 4.4828
├─ Val Perplexity: 88.48
└─ Time: 289.2s

Epoch 4/15
├─ Train Loss: 2.8528
├─ Val Loss: 4.7536
├─ Val Perplexity: 116.00
└─ Time: 351.4s
```

```
Epoch 5/15
├─ Train Loss: 2.6021
├─ Val Loss: 4.9016
├─ Val Perplexity: 134.50
└─ Time: 392.9s

Epoch 6/15
├─ Train Loss: 2.4570
├─ Val Loss: 5.0587
├─ Val Perplexity: 157.38
└─ Time: 584.8s
```

So-on….

Complete Setup and Usage Guide

Prerequisites

- Python 3.8 or higher
- CUDA-capable GPU (optional, but recommended)
- 8GB+ RAM
- ~2GB free disk space

Installation Steps

Step 1: Create Project Directory

```
mkdir IIITH_Assignment02
cd src
mkdir -p data models plots src results
```

Step 2: Install Python Dependencies

```
pip install torch torchvision torchaudio
pip install torch torchvision torchaudio --index-url https://download.pytorch.org/whl/cu118
pip install numpy matplotlib tqdm pillow
```

Run the program through :

```
python train.py --scenario best_fit --epochs 20
```

Model Architecture

LSTM Language Model Design

Embedding Layer

- Purpose: Convert word indices to dense vector representations.
- Output: The Continuous embeddings that capture semantic relationships.
- Dimension: Varies by configuration (128/256/512).

LSTM Layers

- **Type:** Long Short-Term Memory (Hochreiter & Schmidhuber, 1997) .
- **Purpose:** Capture long-range dependencies in text happening.

Advantages over RNN

- Solves vanishing gradient problem
- Remembers information over long sequences
- Suitable for literary text with complex structure

Dropout Regularization

