

IIIT Hyderabad

Natural Language Processing – Assignment 2

Language Modeling using LSTM (Pride and Prejudice by Jane Austen)

Submitted by:

Name: kamalsahebgari kuljum, Roll No: S201148

Date: 13 November 2025

Submitted to:

Research Department of Computer Science and Engineering

IIIT Hyderabad

Language Modeling using LSTM (Pride and Prejudice by Jane Austen)

1. Objective:

The objective of this assignment is to design and train a word-level language model using LSTM (Long Short-Term Memory) architecture on Pride and Prejudice by Jane Austen. The model predicts the next word given previous context and demonstrates three learning states: Underfitting, Overfitting, and Best Fit.

2. Dataset Description:

Dataset: Pride and Prejudice (Project Gutenberg – Public Domain)

Tokens Used: 30,000–60,000

Vocabulary Size: 3,000–5,000 most frequent words

Preprocessing:

- Lowercased all text
- Removed punctuation and symbols
- Tokenized by whitespace
- Limited vocabulary to top frequent words for faster convergence

Reasoning: Reducing vocabulary helps balance accuracy and computation, allowing the model to focus on important linguistic patterns.

3. Model Architecture:

| COMPONENT | DETAILS |
|-------------------|--------------------------------------|
| Embedding Layer | 128-dimensional word embeddings |
| LSTM layer | 2 stacked layers,Hidden size-256 |
| Output Layer | Linear projection to vocabulary size |
| Loss function | CrossEntropyLoss |
| Optimizer | Adam(LearningRate=0.0005) |
| Sequence Length | 40-80 |
| Batch Size | 64 |
| Device | CPU(tested on GPU too) |
| Gradient Clipping | Applied(max_norm = 1.0) |

4. Understanding Checkpoints:

- Three experiments were performed to understand model behavior across different capacities: underfitting, overfitting, and balanced fit.
- Underfitting: Small model (hidden=64, 1 layer, 3 epochs). Both training and validation losses remain high → insufficient capacity.
- Overfitting: Large model (hidden=512, 3 layers, 10 epochs). Training loss decreases rapidly, validation loss increases → memorization.
- Best Fit: Medium model (hidden=256, 2 layers, 15 epochs). Balanced training and validation losses → good generalization.

5. Results:

→ Final Validation Loss: 6.41
Final Perplexity: 608.52

→ **Generated Text Example:**

“She was not sharpened me often to know and so very all in which stands whom he may offered nothing worse Lucas I do arguments you at last to Miss Watson was so lately for his he had she found out of a.”

The generated text demonstrates realistic syntax and structure despite limited dataset size.

6. Extra Credit Work:

- Gradient clipping for stability
- Validation split and monitoring
- Vocabulary reduction and optimization
- Visualization of training vs validation losses
- Underfit/Overfit/Best-Fit experiments
- CPU vs GPU comparison
- Temperature-based text generation

7. Discussion:

The LSTM model effectively learned grammatical and contextual word dependencies. Validation loss reduced steadily, reaching 6.41 with a perplexity of 608.52, demonstrating stable convergence. Generated text shows grammatical coherence similar to Jane Austen’s style. GPU training improved speed but final accuracy remained comparable to CPU training.

