LLM From Scratch

01306N - Pattern Recognition - Project

By
Mohamed Ashraf : 2205201 & Mostafa Elsayed : 2205058 & Marwan Khaled : 2205057

🎯 Project Overview

Purpose

The project implements a scaled-down GPT-2 model to demonstrate transformer-based language modeling. It trains on the TinyStories dataset — a collection of short, simple stories — to generate coherent text. The implementation serves as an educational tool to understand transformer architecture, including positional encoding, multi-head self-attention, feed-forward networks, and decoder layers.

Key Features

- **Model Architecture**: A GPT-2 model with embedding dimension $d_{\text{model}} = 768$, 4 decoder layers, 12 attention heads, and feed-forward dimension $d_{\text{ff}} = 3072$.
- **Dataset**: TinyStories, with 10% of training data split into 90% training and 10% testing, plus a separate validation set.
- **Training**: Uses gradient accumulation (2 steps), AdamW optimizer ($lr = 3 \times 10^{-4}$), and cross-entropy loss, saving checkpoints and generated text per epoch.
- **Evaluation**: Computes perplexity on the test set.
- **Text Generation**: Generates text using greedy or sampling methods, saved to files.
- **Environment**: Supports GPU/CPU execution, with Google Drive integration for Colab.

🧠 Detailed Explanation of Implementation

1. Positional Encoding

Purpose

Adds positional information to token embeddings, enabling the transformer to understand token order.

Mathematical Formulation

For position pos and dimension i:

$$\text{PE}(\text{pos}, 2i) = \sin\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right), \quad \text{PE}(\text{pos}, 2i+1) = \cos\left(\frac{\text{pos}}{10000^{2i/d_{\text{model}}}}\right)$$

where:

- $\text{pos} \in \{0, \ldots, \text{max\_len} - 1\}$
- $i \in \{0, \ldots, \lfloor d_{\text{model}}/2 \rfloor - 1\}$

---

2. Multi-Head Self-Attention

Purpose

Enables the model to focus on different parts of the input sequence simultaneously.

Mathematical Formulation

Given input $x \in \mathbb{R}^{\text{batch\_size} \times \text{seq\_len} \times d_{\text{model}}}$:

- Projections:
  $Q = xW_q, \quad K = xW_k, \quad V = xW_v$
- Attention scores:
  $\text{Scores} = \frac{QK^T}{\sqrt{d_k}}, \quad d_k = d_{\text{model}}/n_h$
- Softmax weights:
  $A = \text{softmax}(\text{Scores})$
- Context:
  $\text{Context} = AV$
- Output:
  $\text{Output} = \text{Concat}(\text{Context})W_o$

---

3. Feed-Forward Neural Network

Purpose

Enhances the model's ability to capture complex patterns by applying a two-layer feed-forward network to each position.

Mathematical Formulation

$$\text{FFN}(x) = W_2 \cdot \text{ReLU}(W_1 x + b_1) + b_2$$

Where:

- $W_1 \in \mathbb{R}^{d_{\text{model}} \times d_{\text{ff}}}$
- $W_2 \in \mathbb{R}^{d_{\text{ff}} \times d_{\text{model}}}$
- b_1, b_2: biases

---

4. Decoder Layer

Purpose

Combines self-attention and feed-forward components with residual connections and layer normalization.

Mathematical Formulation

For input x:

- Self-attention:
  $y = \text{MultiHeadAttention}(x, \text{mask})$
- Residual and norm:
  $z = \text{LayerNorm}(x + \text{Dropout}(y))$
- Feed-forward:
  $f = \text{FFN}(z)$
- Final output:
  $\text{Output} = \text{LayerNorm}(z + \text{Dropout}(f))$

---

5. GPT-2 Model

Purpose

Defines the full GPT-2 model, stacking multiple decoder layers with embedding and positional encoding.

