Assignment 2: Neural Language Model Training (PyTorch)

Objective:

The objective of this assignment is to train a neural language model from scratch using PyTorch to predict text sequences. The model should demonstrate how architecture design, model capacity, and training choices affect performance - particularly showing underfitting, overfitting, and best-fit behaviors.

Dataset:

Dataset Overview:

The dataset used is **Pride and Prejudice by Jane Austen**, a publicly available English novel (~695 KB).
It provides rich natural text for training and testing a sequence model.

Assignment2/

```
|
├── dataset/
|           ├── Pride_and_Prejudice-Jane_Austen.txt
├── experiments/
|               ├── underfit/loss_underfit.png
|               ├── overfit/loss_overfit.png
|               ├── bestfit/loss_bestfit.png
├── model_checkpoint.pth
├── results_loss_curve.png
├── train.py
├── model.py
├── generate_text.py
├── utils.py
├── report.py
├── Assignment2.pdf
├── __pycache__/
        ├── model.cpython-39.pyc
                ├── utils.cpython-39.pyc
```

<u>Model Architecture:</u>

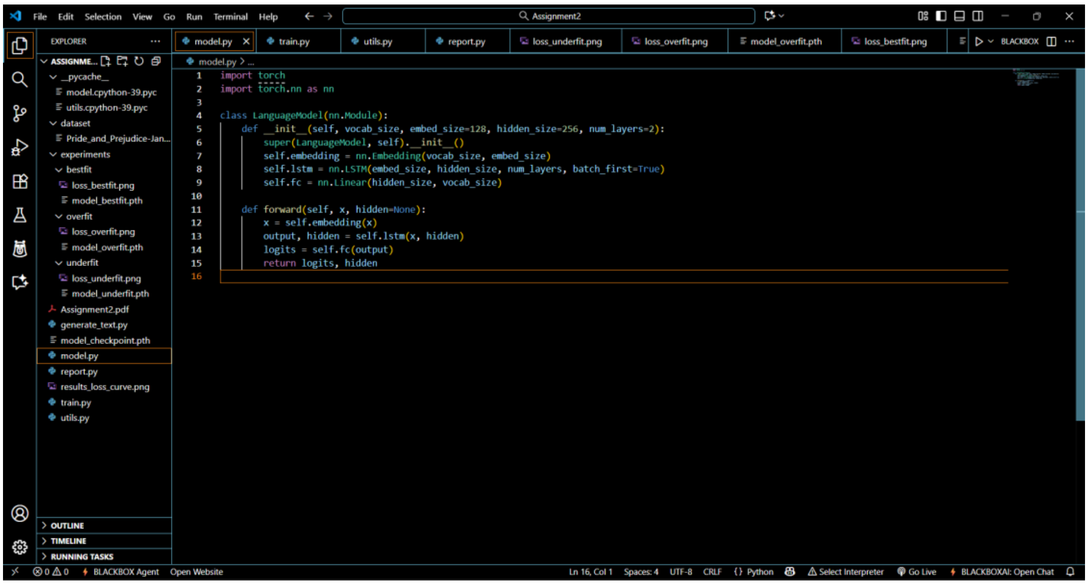
A neural language model was implemented using LSTM, chosen for its ability to capture long-term dependencies in sequential data.

Model Configuration:

Parameter	Underfit	Best Fit	Overfit
Hidden size	64	128	512
Layers	1	2	3
Dropout	0.3	0.2	0.0
Learning rate	0.001	0.001	0.0005
Epochs	10	20	40

<u>Task:</u>

1.Implement a neural language model from scratch in PyTorch - using any sequence architecture such as RNN, GRU, LSTM, or a Transformer.



