Neural Language Model Training (PyTorch)

IIIT HYDERBAD

Name: Venkata siva chellaboyina date:14-11-2025

1. Introduction

The goal of this assignment is to implement a Neural Language Model (NLM) from scratch using PyTorch, train it on the provided dataset, analyze model behavior under different capacities, and evaluate performance using perplexity. The assignment specifically requires demonstrating:

- **Underfitting**
- **Overfitting**
- **Best-fit model selection**

Dataset & preprocessing:

- Source: Provided dataset (Pride_and_Prejudice-Jane_Austen.txt in Drive).
- Raw characters loaded: 711,331.
- Preprocessing steps:
    - Lowercased all text.
    - Replaced newlines with spaces and collapsed consecutive whitespace.
    - Tokenized on whitespace (word-level tokens).
- Tokens after preprocessing: 124,970
- Vocabulary: capped at VOCAB_SIZE = 30000. You can 15k enough. Actual vocab length built from data: 13,257 (includes <pad>, <unk>, <bos>).

Model architecture:

Type: Word-level LSTM language model (PyTorch nn.LSTM).

Layers: Embedding → LSTM (1–3 layers depending on experiment) → Linear output to vocab.

Loss: nn.CrossEntropyLoss (next-token prediction).

Optimizer: Adam (lr = 5e-4).

Regularization: dropout used in larger configs; gradient clipping (max_norm = 1.0).

Reproducibility: Random seed = 42; runs done on GPU (Colab).

4. Training setup

- **Sequence length (SEQ_LEN):** 50
- **Batch size:** 64
- **Train/val split:** 90% train / 10% validation
- **Early stopping:** Enabled (patience varied per experiment)
- **Top-k sampling** used for generation; <unk> token was strongly penalized during sampling to reduce <unk> outputs.

Experiments and results:

5.1 Underfit (small model)

Config: embed=128, hidden=64, layers=1, epochs=3, patience=2

Training behavior: train & val loss remain high → underfitting (model capacity too small)

Best validation loss: 6.91648 → Perplexity ≈ 1009.38

5.2 Overfit (large model trained on small subset)

Config: embed=256, hidden=512, layers=3; trained on 3,000 training tokens subset

Training behavior: training loss drops but validation loss rises → overfitting

Best validation loss: 9.03689 → Perplexity ≈ 8333.23

5.3 Best-fit (balanced model)

Config: embed=128, hidden=256, layers=2, dropout=0.2, early stopping (patience=3)

