Assignment 2: Neural Language Model Training (PyTorch)

Objective:

Train a neural language model from scratch using PyTorch. The goal is to demonstrate understanding of how sequence models learn to predict text and how model design and training affect performance.

Dataset:

A pre-selected dataset will be provided for this assignment. Candidates should use only the given dataset for all experiments. Data preprocessing, tokenization, and batching should still be implemented by the candidate using PyTorch and standard Python tools.

Task Overview:

You are required to:

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.

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

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

5. Prepare a short report summarizing your setup, results, and key observations.

Computational Resources:

Candidates may utilize platforms such as Google Colab or Kaggle to access GPU resources for model training and experimentation.

Understanding Checkpoints:

To demonstrate understanding of model capacity and generalization, include experiments that show:

1. Underfitting

2. Overfitting

3. Best Fit

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.

· Report: Concise explanation of dataset, model, results, and rationale for selecting the best model.

Submission Instructions:

· Push the complete code to a public GitHub repository.

· Include a README in the repo with clear instructions on how to run training and inference, along with links to any trained models (e.g., Google Drive links).

· Submit the report via email, including the link to the GitHub repository. The repository and Google Drive links must be accessible publicly for evaluation.

· Extra credit: Points may be awarded for additional effort, such as improved model performance, advanced data preprocessing, or any innovative approach that enhances the final model.

Rules:

· Use only the provided dataset.

· Implement everything from scratch using PyTorch (no pre-trained models or high-level LM libraries).

· Ensure your code is reproducible with fixed random seeds and instructions for execution.