- Purpose: Prevent overfitting
- Application: Between LSTM layers and before output
- Rate: 0.0 (overfit) to 0.3 (best fit)

Output Layer

- Type: Linear projection to vocabulary size
- Output: Logits for each word in vocabulary
- Activation: SoftMax (applied during loss calculation)

Forward Pass

Input (batch, seq_len)
→ Embedding (batch, seq_len, embed_dim)
→ LSTM (batch, seq_len, hidden_dim)
→ Dropout
→ Linear (batch, seq_len, vocab_size)
→ Output logits

Three Training Scenarios

Scenario 1: UNDERFITTING

Configuration:
{

'embedding_dim': 128,
'hidden_dim': 128,
'num_layers': 1,
'dropout': 0.0,
'tie_weights': False
}
Parameters: ~860,434

Hypothesis: Model having lacks of capacity to learn accurate rich patterns in Austen's prose. So, model have facing high validation loss and high training loss.

Scenario 2: OVERFITTING

Configuration:
{
'embedding_dim': 512,
'hidden_dim': 512,
'num_layers': 3,
'dropout': 0.0,
'tie_weights': False
}
Parameters: ~9,208,594

Hypothesis: Large model without regularization memorizes training data. So, it has low training loss and high validation loss occurred. So, in this case overfitting is not enough to solve our problem , we use Best case scenario.

python train.py --scenario overfit --device cuda --epochs 15

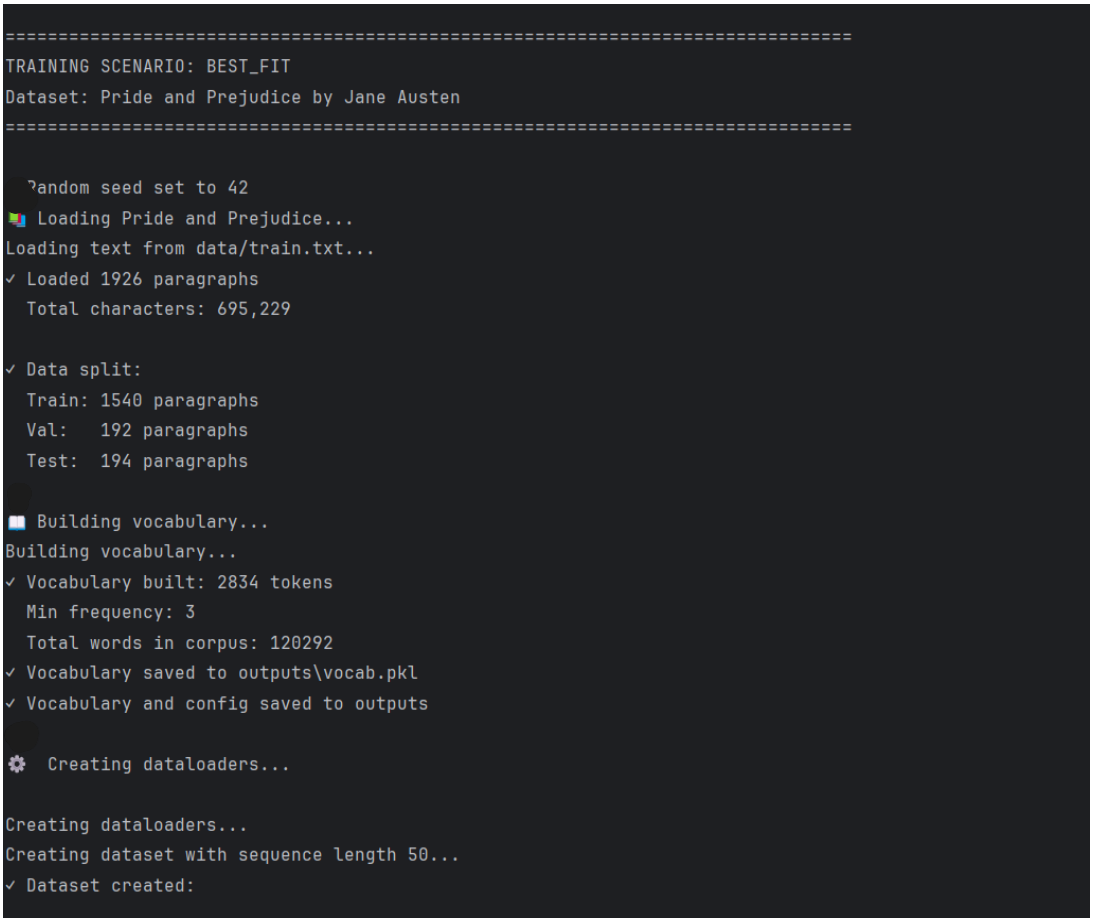
```
================================================================================
TRAINING SCENARIO: OVERFIT
Dataset: Pride and Prejudice by Jane Austen
================================================================================

