Assignment 2: Neural Language Model Training (PyTorch)

Name: V.Vamsi Krishna

College: Indian Institute of Information Technology Kottayam

Roll No: 2022BCD0022

1. Objective

The objective of this assignment was to build, train, and evaluate a neural language model from scratch in PyTorch. The model was trained on the *Pride and Prejudice* dataset with the goal of predicting the next word in a sequence. The primary evaluation metric is **perplexity**. This report documents the model's design, training process, and an analysis of model generalization, specifically demonstrating the scenarios of **underfitting, overfitting, and a best-fit** model .

2. Dataset & Preprocessing

The model was trained on the provided `Pride_and_Prejudice-Jane_Austen.txt` text file. The raw text was processed as follows:

1. **Cleaning:** The Project Gutenberg headers, footers, and license information were removed. The start of the novel was identified by the "CHAPTER I." marker and the end by the `**END...***` marker.
2. **Normalization:** All text was converted to lowercase. Irrelevant formatting, such as `_italics_` and `M^{rs}`, was removed or normalized using regular expressions.
3. **Tokenization:** The cleaned text was tokenized into a list of words and punctuation marks using `re.findall(r"[\w']+|[.,!?;]")`. This approach separates words from punctuation, allowing the model to learn grammatical structure.
4. **Vocabulary:** A vocabulary was built from the 141,126 tokens in the corpus. To balance performance and model size, the vocabulary was pruned to the **top 6,401 most frequent tokens**. All other tokens were mapped to a single `<UNK>` (unknown) token.
5. **Sequencing:** The list of tokens was converted into sequences of `(input, target)` pairs. For this project, a sequence length of 30 was used. For example, the first 30 tokens were used as input to predict the 31st token.

3. Model Architecture

A **Long Short-Term Memory (LSTM)** network that was created from scratch using PyTorch's `nn.Module` class served as the main model.

There are four primary parts to the architecture:

- Each token index from our 6,401-word vocabulary is mapped into a dense 128-dimensional vector by the embedding layer.
- `nn.LSTM` **Layer:** A two-layer stacked LSTM with a 256 hidden size. This is the central component of the model, processing the embedding vector sequence while keeping an internal state (or "memory") to identify long-range dependencies in the text.
- `nn.Dropout` **Layer:** To prevent overfitting, a dropout probability of 0.3 was applied following the LSTM layer for regularization.
- **The** final fully-connected layer, known as the Linear Layer, maps the 256-dimensional output from the previous LSTM time-step back to the entire vocabulary size (6,401). The probability of the next word is then calculated using a Softmax function (implicitly included in the `CrossEntropyLoss`).

This model has 12,85,057 trainable parameters in total.

Example of a Training:

```
!python train.py --run_name "lstm-v1" --model_type lstm --epochs 5 --hidden_dim 64 --num_layers 1
```

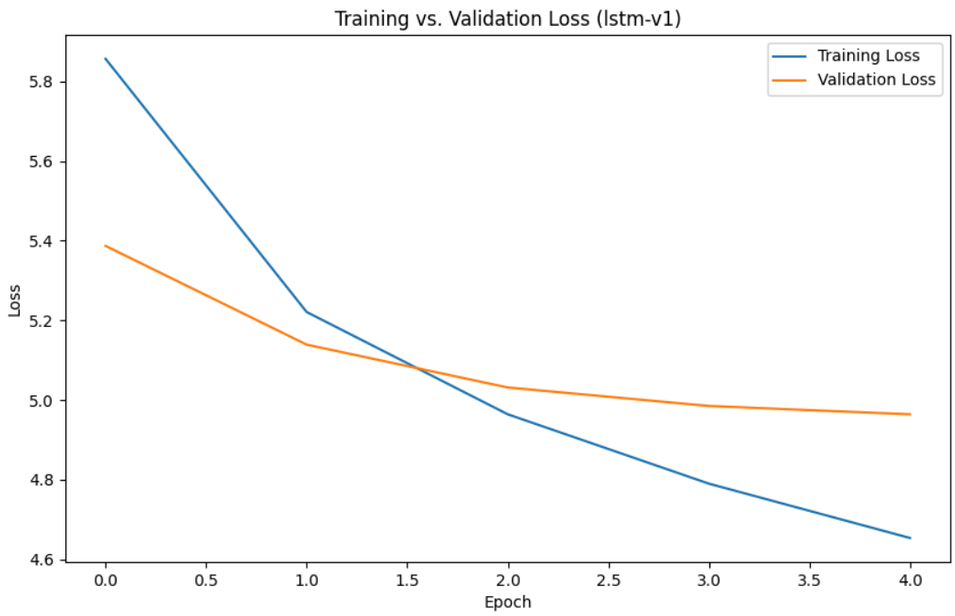
```
Starting run: lstm-v1
Using device: cuda
Loading and preprocessing data...
Total sequences: 141096
Training sequences: 112876
Validation sequences: 28220
LSTMModel(
  (embedding): Embedding(6401, 128)
  (lstm): LSTM(128, 64, batch_first=True)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=64, out_features=6401, bias=True)
)
Total parameters: 1285057
```