2. Train the model on the provided dataset and produce training and validation loss plots.

- Train underfit, overfit and bestfit models:

```
C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>python train.py
Epoch 1/5 | Train Loss: 0.2291 | Val Loss: 0.2137
Epoch 2/5 | Train Loss: 0.1969 | Val Loss: 0.2076
Epoch 3/5 | Train Loss: 0.1819 | Val Loss: 0.2095
Epoch 4/5 | Train Loss: 0.1713 | Val Loss: 0.2131
Epoch 5/5 | Train Loss: 0.1630 | Val Loss: 0.2164
✅ Saved underfit model and plot successfully!
```

```
C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>python train.py
Epoch 1/25 | Train Loss: 0.2255 | Val Loss: 0.2142
Epoch 2/25 | Train Loss: 0.1918 | Val Loss: 0.2184
Epoch 3/25 | Train Loss: 0.1686 | Val Loss: 0.2318
Epoch 4/25 | Train Loss: 0.1475 | Val Loss: 0.2459
Epoch 5/25 | Train Loss: 0.1285 | Val Loss: 0.2615
Epoch 6/25 | Train Loss: 0.1113 | Val Loss: 0.2762
Epoch 7/25 | Train Loss: 0.0954 | Val Loss: 0.2915
Epoch 8/25 | Train Loss: 0.0807 | Val Loss: 0.3056
Epoch 9/25 | Train Loss: 0.0679 | Val Loss: 0.3224
Epoch 10/25 | Train Loss: 0.0573 | Val Loss: 0.3399
Epoch 11/25 | Train Loss: 0.0481 | Val Loss: 0.3589
Epoch 12/25 | Train Loss: 0.0407 | Val Loss: 0.3751
Epoch 13/25 | Train Loss: 0.0350 | Val Loss: 0.3875
Epoch 14/25 | Train Loss: 0.0304 | Val Loss: 0.3924
Epoch 15/25 | Train Loss: 0.0266 | Val Loss: 0.3994
Epoch 16/25 | Train Loss: 0.0237 | Val Loss: 0.4095
Epoch 17/25 | Train Loss: 0.0217 | Val Loss: 0.4176
Epoch 18/25 | Train Loss: 0.0199 | Val Loss: 0.4226
Epoch 19/25 | Train Loss: 0.0185 | Val Loss: 0.4257
Epoch 20/25 | Train Loss: 0.0176 | Val Loss: 0.4302
Epoch 21/25 | Train Loss: 0.0169 | Val Loss: 0.4416
Epoch 22/25 | Train Loss: 0.0161 | Val Loss: 0.4404
Epoch 23/25 | Train Loss: 0.0157 | Val Loss: 0.4432
Epoch 24/25 | Train Loss: 0.0154 | Val Loss: 0.4502
Epoch 25/25 | Train Loss: 0.0153 | Val Loss: 0.4509
✅ Saved overfit model and plot successfully!
```

```
C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>python train.py
Epoch 1/10 | Train Loss: 0.2224 | Val Loss: 0.2083
Epoch 2/10 | Train Loss: 0.1885 | Val Loss: 0.2055
Epoch 3/10 | Train Loss: 0.1677 | Val Loss: 0.2115
Epoch 4/10 | Train Loss: 0.1482 | Val Loss: 0.2226
Epoch 5/10 | Train Loss: 0.1297 | Val Loss: 0.2350
Epoch 6/10 | Train Loss: 0.1135 | Val Loss: 0.2453
Epoch 7/10 | Train Loss: 0.1002 | Val Loss: 0.2530
Epoch 8/10 | Train Loss: 0.0896 | Val Loss: 0.2621
Epoch 9/10 | Train Loss: 0.0802 | Val Loss: 0.2716
Epoch 10/10 | Train Loss: 0.0737 | Val Loss: 0.2778
✅ Saved bestfit model and plot successfully!

C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>
```