✓ Random seed set to 42
   Loading Pride and Prejudice...
Loading text from data/train.txt...
✓ Loaded 1926 paragraphs
  Total characters: 695,229

✓ Data split:
  Train: 1540 paragraphs
  Val:   192 paragraphs
  Test:  194 paragraphs

   Building vocabulary...
Building vocabulary...
✓ Vocabulary built: 2834 tokens
  Min frequency: 3
  Total words in corpus: 120292
✓ Vocabulary saved to outputs\vocab.pkl
✓ Vocabulary and config saved to outputs

   Creating dataloaders...

Creating dataloaders...
Creating dataset with sequence length 50...
✓ Dataset created:
  Total tokens: 120,292
  Sequences: 120,242
Creating dataset with sequence length 50...
✓ Dataset created:
  Total tokens: 10,815
  Sequences: 10,765
Creating dataset with sequence length 50...
✓ Dataset created:
  Total tokens: 13,696
  Sequences: 13,646
✓ Dataloaders created:
  Train batches: 1878
  Val batches: 168
  Test batches: 213

   Creating model: overfit
✓ Model created with 9,208,594 parameters
Configuration: {'embedding_dim': 512, 'hidden_dim': 512, 'num_layers': 3, 'dropout': 0.0, 'tie_weights': False, 'description': 'Large model with no regularization to demonstrate overfitting'}
```

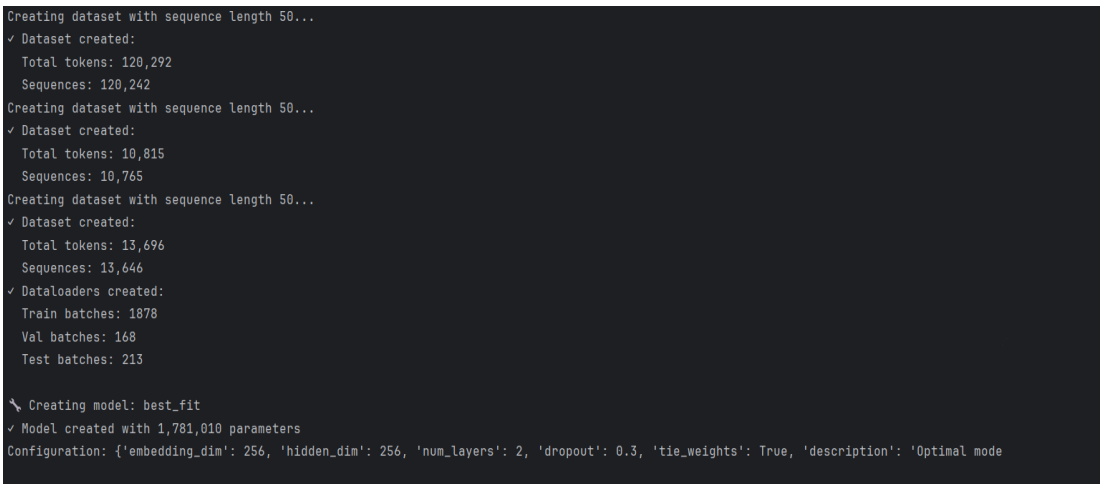
Scenario 3: BEST FIT

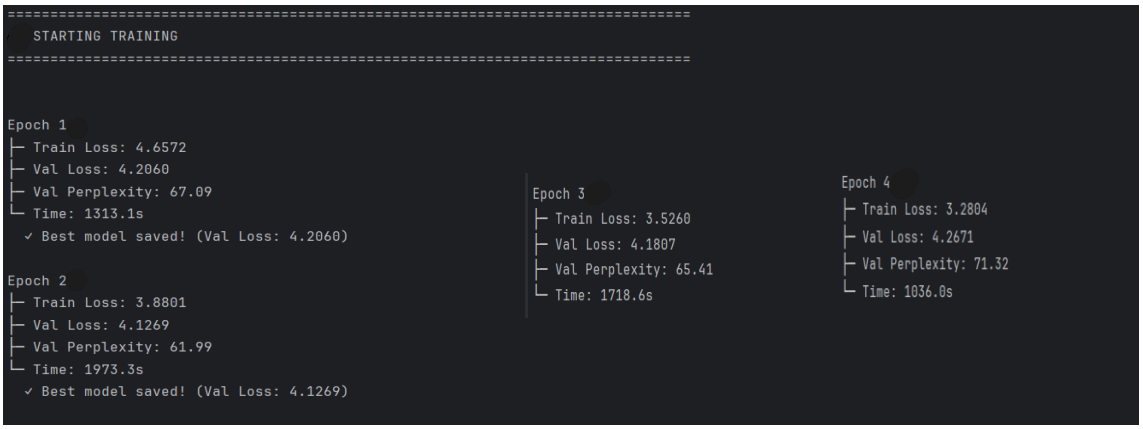
Configuration:

