Assignment 2 : Neural Language Model Training from scratch

Github repository : https://github.com/Puneeth-Abhishek-6622/Research-Internship-Assignment-2

1. Objective

To build, train, and evaluate neural language models (LSTM, GRU, Transformer) from scratch in PyTorch to analyze how model architecture and design choices affect performance, as measured by Perplexity (PPL).

2. Dataset and pre-processing

➢ Dataset source

The raw text for this analysis was sourced from **Project Gutenberg's** public domain library, using the full text of Jane Austen's **"Pride and Prejudice"**.

➢ Preprocessing steps

Before tokenization, the raw text underwent the following sequence of preprocessing steps to ensure clean, standardized data:

- **Content Extraction:** The text was strictly isolated to the main narrative content. It was programmatically sliced from the starting marker "CHAPTER I" up to the defined end marker ("End of the Project Gutenberg"), removing all headers and footers.
- **Lowercasing:** All characters in the extracted text were converted to lowercase to ensure case-insensitivity (e.g., "The" and "the" are treated as the same word).
- **Cleaning:** Punctuation, numbers, and extra spaces (including tabs and multiple consecutive spaces) were removed using regular expressions (re.sub(r'[^a-z\s]', ' ', text)). This simplifies the vocabulary, forcing the model to learn relationships purely from the words themselves.

➢ Tokenization and Sequence Generation

This project implemented and compared two tokenization strategies: **word-level** and **subword-level (BPE).** The final models reported below were trained using word-level tokenization for a direct comparison of architectures.

- **Method:** Simple space-separated tokenization. A minimum frequency cutoff of 3 (min_freq = 3) was applied to remove extremely rare words, reducing noise and vocabulary size.
- **Vocabulary:** A vocabulary was built from the set of all unique words appearing 3 or more times. Special tokens "<pad>" (for padding) and "<unk>" (for unknown words) were added. This resulted in a final vocabulary size of 3,740 tokens.
- **Sequencing (RNN):** For the RNN models (LSTM, GRU), a sliding window with a sequence length of 20 tokens was used to create input-target pairs (seq_len = 20). This resulted in approximately 127,000 sequences.
- **Sequencing (Transformer):** For the Transformer model, a larger sequence length of 64 tokens was used.
- **Data Split (RNN):** The RNN models were trained on an 80% training / 20% validation split.
- **Data Split (Transformer):** The Transformer model was trained on a 90% training / 10% validation split.

3. Evaluation metrics : Perplexity (PPL)

Perplexity (PPL) was the primary metric. It measures how well a model predicts a test set, with a lower score being better. It can be interpreted as the model's "average branching factor," or its uncertainty in predicting the next token.

4. Experimental Setup & Project flow

The RNN and Transformer models were trained with different hyperparameters, as each architecture has different requirements for optimal performance.

- **Model Architectures** Nine distinct model architectures were implemented from scratch in PyTorch to compare their effectiveness on this task:

1. **LSTM**: A standard 2-layer LSTM model.
2. **BiLSTM**: A 2-layer bidirectional LSTM.
3. **BiLSTM + Attention**: A 2-layer BiLSTM with a simple attention mechanism.
4. **GRU**: A standard 2-layer GRU model.
5. **BiGRU**: A 2-layer bidirectional GRU.

6. **BiGRU + Attention**: A 2-layer BiGRU with a simple attention mechanism.
7. **Transformer (Base)**: A 4-layer Transformer Encoder model.
8. **Transformer + Attention**: A Transformer with an additional attention layer on the output.
9. **Transformer + Head**: A Transformer with a modified output linear head.