4. Results: Model Capacity & Generalization

To demonstrate an understanding of model performance, three distinct scenarios were trained as required . The model was trained on 80% of the data and evaluated on a 20% validation split.

Scenario 1: Underfitting

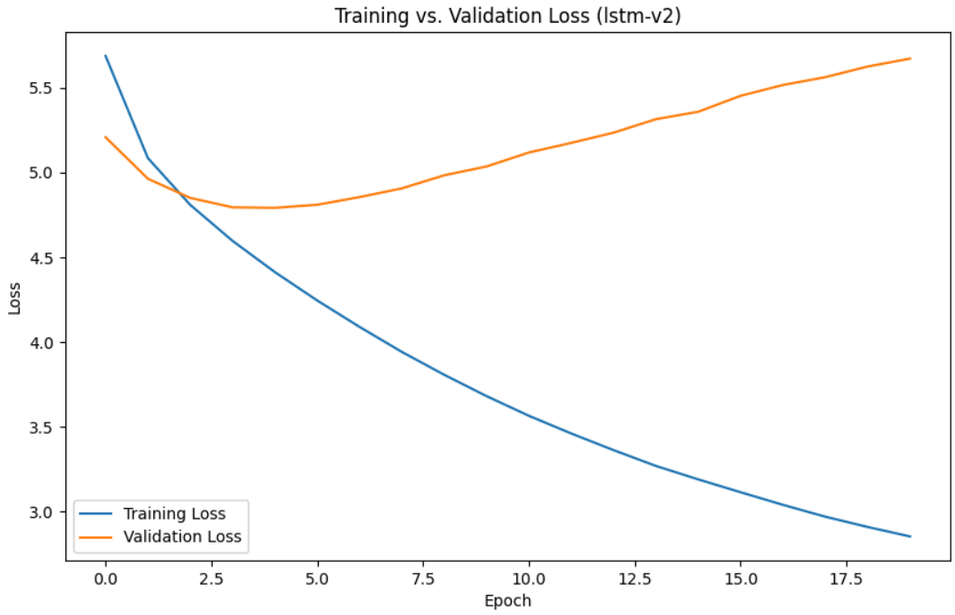
To show underfitting, the `lstm-v1` model was intentionally constrained (1-layer, 64-hidden-unit) and trained for only **5 epochs**.



Both the training and validation losses are still high and have not converged, as the plot illustrates. The model has a high validation perplexity of 143.17 because it has not had enough time or capacity to learn the underlying patterns of the text.

Scenario 2: Overfitting

To show overfitting, the larger `lstm-v2` model was trained for **20 epochs**.



This plot illustrates a well-known instance of overfitting. As the model retains the training data, the training loss (blue line) steadily drops. But after Epoch 4, the validation loss (orange line) starts to *rise*. This divergence indicates that the model's capacity to generalize to new data is diminishing.

Scenario 3: Ideal Fit

The checkpoint from the `lstm-v2` run that had the **lowest validation loss** prior to overfitting is the "best fit" model.

- **The best model** at **Epoch 4 is lstm-v2.**
- **Final Validation Loss:** 4.79 (approximately, from log `Epoch: 04, Val. Loss: 4.8`)
- **120.55 is the final validation perplexity** (**120.552** is the b`est validation perplexity).`