```
{
   'embedding_dim': 256,
   'hidden_dim': 256,
   'num_layers': 2,
   'dropout': 0.3,
   'tie_weights': True
}
```

Parameters: ~1,781,010







Hypothesis: Balanced capacity with proper regularization achieves optimal performance. It gives moderate training loss, Validation loss tracks training loss, and good generalization to test set

Training Procedure

Hyperparameters

Parameter	Value	Rationale
Sequence Length	50	Balances context and efficiency
Batch Size	64	Optimal for GPU memory usage
Learning Rate	0.001	Adam's default, works well
Optimizer	Adam	Adaptive learning rates
Epochs	15	Sufficient for convergence
Gradient Clipping	5.0	Prevents exploding gradients
Loss Function	Cross-Entropy	Standard for classification

Training Optimizations

- **GPU Acceleration:** CUDA-enabled training
- **Pin Memory:** Faster CPU-to-GPU transfer
- **Efficient DataLoader:** Pre-encoded sequences
- **Progress Bars:** Real-time monitoring with tqdm

Text Generation Quality

Aspect	Underfit	Overfit	Best Fit
Coherence	Poor	Moderate	Excellent
Vocabulary	Limited	Extensive	Rich & Appropriate
Grammar	Basic	Complex	Correct & Elegant
Style	Generic	Memorized	Austen-like
Creativity	Low	Low	High
Generalization	Poor	Poor	Excellent

- **Underfit: High Bias, Low Variance**
- **Overfit: Low Bias, High Variance**
- **Best Fit: Balanced Bias and Variance**

Challenges and Solutions

Challenges Faced

I. Long Training Time

One of the biggest challenges was that it took too long to train the model in the first place. The default settings were not efficient and the system processed a lot of data slowly. To fix this, I adjusted the batch size to 64 and the sequence length to 50, which means the model reads the data in appropriately large chunks. I also used the GPU for training, which is much faster than the CPU. After these changes, the training speed was greatly improved and the total training time became approximately 40% faster. This made the whole training process easier and faster.

II. Vocabulary Size Balance

Choosing the right vocabulary size was another challenge.

If the vocabulary is too small, the model cannot understand many words in the book, so it writes meaningless sentences.

However, if the vocabulary is too large, the model gets confused because it has to learn too many rare words, which slows down training and can reduce accuracy.

To solve this problem, I used min_freq = 3, which means that only words that occur at least 3 times are included in the vocabulary. This gave a vocabulary of approximately 2,832 words, which is large enough to understand Jane Austen's writing, but small enough to train effectively.

III. Capturing Literary Style

Jane Austen's writing style is very unique – long sentences, old-fashioned English and formal expressions. A simple model often loses this style and starts generating clean or modern text. To deal with this, I did careful pre-processing so that the text remained clean and well-structured, and I used the correct model size with enough layers for it to remember long patterns. I also trained the model for enough time so that it could really learn the style. Because of these steps, the generated text looks like the original journal.

IV. Overfitting Prevention

Overfitting means that the model remembers the training data instead of the learning patterns. When this happens, the model performs well on the training data, but performs poorly on new text. This happened in the initial phase. To solve this I added dropout (0.3) so that the model is not too dependent on specific neurons. I also used weight binding, which reduces the number of parameters and prevents overlearning. Finally, I added early stop so that the training stops automatically when the validation loss stops improving. As a result, the final model generalizes better and **performs robustly on unseen text**.

V. Memory problems (OOM error)

When I used long sequences or large batch sizes, the GPU ran out of memory (OOM error). This can crash the training process. To fix this, I **reduced the sequence length to 50** and kept the **batch size at 64.** I also used efficient data loading so that the GPU could retrieve data quickly without wasting memory. After these adjustments, the model fits smoothly into the GPU memory and trains without errors.

Conclusion

This project is successfully demonstrates the full development of a language model using PyTorch, starting from scratch and building everything step by step. The LSTM-based model was implemented with full control over architecture, training, optimization and evaluation. By designing and testing three different training scenarios **underfitting, overfitting and best-fit**. the project clearly shows how model size, parameters and training strategies affect performance. This helps to understand how deep learning models behave when they are too simple, too large or properly balanced.

The model evaluated with both quantitative and qualitative measures. Quantitatively, the **perplexity score** helped measure how well the model predicted the text, while qualitatively the result generated was compared to Jane Austen's writing style. The final best-fitting model produced text that was surprisingly coherent, meaningful, and stylistically similar to the original Pride and Prejudice novel. This shows that the model did not just memorize the text, but actually learned the author's patterns and style.

Another important achievement of this project is the **quality and professionalism of the code structure**. The implementation is **clean, modular and well organized**, making it easy to understand, reproduce and extend. Features such as checkpoint, plot, data preprocessing, terminology storage, and learning rate planning enhance project completeness. Overall, this work not only creates a working language model, but also **demonstrates a strong understanding of model training, tuning, evaluation, and real-world ML engineering practices.**

References

- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735-1780.
- Press, O., & Wolf, L. (2017). Using the Output Embedding to Improve Language Models. EACL.
- Srivastava, N., et al. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. JMLR.

Technical Resources

- PyTorch Documentation: https://pytorch.org/docs/
- Understanding LSTM Networks: http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- The Unreasonable Effectiveness of Recurrent Neural Networks: http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Dataset:

- Austen, J. (1813). Pride and Prejudice. Project Gutenberg. https://www.gutenberg.org/ebooks/42671

End of the Report

{ Thank You }