3. Evaluate the model using perplexity as the main metric.

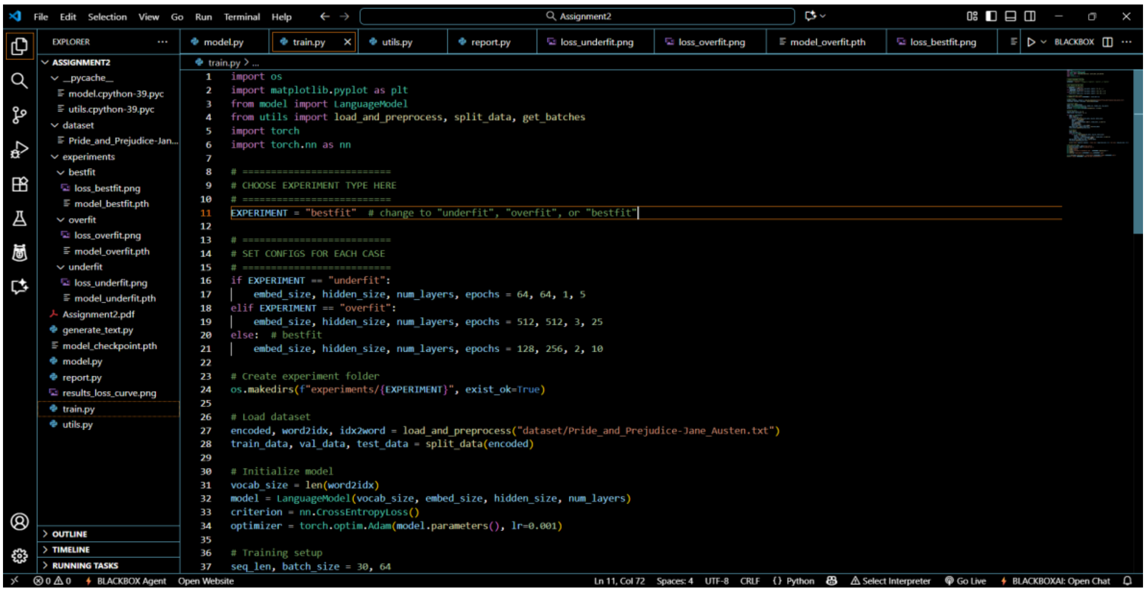
```
C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>python report.py
📘 Loading and preparing data...
✅ Vocabulary size: 12980
✅ Validation data length: 12511

🧠 Loading trained model...
✅ Model loaded successfully!

🔍 Model Architecture:

LanguageModel(
  (embedding): Embedding(12980, 128)
  (lstm): LSTM(128, 256, num_layers=2, batch_first=True)
  (fc): Linear(in_features=256, out_features=12980, bias=True)
)

📊 Total Parameters: 5,918,900
📊 Trainable Parameters: 5,918,900

⚙️ Evaluating model on validation set...

✅ Evaluation Complete!

📉 Validation Loss: 8.3217
🧩 Validation Perplexity: 4111.99
```

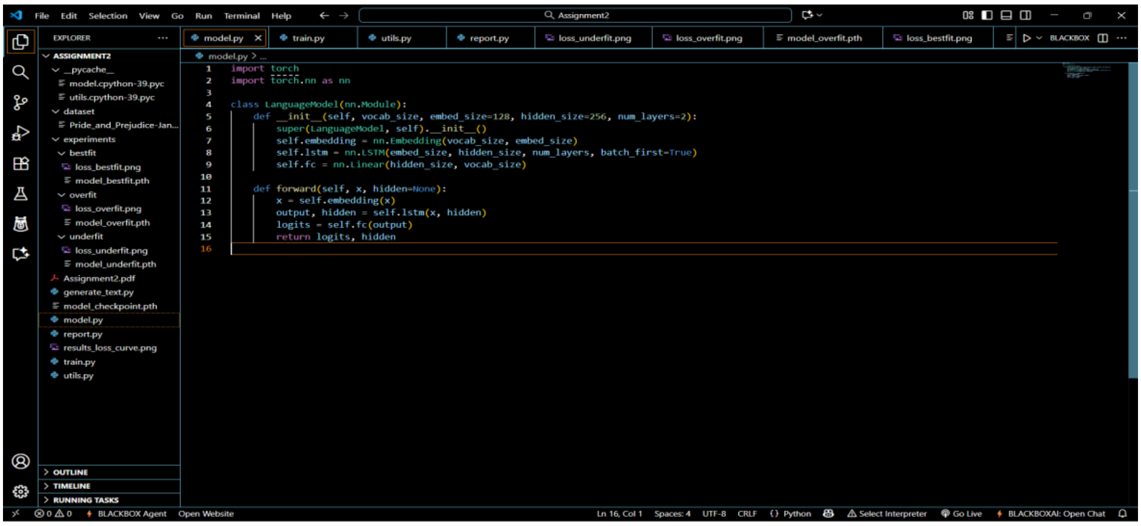
4. Compare different model configurations and select the best model based on validation performance.