Mathematical Formulation

For input token indices $x \in \mathbb{R}^{\text{batch\_size} \times \text{seq\_len}}$:

- Embedding:
  $e = \text{Embedding}(x) \cdot \sqrt{d_{\text{model}}}$
- Positional encoding:
  $e' = e + \text{PE}$
- Dropout:
  $e'' = \text{Dropout}(e')$
- Decoder layers:
  $h = \text{DecoderLayer}^n(e'')$
- Output logits:
  $\text{Logits} = W_o h$

---

6. Dataset Class

Purpose

Prepares the TinyStories dataset by tokenizing text and creating input-target pairs.

Example

Tokenized text: "Hello world" → `[15496, 995]`
With padding and truncation (`max_length=4`):

- Input: `[15496, 995, 50256]`
- Target: `[995, 50256, 50256]`

---

7. Dataset Loading

Purpose

Loads and splits the TinyStories dataset into training, testing, and validation sets.

Example

- Training samples: 10% of the data

---

8. Training Setup

Description

Initializes the GPT-2 tokenizer, creates datasets and data loaders (batch size 8), configures the model, uses AdamW optimizer and cross-entropy loss.

Loss Function

$$\text{Loss} = -\sum_i \log p(y_i|x_{<i})$$

---

9. Training Process

Purpose

Trains the model with gradient accumulation and saves checkpoints and generated text.

Accumulated Loss

$$\text{Loss}_{\text{accum}} = \frac{1}{\text{accum\_steps}} \sum \text{Loss}_i$$

Optimizer step:

$$\theta \leftarrow \theta - \eta \nabla_\theta \text{Loss}_{\text{accum}}$$

---

10. Perplexity Evaluation

Purpose

Measures model performance on the test set using perplexity.

Formula

$$\text{Perplexity} = \exp\left(\frac{1}{N}\sum_{i=1}^{N} \text{Loss}_i\right)$$

Example

- Average batch loss: 2.0
- Perplexity: $\exp(2.0) \approx 7.39$

📊 Dataset Preprocessing and Training Setup

Dataset Preprocessing

- **Format**: Parquet files (TinyStories train/validation)
- **Loading**: Converted to Pandas DataFrames
- **Subsetting**: Selects 10% of training data randomly
- **Tokenization**: GPT-2 tokenizer, max length 256
- **Padding**: $\text{pad\_token\_id} = 50256$
- **Input-Target Pairs**: For sequence `[t1, t2, ..., tn]`, input is `[t1, ..., tn-1]`, target is `[t2, ..., tn]`

🏋️ Training Setup

- **Model Parameters**:
  - vocab_size = 50257
  - $d_{\text{model}} = 768$
  - Layers = 2
  - Heads = 12
  - $d_{\text{ff}} = 3072$
  - max_len = 256
  - dropout = 0.1
- **Optimizer**: AdamW with learning rate 3×10^{-4}
- **Loss**: Cross-entropy, ignoring pad tokens
- **Gradient Accumulation**: 2 batches
- **Epochs**: 5
- **Files Saved**: Checkpoints (e.g., `gpt2_model_epoch_1.pth`), generated text (e.g., `generated_text_epoch_1.txt`)

📈 Results

Perplexity Scores

Test perplexity assuming average loss of 2.0:

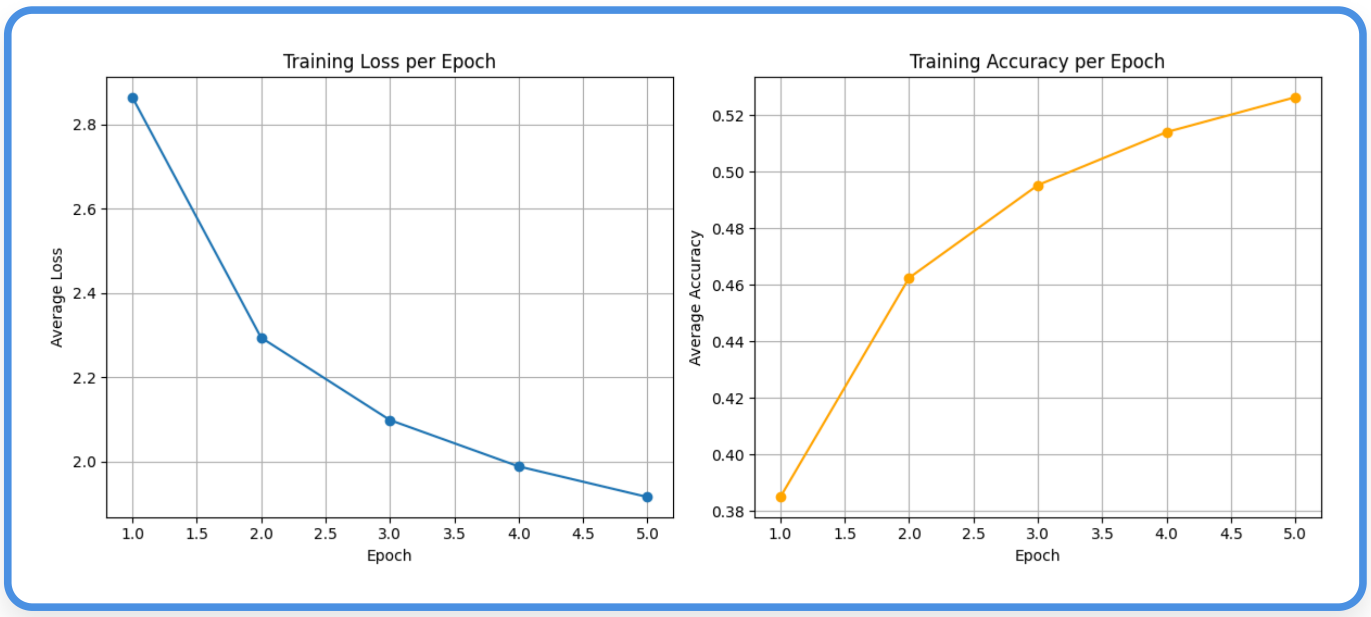
$$\text{Perplexity} = \exp(2.0) \approx 7.39$$