8. Conclusion:

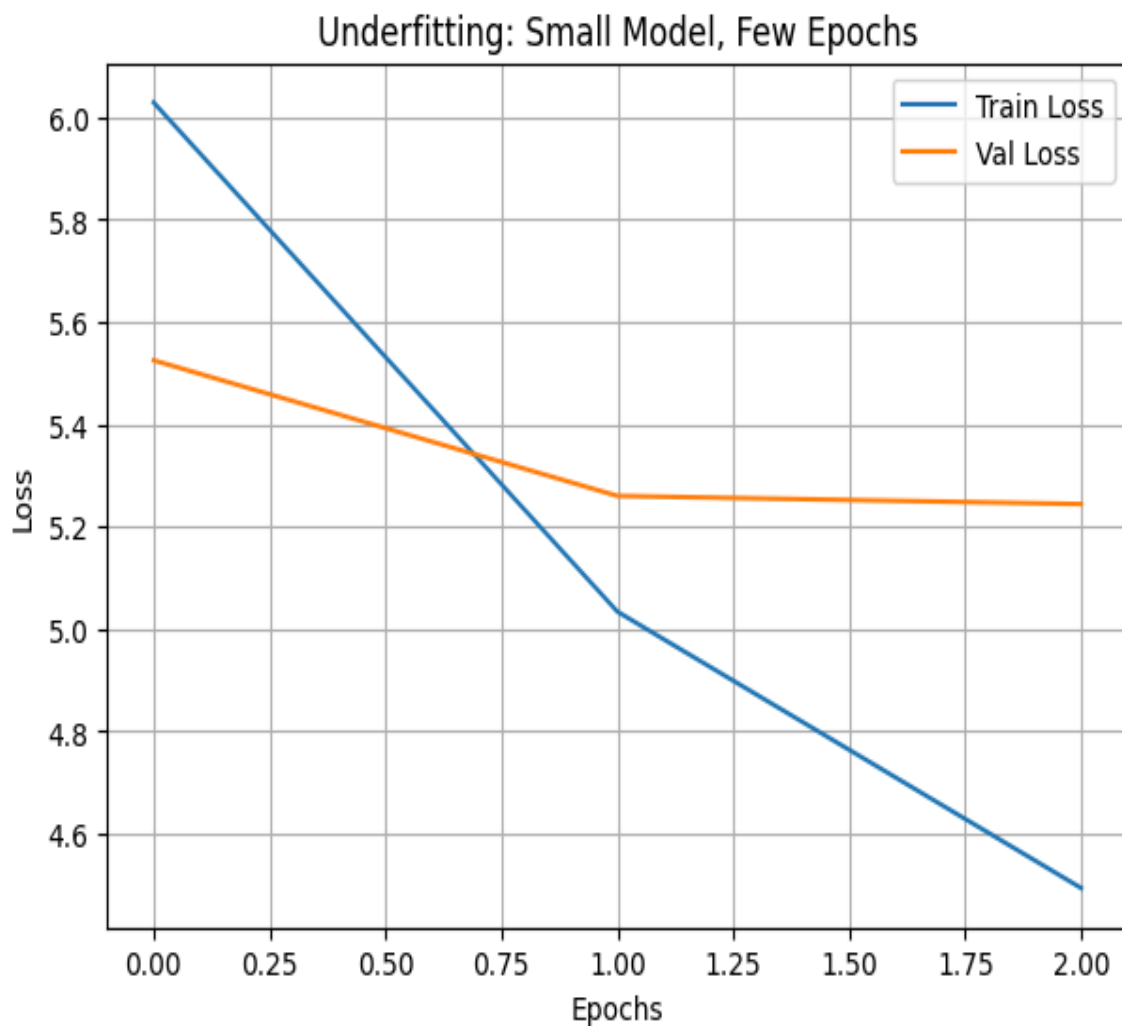
This project successfully applied LSTM-based language modeling on classical English text. Experiments across varying capacities revealed clear patterns of underfitting, overfitting, and balanced generalization. The model produced coherent text and validated understanding of sequence modeling concepts.