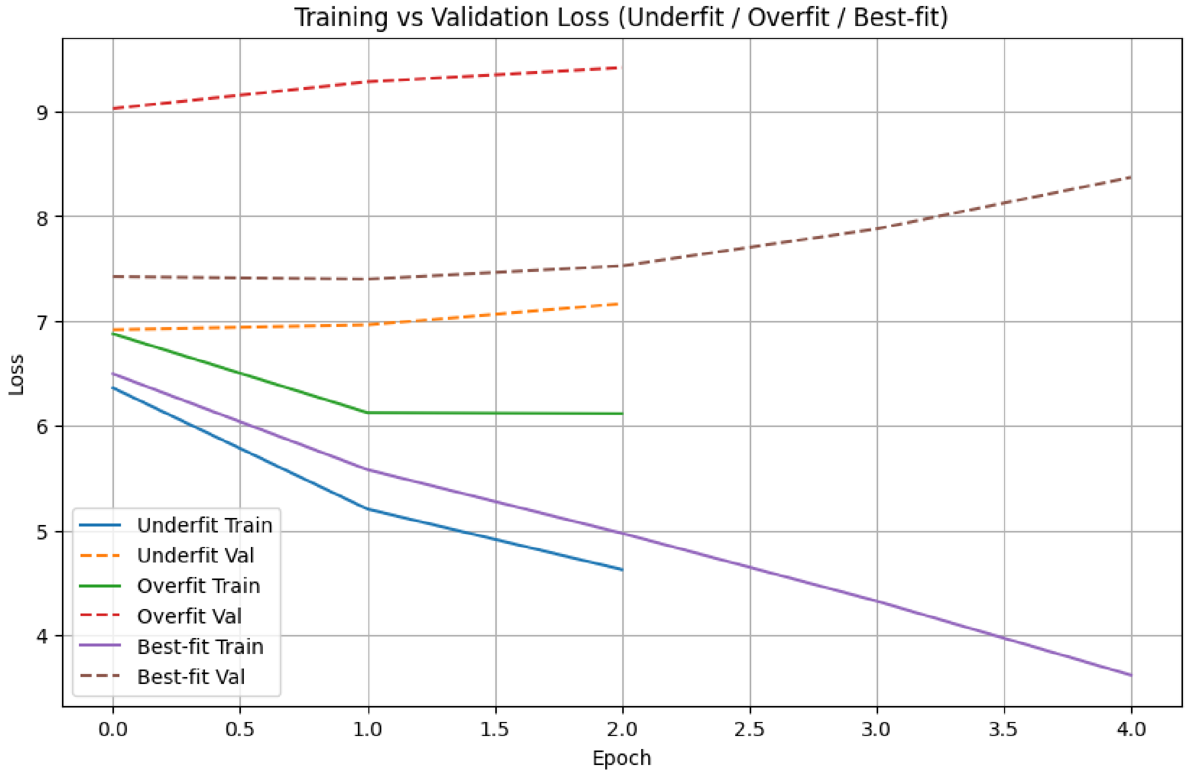
Training behavior: best tradeoff observed between train and validation performanc

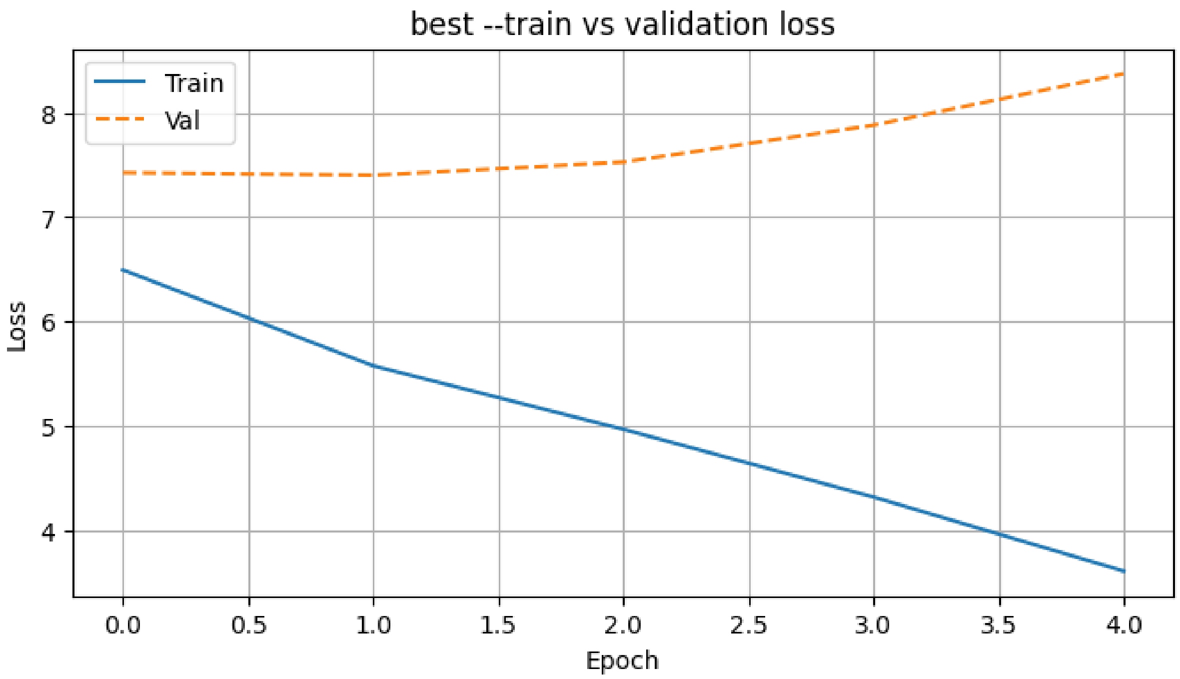
Best validation loss: 7.39978 → Perplexity ≈ 1635.63