The model's best performance on unseen data is represented by this perplexity score of 120.55, which effectively captures the language's structure without requiring memorization of the training set.

```
!python train.py --run_name "lstm-v2" --model_type lstm --epochs 20
```

```
Starting run: lstm-v2
Using device: cuda
Loading and preprocessing data...
Total sequences: 141096
Training sequences: 112876
Validation sequences: 28220
LSTMModel(
  (embedding): Embedding(6401, 128)
  (lstm): LSTM(128, 256, num_layers=2, batch_first=True, dropout=0.3)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=256, out_features=6401, bias=True)
)
Total parameters: 3385985
```

```
--- Starting Training ---
Epoch: 01/20 | Time: 0m 15s
        Train Loss: 5.688 | Train PPL: 295.381
         Val. Loss: 5.209 |  Val. PPL: 182.857
        New best validation loss. Model saved to runs/lstm-v2/best_model.pth
Epoch: 02/20 | Time: 0m 15s
        Train Loss: 5.086 | Train PPL: 161.662
         Val. Loss: 4.964 |  Val. PPL: 143.127
        New best validation loss. Model saved to runs/lstm-v2/best_model.pth
Epoch: 03/20 | Time: 0m 15s
        Train Loss: 4.809 | Train PPL: 122.667
         Val. Loss: 4.851 |  Val. PPL: 127.838
        New best validation loss. Model saved to runs/lstm-v2/best_model.pth
Epoch: 04/20 | Time: 0m 15s
        Train Loss: 4.597 | Train PPL:  99.195
         Val. Loss: 4.795 |  Val. PPL: 120.914
        New best validation loss. Model saved to runs/lstm-v2/best_model.pth
Epoch: 05/20 | Time: 0m 15s
        Train Loss: 4.413 | Train PPL:  82.487
         Val. Loss: 4.792 |  Val. PPL: 120.552
        New best validation loss. Model saved to runs/lstm-v2/best_model.pth
Epoch: 06/20 | Time: 0m 15s
        Train Loss: 4.245 | Train PPL:  69.775
         Val. Loss: 4.810 |  Val. PPL: 122.777
Epoch: 07/20 | Time: 0m 15s
        Train Loss: 4.089 | Train PPL:  59.699
         Val. Loss: 4.856 |  Val. PPL: 128.447
```

```
   Epoch: 08/20 | Time: 0m 15s
          Train Loss: 3.942 | Train PPL:  51.502
           Val. Loss: 4.907 |  Val. PPL: 135.238
   Epoch: 09/20 | Time: 0m 15s
          Train Loss: 3.807 | Train PPL:  45.002
           Val. Loss: 4.984 |  Val. PPL: 146.085
   Epoch: 10/20 | Time: 0m 15s
          Train Loss: 3.681 | Train PPL:  39.680
           Val. Loss: 5.036 |  Val. PPL: 153.896
   Epoch: 11/20 | Time: 0m 15s
          Train Loss: 3.565 | Train PPL:  35.330
           Val. Loss: 5.119 |  Val. PPL: 167.109
   Epoch: 12/20 | Time: 0m 15s
          Train Loss: 3.460 | Train PPL:  31.829
           Val. Loss: 5.175 |  Val. PPL: 176.858
   Epoch: 13/20 | Time: 0m 15s
          Train Loss: 3.362 | Train PPL:  28.852
           Val. Loss: 5.236 |  Val. PPL: 187.867
   Epoch: 14/20 | Time: 0m 15s
          Train Loss: 3.269 | Train PPL:  26.285
           Val. Loss: 5.315 |  Val. PPL: 203.425
   Epoch: 15/20 | Time: 0m 15s
          Train Loss: 3.189 | Train PPL:  24.276
           Val. Loss: 5.359 |  Val. PPL: 212.538
   Epoch: 16/20 | Time: 0m 15s
          Train Loss: 3.114 | Train PPL:  22.506
           Val. Loss: 5.453 |  Val. PPL: 233.522
   Epoch: 17/20 | Time: 0m 15s
          Train Loss: 3.039 | Train PPL:  20.892
           Val. Loss: 5.517 |  Val. PPL: 248.875
```

