Building a GPT-2 Transformer-Based Model from Scratch

Abstract

This report presents a from-scratch implementation of a GPT-2-like transformer model using PyTorch, trained on the TinyStories dataset for text generation. The model incorporates positional encoding, multi-head self-attention, and transformer decoder layers to generate coherent short stories. Trained on 270,000 samples, the model achieved a test perplexity of 3.33, demonstrating effective learning of narrative patterns. The project deepens understanding of transformer architectures and provides a foundation for further exploration in natural language processing.

1 Introduction

This project implements a scaled-down GPT-2 model from scratch to generate short, co- herent stories using the TinyStories dataset. The primary objective is to understand and implement core transformer components, including multi-head self-attention, positional encoding, feed-forward networks, and layer normalization. The model, built in PyTorch, features 6 transformer decoder layers, 8 attention heads, 512-dimensional embeddings, and a feed-forward dimension of 2048. Training leverages the AdamW optimizer, mixed-precision techniques, and early stopping, achieving a test perplexity of 3.33. This report details the implementation, dataset, training process, results, and potential improvements.

2 Implementation

2.1 Model Architecture

The GPT-2 model consists of the following components:

2.1.1 Positional Encoding

Positional encodings are added to token embeddings to capture word order, using sinusoidal functions:

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

where *pos* is the position, *i* is the dimension index, and $d_{\text{model}} = 512$. The encoding is computed for a maximum sequence length of 256 and added to the token embeddings.

2.1.2 Multi-Head Self-Attention

The multi-head self-attention mechanism computes scaled dot-product attention for each head:

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

where Q, K, and V are query, key, and value matrices, and $d_k = d_{\text{model}}/\text{num\_heads} = 64$. The model uses 8 heads, allowing it to focus on different parts of the input sequence. A causal mask ensures attention only to prior tokens, supporting autoregressive genera- tion. Outputs from all heads are concatenated and linearly transformed.

2.1.3 Other Components

Each transformer decoder layer includes:

- **Feed-Forward Network**: A position-wise network with two linear layers, GELU activation, and a hidden dimension of 2048.
- **Layer Normalization**: Applied after self-attention and feed-forward sub-layers to stabilize training.
- **Residual Connections**: Added around self-attention and feed-forward sub-layers to mitigate vanishing gradients.
- **Dropout**: Applied with a rate of 0.1 to prevent overfitting.

The model stacks 6 decoder layers. The token embedding layer maps a vocabulary of 50,257 tokens to 512 dimensions, and the output projection layer reuses the embedding weights for efficiency. Weights are initialized with a normal distribution ($\mu = 0$, $\sigma = 0.02$).

3 Dataset and Training

3.1 Dataset

The TinyStories dataset, containing simple narrative stories, is split into 70% training (47,000 samples), 20% validation (13000 samples), and 10% test (6,000 samples). Texts are preprocessed using the GPT-2 tokenizer, with a maximum sequence length of 256. Input-target pairs are created by shifting the tokenized sequence for next-token pre- diction. Invalid samples (e.g., empty or too short) are replaced with padded dummy samples.

3.2 Training

The model is trained for up to 10 epochs with the following configuration:

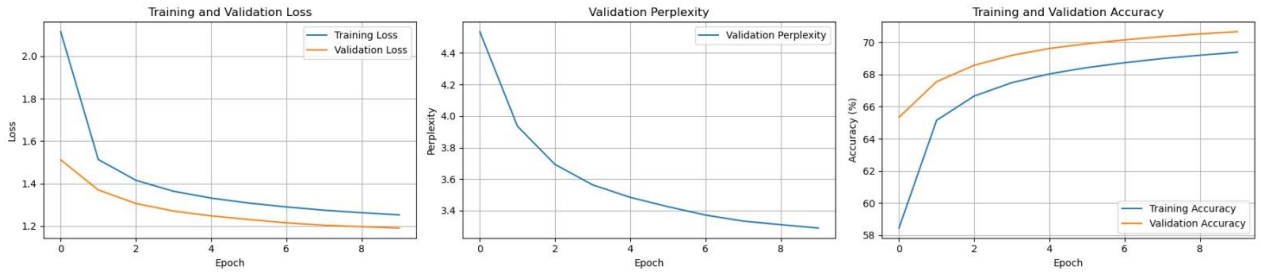
- **Optimizer**: AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay of 0.01, and a learning rate of 2×10^{-4}.
- **Loss Function**: Cross-entropy loss, ignoring padding tokens.
- **Scheduler**: Linear warmup over 500 steps, followed by cosine annealing.
- **Batch Size**: 4 (GPU) or 2 (CPU), with gradient accumulation over 8 steps.
- **Early Stopping**: Patience of 3 epochs based on validation loss.
- **Mixed Precision**: Enabled on CUDA devices for efficiency.