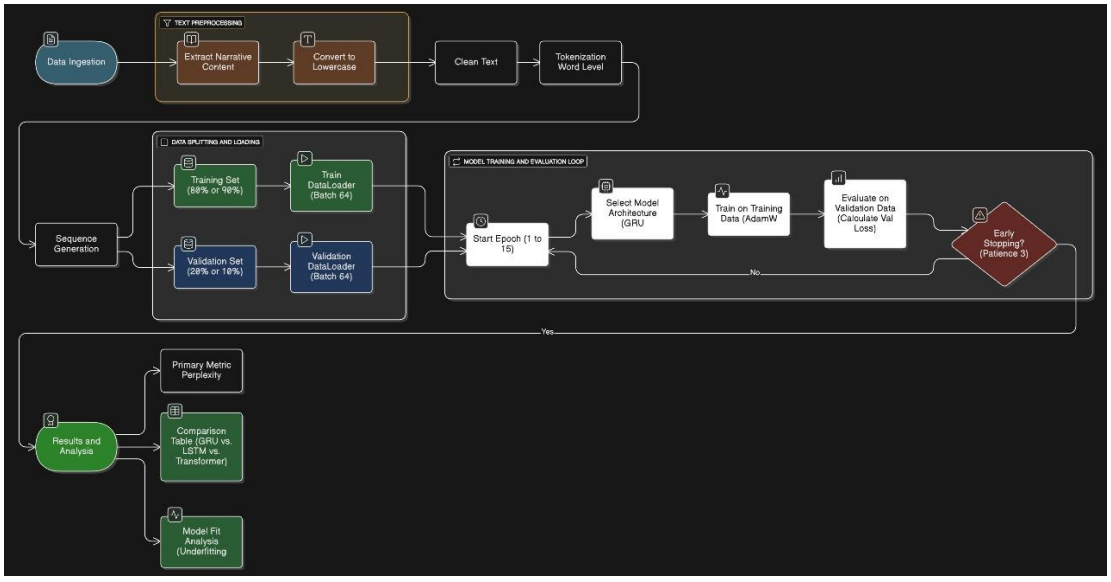
- **RNN (LSTM/GRU) training parameters** All RNN models were trained with the following shared hyperparameters:

Hyperparameter	Value
Optimizer	`torch.optim.AdamW`
Learning Rate	`3e-4`
Loss Function	`nn.CrossEntropyLoss`
Max Epochs	15
Early Stopping	Patience of 3 epochs
Batch Size	64
Sequence Length	20
Data Split	80% Train / 20% Val
Embedding Dimension	128
Hidden Dimension	256
RNN Layers	2

- **Transformer training parameters**

Hyperparameter	Value
Optimizer	`torch.optim.AdamW`
Learning Rate	`5e-4` (with `OneCycleLR` scheduler)
Loss Function	`nn.CrossEntropyLoss` (label_smoothing=0.05)
Max Epochs	15
Early Stopping	Patience of 3 epochs
Batch Size	64
Sequence Length	64
Data Split	90% Train / 10% Val
Embedding Dimension	512
Feed-Forward Dim	2048
Encoder Layers	4
Attention Heads	8
Weight Tying	Applied (Embedding weights tied to final FC layer)

- **Project architecture**



5. Results and Analysis

All models were trained until validation loss failed to improve for 3 consecutive epochs. The best-performing epoch for each model is recorded below. The models started to memorize the data after 8 to 10 epochs for almost all the cases , so early stopping was also included to prevent that from happening .

Model	Validation Loss	Validation PPL
GRU	**4.8097**	**127.98**
BiLSTM	4.8314	134.37
BiGRU	4.8162	136.05
BiGRU + Attn	4.8256	136.98
BiLSTM + Attn	4.8234	137.52
LSTM	4.8948	142.27
Transformer (Base)	4.9786	163.65
Transformer + Head	5.2019	181.61
Transformer + Attn	5.9114	385.97

➢ Analysis of Findings

1. **Best Model**: The **standard unidirectional GRU** was the clear winner, achieving the lowest validation perplexity of **127.98**. This suggests that a simpler, lighter-weight architecture was the most effective and efficient for this specific dataset and task.
2. **GRU vs. LSTM**: In both unidirectional and bidirectional forms, the GRU models consistently outperformed their LSTM counterparts (e.g., GRU PPL 127.98 vs. LSTM PPL 142.27).
3. **Bidirectionality & Attention**: A counter-intuitive finding was that adding **bidirectionality or an attention mechanism consistently *worsened* the model's performance** (i.e., increased perplexity). This may indicate that for this narrative text, the most recent context is a much stronger predictor than the full-sequence context.
4. **Transformer Performance**: The Transformer models performed significantly worse than the RNNs, even with a larger model and more advanced training loop. This is a common finding on smaller, single-book datasets, as Transformers typically require massive amounts of data to learn effectively.