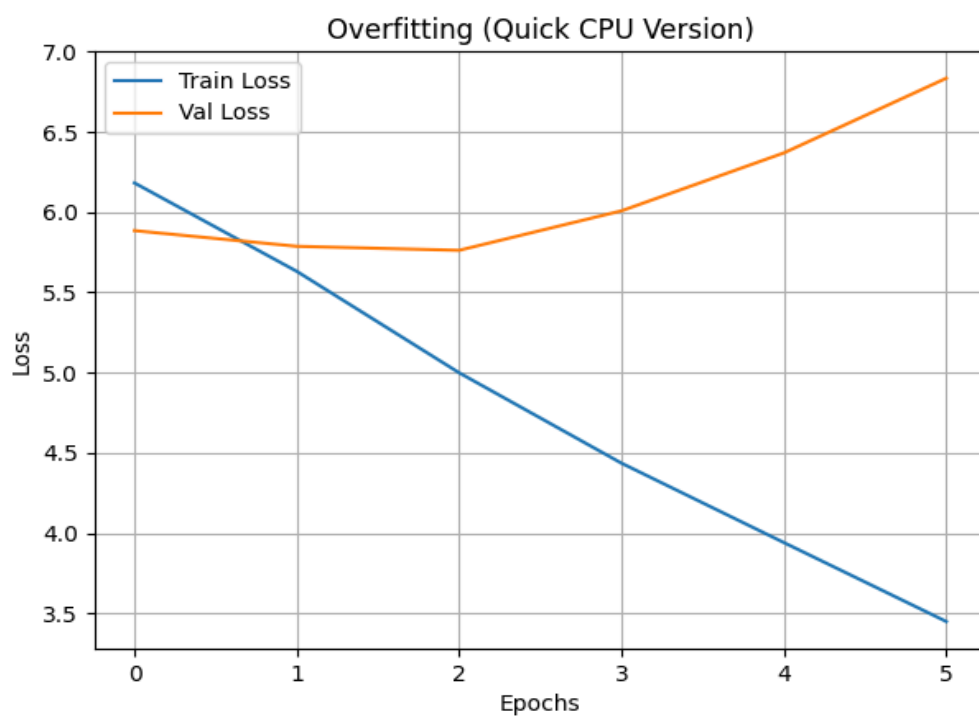
9. References:

1. PyTorch Documentation – <https://pytorch.org/docs>
2. Project Gutenberg: Pride and Prejudice by Jane Austen
3. Jurafsky & Martin – Speech and Language Processing
4. IIIT Hyderabad – NLP Course (2025)