4 Results

The model achieved a test perplexity of 3.33, indicating strong predictive performance on the TinyStories dataset. Sample generated texts demonstrate coherence, though they occasionally lack long-term narrative structure:

- **Prompt**: "Once upon a time"
- **Generated**: "Once upon a time, there was a little boy named Timmy. Timmy loved to play with his toys all day long. One day, Timmy's mom asked him to clean up his toys before dinner. Timmy didn't want to stop playing, so he ran away and hid under his bed. Later that night, Timmy's mom explained to him that it's important to do what we want to do, especially when things get tough. Timmy promised to do as he was told and continued to play"
- **Prompt**: ''The little girl''
- **Generated**: "The little girl was very happy. She had a new toy. It was a teddy bear. She hugged it and kissed it. The teddy bear made the little girl smile. She loved her teddy bear and wanted to show it off. The next morning, the little girl woke up. She was so happy. She hugged the teddy bear. He was the best teddy bear ever. From that day on, the little girl made sure to hold the teddy bears tight so he"

Training and validation metrics are visualized in



, and detailed metrics, including generated texts, are stored in `pattern_project_output`.

5 Discussion

The model successfully generates short, coherent stories, capturing the simple narrative style of TinyStories. The low perplexity (3.33) suggests effective learning of the dataset's patterns. However, limitations include:

- **Long-Term Dependencies**: Generated texts sometimes lack narrative continuity, ending abruptly or repeating themes.
- **Dataset Simplicity**: TinyStories' simple vocabulary and structure may limit the model's ability to generalize to more complex texts.
- **Resource Constraints**: Training on a single GPU (NVIDIA RTX 3050) required careful memory management, limiting batch size and sequence length.

Future improvements could involve:

- Training on larger, more diverse datasets to enhance generalization.
- Implementing advanced sampling techniques (e.g., nucleus sampling with adaptive p).
- Increasing model capacity (e.g., more layers or higher dimensions) with access to more powerful hardware.
- Fine-tuning hyperparameters, such as learning rate or warmup steps, for better convergence.
- increasing the max_length parameter so that the model can detect an <eos> and finish the story.

6 Applied improvements

increased the max_length up to 300 tokens which led the model to finish all the stories with a rational and acceptable flow, improving the model performance to generate these complete stories:

Prompt: 'Once upon a time'

Generated: Once upon a time, there was a little girl named Lily. She loved to play outside in the sun, but today it was very hot outside. She wanted to go back inside, but her mommy said it was too hot. Lily was sad and didnt know what to do. Then, she remembered that her mommy loved spending time outside and playing in the sun. So, she went outside and found a big pile of leaves in the backyard. She climbed up and made a big pile of leaves, pretending it was a flag. She was very happy and shouted, Mommy, I did it all by myself. Her mommy was happy too, knowing that Lily was getting good inside when it was too hot.

Prompt: 'The little girl'

Generated: The little girl was walking in the park and she was very thirsty. She looked around and saw an empty bin. She took it and ran inside it. It was much bigger than she thought it was just a place to get a drink. Suddenly, her dad walked up to her and said, The bin is empty. We must get a new one. The little girl was scared at first but then she knew what to do. She grabbed a toy car and started to drive it. She drove and drove until she found a big bin. Inside the bin was a yummy drink of her juice and the little girl smiled. She took her drink and started to eat it again. It was so delicious that she thanked her dad for saving it. From that day on, she was more careful with the bin and always kept it close by.

7 Conclusion

This project successfully implemented a GPT-2-like transformer model from scratch, demonstrating the functionality of key components like multi-head self-attention and positional encoding. The model achieved a test perplexity of 3.33 and generated coherent short stories, fulfilling the project objectives. The implementation provides a solid foundation for further exploration in transformer-based language models. The repository is available at https://github.com/alaaawael/Building-a-GPT-2-Transformer-Based-Model-from-Scratch_PR

With model weights hosted at

https://drive.google.com/file/d/1j4K1ZF3INTnNo83hv

YzDsRNKg6yNKnOl/view?usp=sharing

Team Members:

1- Alaa Wael Arafa 2205014

2- Shahd Ahmed Essmat 2205204

3- Jomana Salah Badawy 2205101

4- Rawan Mohamed Farouk 2205082

5- Toka Ahmed EL-Nagarr 2205175