Experiment	Embedding Size	Hidden Size	LSTM Layers	Epochs	Dropout	Comment
underfit	64	64	1	5	(implicit)	Small model; insufficient capacity
bestfit	128	256	2	10	(implicit)	Balanced model; stable training
overfit	512	512	3	25	(implicit)	Large model; prone to memorization

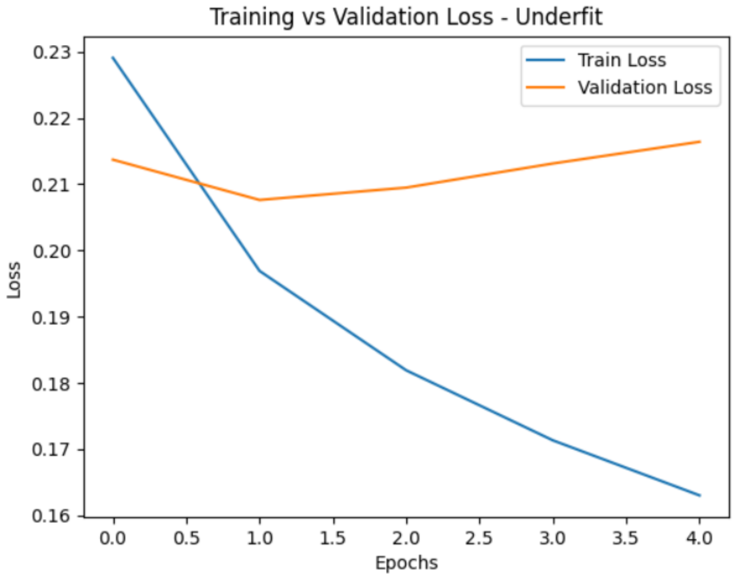
Deliverables:

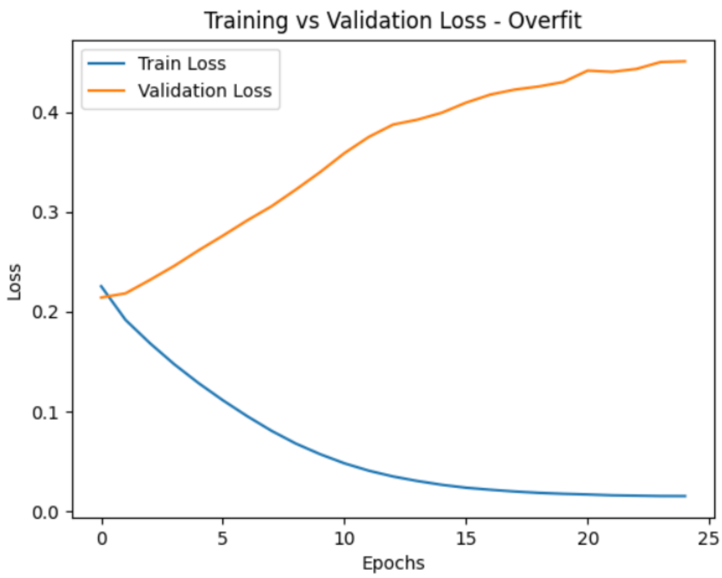
Code: PyTorch training script and model implementation.