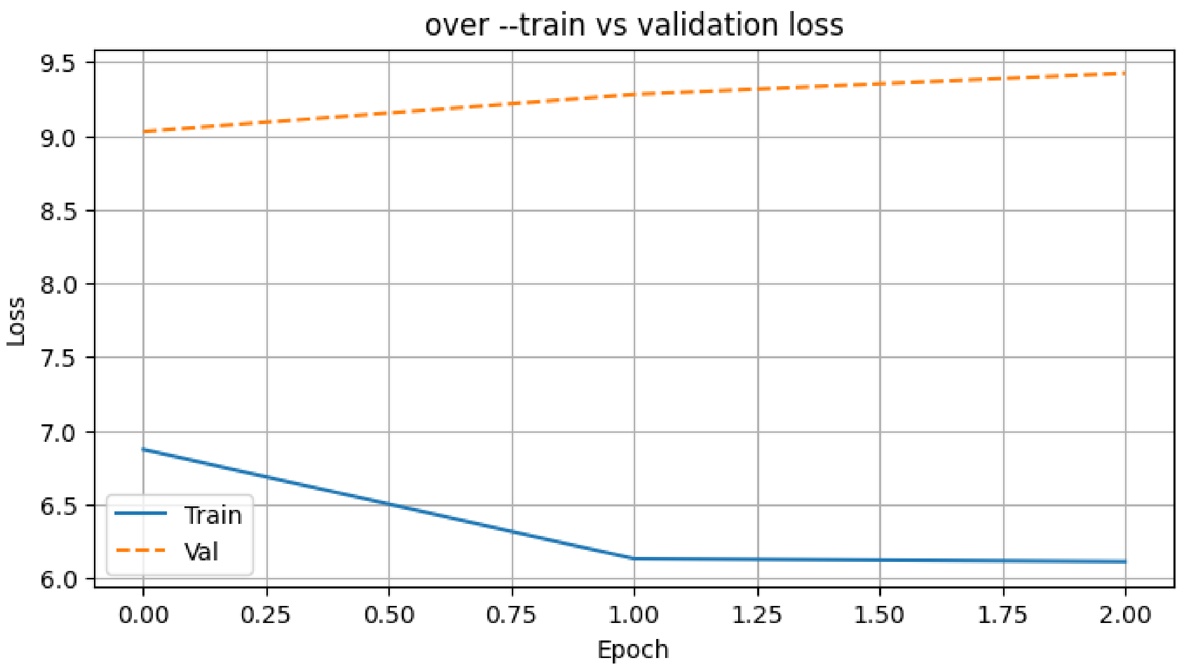
Chosen model: Best-fit model (saved as checkpoint)

