ASSIGNMENT 2: Neural Language Model Training (PyTorch)

1. Objective

The objective of this assignment is to train a neural language model from scratch using PyTorch, evaluate it using perplexity, and demonstrate understanding of model capacity through underfitting, overfitting, and best-fit experiments.

The task also requires preparing a short report, submitting the code via GitHub, and providing loss plots and trained model files.

2. Dataset

The dataset used for training is the public-domain text of *Pride and Prejudice* by Jane Austen obtained from Project Gutenberg (EBook #42671).
All non-essential metadata sections (license, preparer notes) were removed, and the cleaned text contained approximately 120,000+ tokens.

Data preprocessing included:

- whitespace-based tokenization
- vocabulary creation (word-level)
- mapping tokens to integer IDs
- splitting into 80% train, 10% validation, 10% test

This dataset provides rich English prose suitable for training recurrent language models.

3. Model Architecture

A word-level LSTM Language Model was implemented from scratch using PyTorch.

Base model components

- Embedding Layer
- Multi-layer LSTM (num_layers = 1–3 depending on experiment)
- Fully-connected output layer projecting hidden states to vocabulary size
- Cross-entropy loss for next-token prediction
- Adam optimizer with gradient clipping

No pre-trained models or high-level language modeling libraries were used, as required.

4. Experiments and Results

Three experiments were conducted to illustrate different learning behaviors:

4.1 Underfit Model

Configuration:

- Embedding = 32
- Hidden Size = 32
- 1 LSTM Layer
- 5 epochs

Observation:
Both training and validation losses remained high and nearly identical.
The model lacked sufficient capacity to learn patterns in the dataset.

Validation Perplexity: ~900+

4.2 Overfit Model

Configuration:

- Training on only **2000 tokens**
- Large model: Embedding = 256, Hidden = 512, 3 Layers
- 15 epochs

Observation:
Training loss dropped drastically (toward 1.0), while validation loss increased steadily.
The model memorized the small subset → overfitting.

Validation Perplexity: ~1200+

4.3 Best-Fit Model

Configuration:

- Embedding = 128
- Hidden Size = 256
- 2 LSTM Layers
- Dropout = 0.3
- Trained on a reduced-but-large sample (~30k tokens)
- 7 epochs (fast version optimized for Colab)

Observation:
Training and validation losses decreased smoothly.

Validation perplexity was significantly lower than in underfit or overfit cases.
This model demonstrated the best generalization performance.

Validation Perplexity: ~110–150

5. Rationale for Selecting the Best Model

The **best-fit model** was selected because:

1. It achieved the lowest validation perplexity, the primary metric for evaluating language models.
2. It avoided both extremes:
    - Underfitting (too simple → high perplexity)
    - Overfitting (memorization → poor validation performance)
3. It exhibited balanced training and validation loss curves, indicating good generalization.
4. It used an appropriate model size and regularization (dropout), making it efficient and stable.

Thus, the best-fit model represents the optimal trade-off between capacity and generalization.

6. Submission Links

Colab file:

https://colab.research.google.com/drive/13FF8MytXJ0OQr3LwmO6s3_lTOucXkNR5?usp=sharing

Google Drive (Models + Plots):

https://drive.google.com/drive/folders/1ZcRCnwj22whWsQ8uvLTj_ap6hnliltb-?usp=sharing
(Include your folder link containing model_under.pth, model_over.pth, model_best.pth, and all .png plots)

7. Files Included

- model_under.pth – underfit model
- model_over.pth – overfit model
- model_best.pth – best model
- underfit_loss.png, overfit_loss.png, bestfit_loss.png
- vocab.json, results.json
- Full Google Colab Notebook
- Full Python scripts (