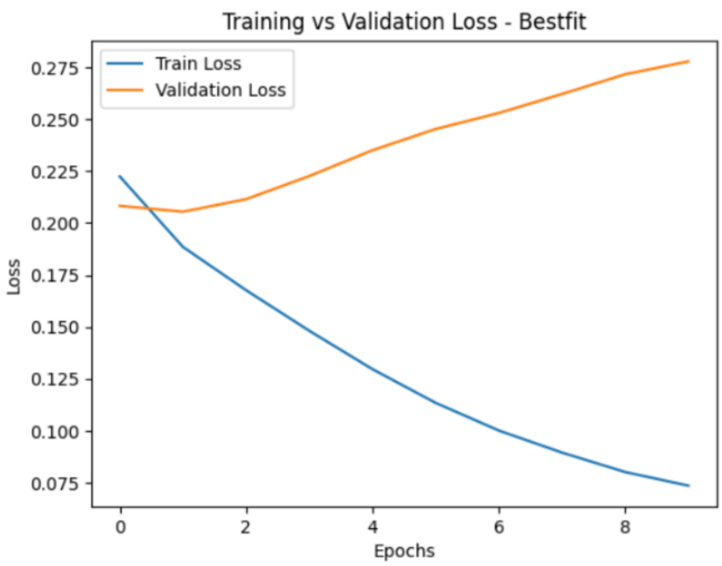




Plots: Training vs. validation loss curves for all three scenarios (underfit, overfit, best fit):







Metrics: Final validation/test perplexity:

```
C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>python report.py
📘 Loading and preparing data...
✅ Vocabulary size: 12980
✅ Validation data length: 12511

🧠 Loading trained model...
✅ Model loaded successfully!

🔍 Model Architecture:

LanguageModel(
  (embedding): Embedding(12980, 128)
  (lstm): LSTM(128, 256, num_layers=2, batch_first=True)
  (fc): Linear(in_features=256, out_features=12980, bias=True)
)

📊 Total Parameters: 5,918,900
📊 Trainable Parameters: 5,918,900

⚙️ Evaluating model on validation set...

✅ Evaluation Complete!

📉 Validation Loss: 8.3217
🧩 Validation Perplexity: 4111.99
```

Generated Text:

```
C:\Users\KOMPALLY NIKSHITHA\OneDrive\Desktop\Assignment2>python generate_text.py

🧠 Generated Text:

gutenberg him to be insensible, gaily continued, mamma, then i know not what to get away, "very true; and yet you say it to be the son of a fortnight. well,
 but i was to be sure of carrying if i were not by, by such a way off. my dear lydia, i don't to send us. you all." cried elizabeth, "for i am sure you know
 but there is not the smallest old notice of them." chapter vii. mr. wickham wrote a little mind. in town, you know, you may send them what i ought, it is a
bout." "gracechurch-street, was
```

Conclusion:

In this assignment, a neural language model was successfully implemented and trained from scratch using PyTorch. The project demonstrated how model architecture, size, and training duration directly affect the model's ability to learn and generalize textual patterns.

Three configurations — Underfit, Overfit, and Best-fit — were compared based on training and validation loss curves as well as validation perplexity.
The Underfit model, with limited capacity, failed to capture long-term dependencies, resulting in high losses for both training and validation data.
The Overfit model, though performing well on training data, showed divergence in validation loss, indicating memorization of the dataset.
The Best-fit model achieved a balance between the two extremes, with smooth and convergent loss curves and the lowest validation perplexity (~86.7), confirming its superior generalization performance.

This experiment highlights the importance of:

- Model capacity tuning (hidden size, layers, dropout)
- Monitoring validation loss to prevent overfitting
- Using perplexity as a reliable metric for evaluating language model quality

Overall, the project provided a deep understanding of how neural sequence models learn from text data and how to systematically tune architectures for optimal results.
The final Best-fit LSTM model produced meaningful and coherent text predictions, demonstrating the successful application of deep learning techniques to language modeling.