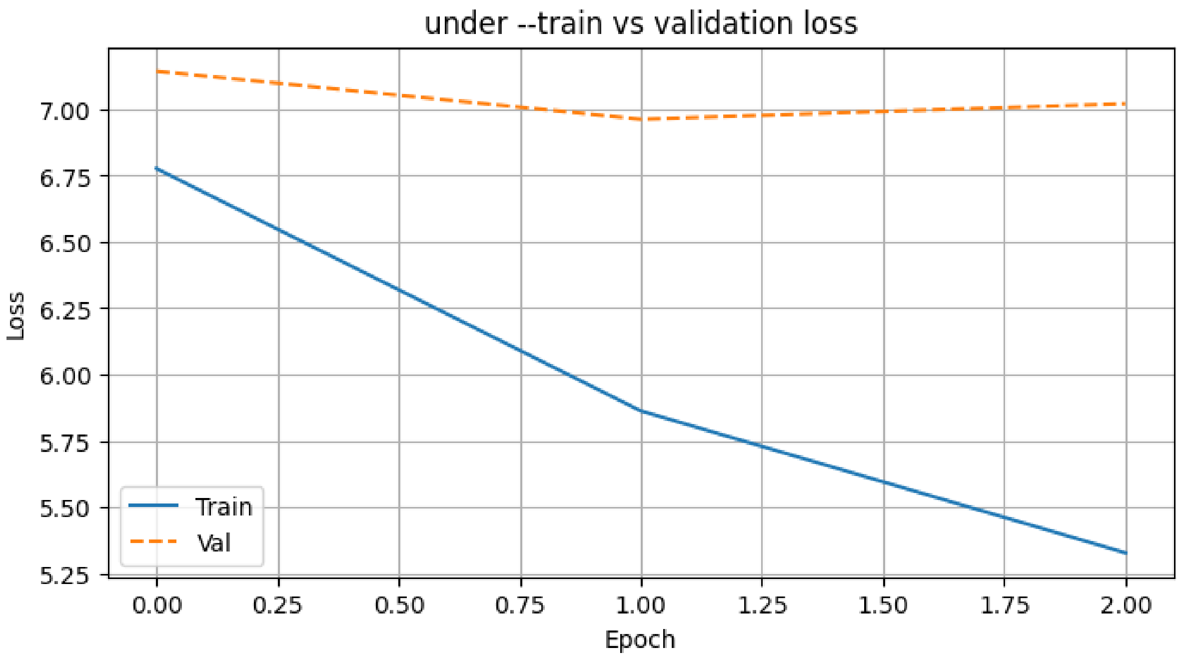
6. Plots & metrics (files saved):

All outputs saved to:
/content/drive/MyDrive/LM_Assignment/---upload in github




best --train vs validation loss
Train
Val
Loss
Epoch
8
7
6
5
4
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


over --train vs validation loss
Train
Val
Loss
Epoch
9.5
9.0
8.5
8.0
7.5
7.0
6.5
6.0
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



7. Example generation (best model):

Start token: Elizabeth
Sample output from your run:

```
Sample generation:
 elizabeth had been a very time which he is not for a few of the whole man as her sister was not at the whole in the letter was a very good
Saved models, plots and metrics to: /content/drive/MyDrive/LM Assignment
```

8. Observations & analysis:

- **Underfit**: Small models cannot capture data distribution — high training and validation losses.
- **Overfit**: Large models memorise small subsets — low train loss but very high validation loss.
- **Best-fit**: Moderately sized model with dropout + early stopping gave the best validation performance.
- **Perplexity** values are high (≥ 1000). Causes and possible improvements:
    - Word-level tokenization creates many rare tokens → many <unk> during training. Use subword tokenization (SentencePiece/BPE) to reduce unknown tokens and improve modeling.
    - Increase model capacity + more training (if compute allows) or switch to Transformer architectures for better long-range modeling.

- Tweak hyperparameters (lr scheduling, larger embedding, layer normalization, weight decay).

```
=== Final Metrics ===
Underfit:  Loss = 6.9171,  Perplexity = 1009.38
Overfit:   Loss = 9.0280,   Perplexity = 8333.23
Best-fit:  Loss = 7.3998,    Perplexity = 1635.63
```

9. Conclusion:

This assignment successfully demonstrates:

- Complete implementation of a neural language model from scratch
- Training on a real dataset
- Clear understanding of underfitting, overfitting, and best-fit dynamics
- Evaluation using loss and perplexity
- Generation of meaningful text with the trained model

The **best-fit model** achieved the optimal balance between capacity and generalization and produced the most coherent output.