MiniProject 2 — Language Model

Ndeye Awa Salane

University of Geneva

December 14, 2025

Table of Contents

Project Overview

- Goal: Train a GPT-style neural network from scratch to generate Shakespeare-like text at the character level.
- Dataset: Tiny Shakespeare ($\approx$ 175K characters)
- Frameworks: PyTorch, Weights & Biases ...
- Model Type: Transformer decoder (GPT-like)

Dataset

- Single raw text file containing Shakespeare excerpts: input.txt
- Split:
  - 90% training
  - 10% validation
- Vocabulary: list of all unique characters

```
['!', '$', '-', ';', 'A', 'B', ..., 'z']
```

Encoding & Batching

Character $\leftrightarrow$ Integer:

$$\text{encode}(c) \rightarrow i, \quad \text{decode}(i) \rightarrow c$$

Context window: $\text{block\_size} = k$

$$x = \text{'Ndeye: He'} \rightarrow y = \text{'deye: He '}$$

Batching with random sampling:

- Random starting positions
- Each batch: $(B, \text{block\_size})$

GPT Architecture (Decoder-Only Transformer)

- Token + Positional Embeddings
- N Transformer Blocks:
  - Multi-Head Self-Attention
  - Feed-Forward Network
  - LayerNorm
  - Residual Connections
- Linear head for next-character prediction

$$p(x_{t+1} \mid x_{\leq t}) = \text{softmax}(Wh_t)$$

Training Setup

- Loss: Cross entropy
- Optimizer: AdamW
- Sweep tracked with W&B:
  - Train/Validation loss
  - Perplexity
  - Gradient norms
  - Model checkpoints

Training Loop (Simplified)

Load the best hyperparameters combination from the previous sweep then train our model.

1. Load random batch (x, y)
2. Forward pass through GPT
3. Compute loss
4. Backpropagation
5. Optimizer update
6. Early stopping
7. Periodic evaluation and sample generation

Sample text generated by step

Prompt: *'O God, O God!'*

```
O God, O God!  What light through yonder darkness breaks?  My
heart in tempest beats...
```

Metrics

1. Cross-Entropy Loss
2. Perplexity

Observations

- Overfitting at first, the sweep helped a lot even it ran for about X hours.
- Early stopping when the validation loss was not improving and it's generally at around 3000 steps.
- Captures Shakespeare-like structure:
  - Character names (ROMEO, RICHARD, DUCHESS OF YORK . . .)
  - Dialogue format and patterns
  - Tries to output 'Shakespeare-coherent' text even with my name (very different from the usual text) as an input
- Limitations:
  - Sometimes, words look very much like correct english but are not
  - Same for sentences

Loading Final Model

Load the weights of our best found model and perform inference. Example outputs generated by our model:

Conclusion

- Implemented mini Shakespeare GPT
- Ran a sweeep in W&B for hyperparameter search
- Tracked experiments and artifacts with W&B

GitHub Repository

Project available at:

link text