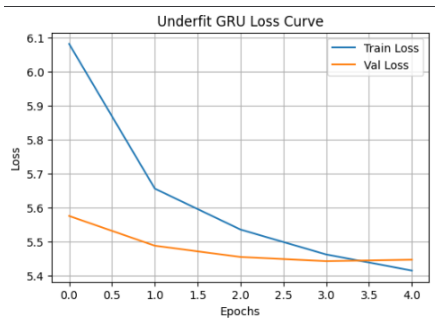
➢ **Sample Generation** : The following text was generated by trained GRU Model

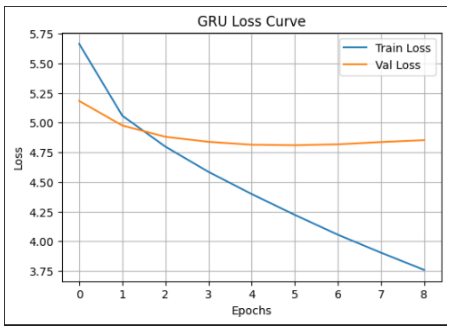
```
🪶 Seed: elizabeth was beginning to feel
✨ Generated: elizabeth was beginning to feel that a woman had been so delighted by her health and that she had
<unk> and so always <unk> a dislike as she could have no more pleasure i have prevented her ladyship to tell her
she is a very
```

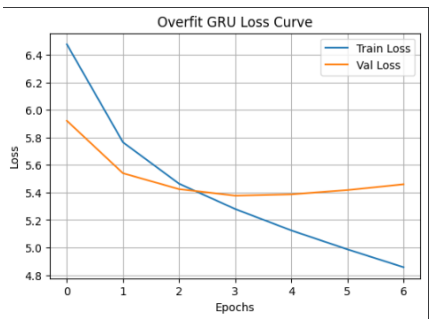
6. Understanding Model Capacity checkpoints

A key part of the assignment was to demonstrate an understanding of model fit by identifying underfitting, overfitting, and a "best fit" model.

- **Underfitting :** An underfit model has not yet learned the patterns in the data. This is characterized by high training and validation loss, where both are still decreasing. The first few epochs of any model's training run demonstrate this.
- **Best Fit :** The "best fit" is achieved at the point where the validation loss is at its minimum, just before it begins to rise. This represents the model's peak generalization performance. Our early stopping mechanism was designed to capture this exact moment. The final GRU model, which **achieved its best performance at Epoch 8**, is the best example of this.
- **Overfitting :** Overfitting occurs when the model begins to memorize the training data, losing its ability to generalize. This is visibly identified when the **training loss continues to decrease** while the **validation loss begins to increase**.







7. Conclusion and rationale for best model

This project successfully implemented and evaluated nine different language models. The results demonstrated that for a medium-sized, single-domain dataset, a simpler architecture like the standard 2-layer GRU provides the best performance.

The GRU (PPL 127.98) was selected as the best model due to its superior perplexity score. More complex additions like bidirectionality and attention, as well as the Transformer architecture, proved less effective, likely due to the limited data size. The training curves clearly illustrated the concepts of underfitting, best fit, and the onset of overfitting, which was effectively managed with early stopping.

8. Links and code

- **Github repository** : https://github.com/Puneeth-Abhishek-6622/Research-Internship-Assignment-2
- **Kaggle Notebook :** https://www.kaggle.com/code/melllogang/lstm-gru-and-transformer-models
- Submitted by – I Puneeth Abhishek , Email : puneethabhishek2004@gmail.com