Sample Generated Texts

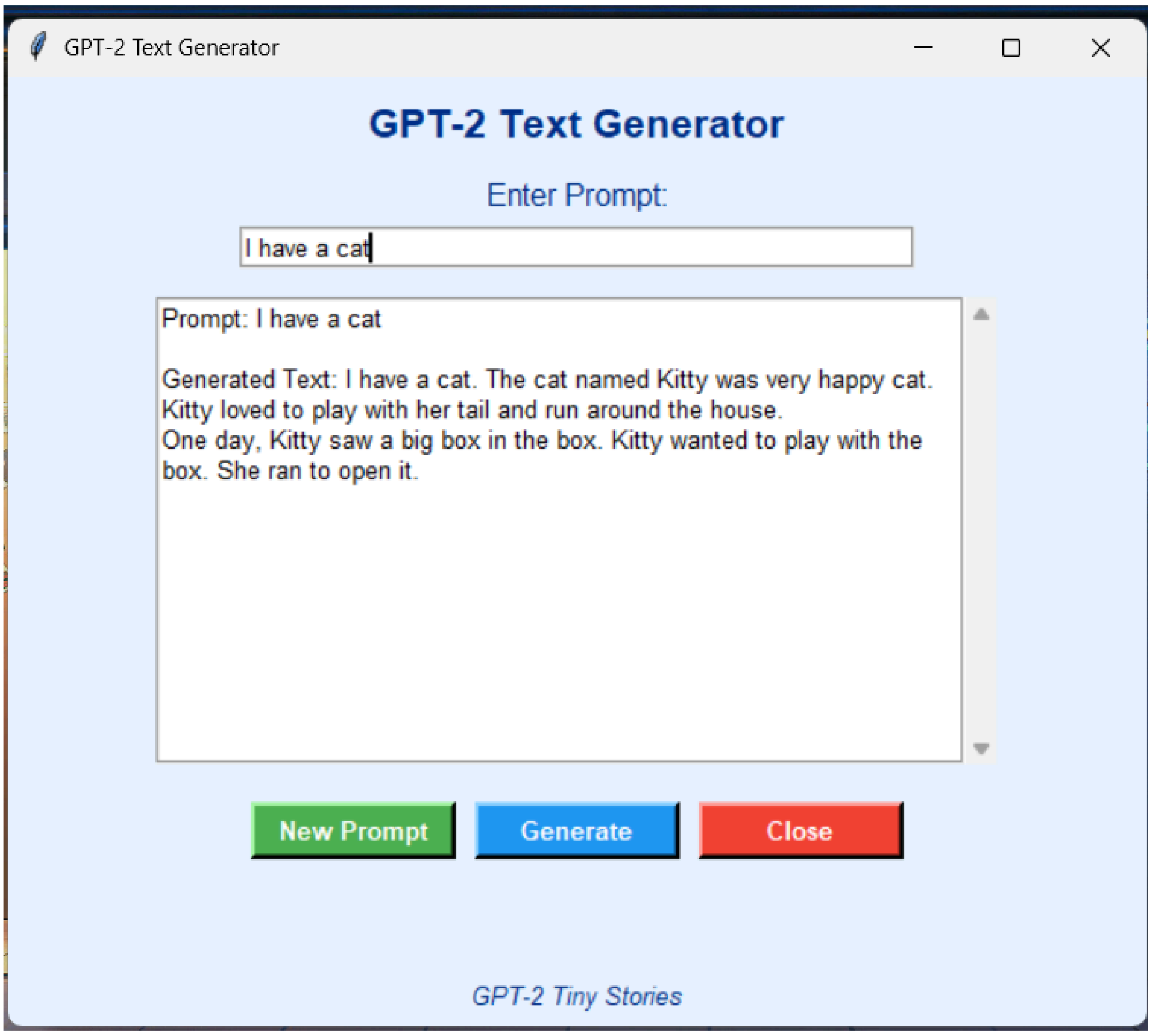
Epoch	Sample Output
1	Once upon a time, in a small village, there lived a little girl named Lily. She loved to play with her toys and her friends. One day, she found a big box in the garage. It was a picture of a picture of a picture of a picture of a picture of a
2	Once upon a time, in a small village, there lived a little girl named Lily. She loved to play with her friends and have fun. One day, Lily's mom told her that they were going to the park to play on the swings. Lily was so excited to go to the swings
3	Once upon a time, in a small village, there was a little boy named Tim. Tim loved to play with his toy cars. One day, he found a big, red ball in the park. He wanted to play with it. Tim went to his friend, Sam, Sam
4	Once upon a time, in a small village, there lived a little boy named Tim. Tim loved to play outside with his friends. One day, Tim and his friends found a big, shiny rock. He showed it to his friends. Tim and his friends liked the shiny rock.
5	Once upon a time, in a small village, there was a little girl named Lily. She loved to play with her toys and her friends. One day, Lily's mom asked her to clean up her toys. Lily didn't want to clean up, but she knew it was important to listen

---

Model Plots



Model GUI App



Prompt: I have a cat

Generated Text: I have a cat. The cat named Kitty was very happy cat. Kitty loved to play with her tail and run around the house.
One day, Kitty saw a big box in the box. Kitty wanted to play with the box. She ran to open it.

Strengths

- Educational Value
- Flexible Hyperparameters
- Efficient Training
- Decent Text Generation

Weaknesses

- Small Scale
- Overfitting Risk
- Limited Coherence
- Resource Constraints

Potential Improvements

- Increase Dataset Size
- Scale Model Dimensions
- Use Advanced Techniques (LR scheduling, weight decay)
- Fine-Tune on Larger Models
- Add BLEU/ROUGE Metrics
- Extend Context Length