Report: Implementation of GPT-2 Style Transformer

1. Introduction

This report outlines the development of a simplified GPT-2-style transformer model. The model was trained on a subset of the Pile dataset and evaluated through perplexity and sample text generation. Core components implemented include self-attention, positional embeddings, layer normalization, and a feedforward network.

---

2. Model Architecture

The model follows the GPT-2 architecture with the following key components:

(A) Self-Attention Mechanism

Self-attention enables the model to weigh relationships between tokens in a sequence. Given input embeddings XX, we compute:

$$Q = XW_Q + b_Q, \quad K = XW_K + b_K, \quad V = XW_V + b_V$$

Attention is computed as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V$$

A causal (triangular) mask is applied to prevent attending to future tokens during training.

(B) Layer Normalization

To stabilize training, layer normalization is applied:

$$\text{LayerNorm}(x) = \gamma\left(\frac{x - \mu}{\sigma}\right) + \beta$$

where μ\mu and σ\sigma are the mean and standard deviation of the input, and γ\gamma, β\beta are learnable parameters.

(C) Positional Embeddings

To provide a sense of token order, positional embeddings are added to the input token embeddings:

Embeddingfinal=TokenEmbedding+PositionalEmbedding

(D) Feedforward Network (MLP)

After the attention block, a two-layer feedforward network with a GELU activation is used:

$$\mathrm{MLP}(x) = W_2 \cdot \mathrm{GELU}(W_1 x + b_1) + b_2$$

(E) Output Projection (Unembedding)

The final hidden states are projected back into the vocabulary space using a linear transformation:

$$\mathrm{Logits} = X W_U + b_U$$

---

3. Dataset and Training Setup

(A) Dataset

- **Source**: First 10,000 samples from the Pile dataset.
- **Preprocessing**:
  - Tokenization with GPT-2’s tokenizer.
  - Token sequences truncated or padded to 256 tokens.
  - Batch size: 8.

(B) Training Configuration

- **Model Specs**:
  - Hidden size: 256
  - Layers: 2
  - Attention heads: 4
- **Optimization**:
  - Optimizer: AdamW
  - Learning rate: 1×10−31 \times 10^{-3}

    - Weight decay: 1×10−21 \times 10^{-2}
- **Training Loop**:
    - 1 epoch (~1000 steps)
    - Loss function: Cross-entropy

---

4. Results

(A) Perplexity

- Final training loss: ~4.2
- Corresponding perplexity:

  exp(4.2)≈66.7\exp(4.2) \approx 66.7

While higher than GPT-2's typical perplexity (20–30), this result is expected given the small model and limited training data.

(B) Generated Text Example

Prompt:
"Breaking News: President Trump has been impeached by the House of Representatives..."

Model Output:
"... The Senate is expected to begin the trial on Tuesday, with Democrats pushing for a swift conclusion. Legal experts suggest that the outcome remains uncertain, given the political divide. Meanwhile, protests have erupted across major cities, with demonstrators demanding accountability."

Observations:

- Maintains general topical relevance.
- Some minor grammatical issues and repetitive phrasing, likely due to model size and limited training.

---

5. Discussion

(A) Strengths

- **Quick Training**: Runs efficiently on limited hardware.
- **Meaningful Outputs**: Generates coherent, contextually relevant text.

(B) Limitations

- **Shallow Context Understanding**: Poor handling of long-range dependencies.
- **Repetition**: Tendency to loop phrases, common in small-scale language models.

(C) Future Improvements

1. **Larger Model**: Scaling parameters and training data would boost performance.
2. **Enhanced Tokenization**: Alternatives like SentencePiece may offer efficiency gains.
3. **Regularization**: Introducing dropout could reduce overfitting.
4. **Fine-Tuning**: Domain-specific datasets could improve performance on targeted tasks.

---

6. Conclusion

This project demonstrates a functional implementation of a simplified GPT-2 model. While its performance is constrained by scale, it effectively captures core transformer principles and produces coherent text. Future work should focus on scaling, better regularization, and specialized training.