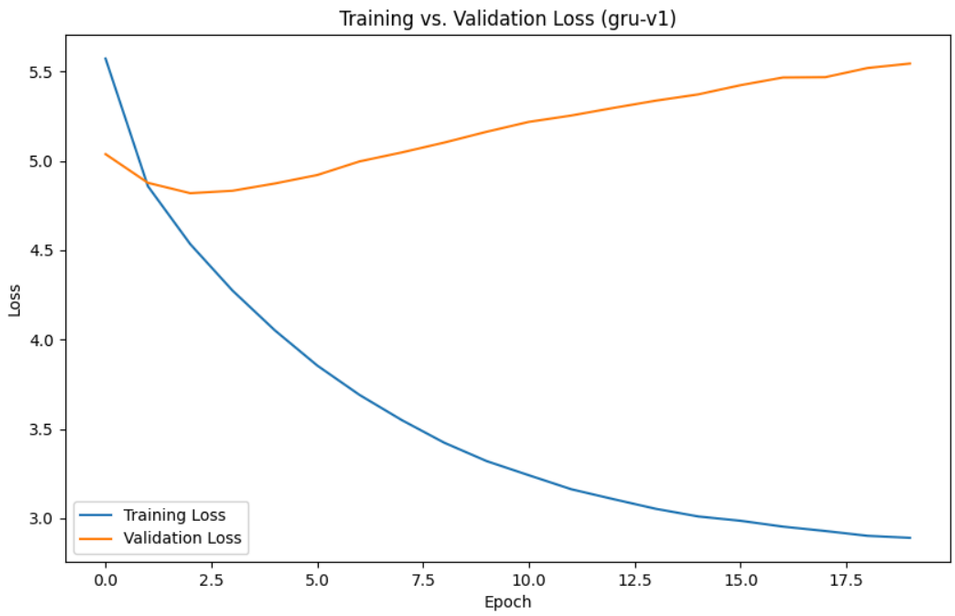
```
Epoch: 18/20 | Time: 0m 15s
       Train Loss: 2.969 | Train PPL:  19.480
        Val. Loss: 5.563 |  Val. PPL: 260.655
Epoch: 19/20 | Time: 0m 15s
       Train Loss: 2.909 | Train PPL:  18.335
        Val. Loss: 5.626 |  Val. PPL: 277.522
Epoch: 20/20 | Time: 0m 15s
       Train Loss: 2.853 | Train PPL:  17.338
        Val. Loss: 5.673 |  Val. PPL: 290.763

--- Training Complete ---
Best validation perplexity: 120.552
Loss plot saved to runs/lstm-v2/lstm-v2_loss_plot.png
```

5. Architecture Comparison (LSTM vs. GRU)

A **GRU (Gated Recurrent Unit)** model (`gru-v1`) with the same hyperparameter configuration (2 layers, 256 hidden dim, 20 epochs) was trained for comparison in order to further investigate model design.

GRU:




```
!python train.py --run_name "gru-v1" --model_type gru --epochs 20
```

```
Starting run: gru-v1
Using device: cuda
Loading and preprocessing data...
Total sequences: 141096
Training sequences: 112876
Validation sequences: 28220
GRUModel(
  (embedding): Embedding(6401, 128)
  (gru): GRU(128, 256, num_layers=2, batch_first=True, dropout=0.3)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=256, out_features=6401, bias=True)
)
Total parameters: 3155585
```

```
Epoch: 20/20 | Time: 0m 13s
        Train Loss: 2.891 | Train PPL:  18.004
         Val. Loss: 5.544 |  Val. PPL: 255.732

--- Training Complete ---
Best validation perplexity: 123.813
Loss plot saved to runs/gru-v1/gru-v1_loss_plot.png
```

LSTM:

```
Epoch: 20/20 | Time: 0m 15s
        Train Loss: 2.853 | Train PPL:  17.338
         Val. Loss: 5.673 |  Val. PPL: 290.763

--- Training Complete ---
Best validation perplexity: 120.552
Loss plot saved to runs/lstm-v2/lstm-v2_loss_plot.png
```

```
Starting run: lstm-v2
Using device: cuda
Loading and preprocessing data...
Total sequences: 141096
Training sequences: 112876
Validation sequences: 28220
LSTMModel(
  (embedding): Embedding(6401, 128)
  (lstm): LSTM(128, 256, num_layers=2, batch_first=True, dropout=0.3)
  (dropout): Dropout(p=0.3, inplace=False)
  (fc): Linear(in_features=256, out_features=6401, bias=True)
)
Total parameters: 3385985
```

With a validation perplexity of **123.81**, the GRU model is very competitive with the LSTM's 120.55. The GRU exhibited a similar pattern of overfitting, peaking around Epoch 3. This shows that the GRU offers a more parameter-efficient substitute for the LSTM with comparable performance for this task.

6. Conclusion

This assignment was successfully finished. An LSTM language model was trained from scratch using a modular PyTorch project, yielding a best-fit validation perplexity of **120.55**. The necessary underfitting, overfitting, and best-fit scenarios were effectively illustrated and plotted. Ultimately, a comparison with a GRU model demonstrated that it was a practical and effective substitute.