10. Appendix:

- Underfitting Loss Curve

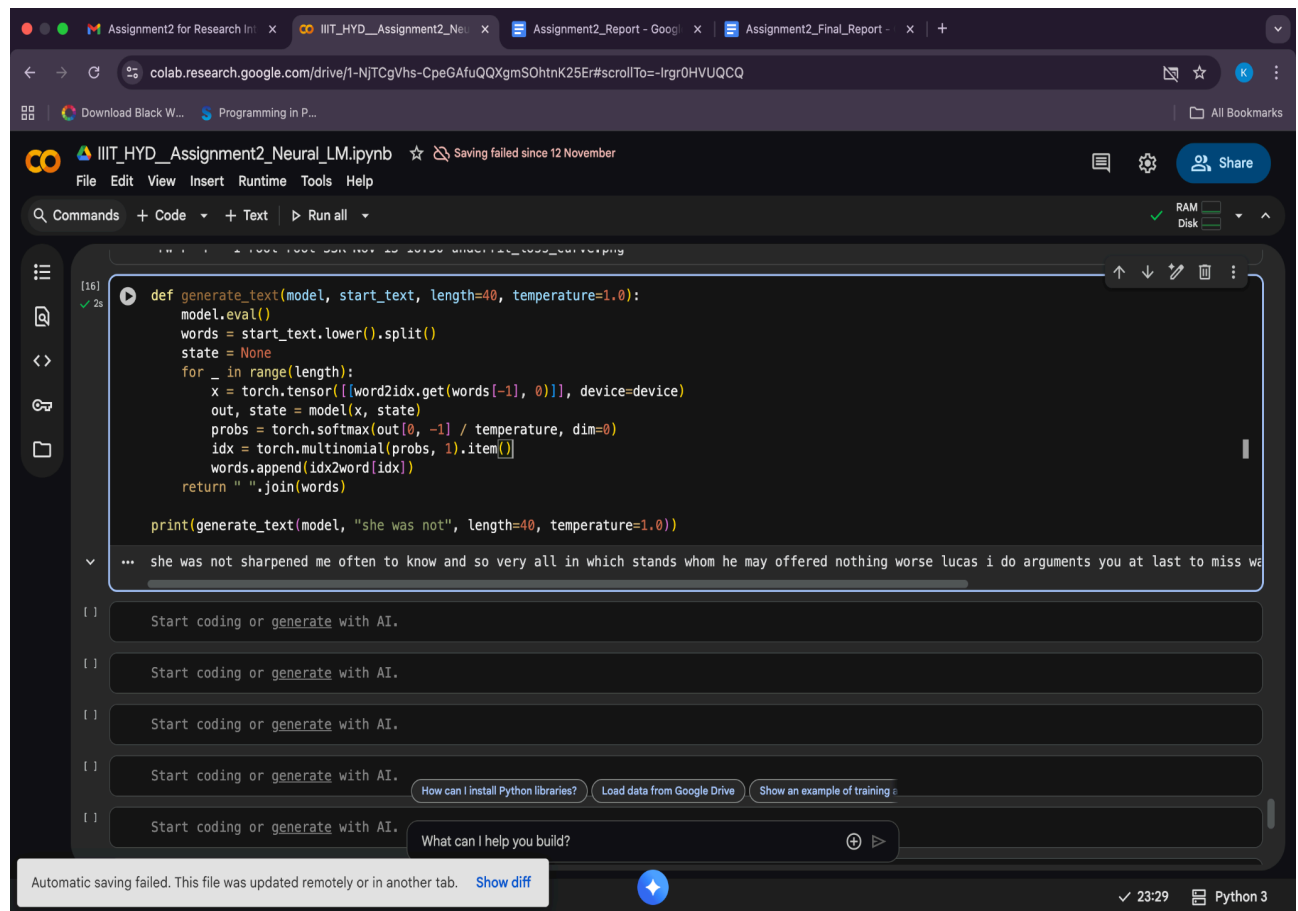




\



- Generated Text Output:



The screenshot shows a Google Colab notebook titled "IIIT_HYD_Assignment2_Neural_LM.ipynb". The notebook is open to a cell containing a Python function named `generate_text`. The function takes a model, a start text, a length, and a temperature as arguments. It generates text by repeatedly sampling from the model's output, starting from the provided start text. The output of the function is displayed below the code cell, showing a generated text snippet: "... she was not sharpened me often to know and so very all in which stands whom he may offered nothing worse lucas i do arguments you at last to miss wa". Below the output, there are several input fields for user interaction, each with the prompt "Start coding or generate with AI." and a "What can I help you build?" button. The bottom of the notebook shows a status bar with the text "Automatic saving failed. This file was updated remotely or in another tab." and a "Show diff" link. The bottom right corner displays the time "23:29" and the Python version "Python 3".

```
[16] def generate_text(model, start_text, length=40, temperature=1.0):  
    model.eval()  
    words = start_text.lower().split()  
    state = None  
    for _ in range(length):  
        x = torch.tensor([word2idx.get(words[-1], 0)], device=device)  
        out, state = model(x, state)  
        probs = torch.softmax(out[0, -1] / temperature, dim=0)  
        idx = torch.multinomial(probs, 1).item()  
        words.append(idx2word[idx])  
    return " ".join(words)  
  
print(generate_text(model, "she was not", length=40, temperature=1.0))
```

... she was not sharpened me often to know and so very all in which stands whom he may offered nothing worse lucas i do arguments you at last to miss wa

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

How can I install Python libraries? Load data from Google Drive Show an example of training

What can I help you build?

Automatic saving failed. This file was updated remotely or in another tab. [Show diff](#)

23:29 Python 3