COMP 560 Final Project

Bennett Shoenbill
COMP 560.002
Jorge Silva
April 2025

Abstract

I present a lightweight LoRA-based fine-tuning of a TinyLlama-Chat model on a 8 436-example "wizardry" NPC interaction dataset. By crafting a clean prompt–completion JSONL schema and leveraging Google Colab's T4 GPU for 8-bit LoRA adapters, I achieved coherent, stylistically consistent fantasy dialogue, reducing perplexity on the dataset by over 75 %. I will contrast this with failed DialoGPT experiments under CPU constraints, and discuss dataset generation via ChatGPT plus manual curation.

1 Introduction

Fine-tuning large language models allows them to adopt domain-specific 'tones' such as NPC dialogue for games, offering both a playful application and a hands-on learning project in parameter-efficient tuning. My aim was to build a conversational style generator that responded to a prompt as if it were a dialogue box in a video game, namely EarthBound, known for its distinct, witty, surreal dialogue; this process functioned as a sandbox for exploring how LLMs can learn artistic "style." This eventually evolved into a general fantasy themed style-generator, but one that still had a unique voice, and still adapted the same dialogue-box-styled responses.

2 Related Work

LoRA Adapters Low-Rank Adaptation (LoRA) freezes the pre-trained weights and injects small rank-decomposition matrices into attention layers, reducing trainable parameters by orders of magnitude while matching full fine-tuning quality [1]. IBM provides an accessible overview of LoRA's efficiency gains [2], and Microsoft's PyTorch integration is available on GitHub [4].

Prompt–Completion Schemas Instruction-tuned LLMs benefit from clear delimiters between user prompt and model response, as demonstrated in Alpaca and other chat-tuned models.

3 Dataset Creation and Cleaning

I generated 8 436 fantasy interaction examples by prompting ChatGPT in batches and manually validating each chunk for style, consistency, and relevance. If a batch contained inconsistent punctuation, was overly consistent, or just didn't adapt well to the desired style, it was thrown out. Using Python, I then normalized all leftover quotes, dashes, and apostrophes to ASCII and reformatted to a '`prompt`'/'`completion`' JSONL schema with '### Response:' delimiters.

4 Fine-Tuning Pipeline

4.1 Google Colab Setup

I used a free Colab T4 GPU, accelerating 8-bit LoRA via bitsandbytes. After installing 'transformers', 'peft', and 'datasets', I loaded TinyLlama-1.1B-Chat, applied a LoRA adapter (rank = 8) via PEFT, and trained for two epochs. While Colab only allows for a certain amount of GPU usage, the necessary amount of fine-tuning was completed before that number was met.

4.2 Data Preprocessing

Each example was tokenized into 'input_ids', 'attention_mask', and masked 'labels', then batched with a custom 'pad_sequence' collator to align tensor dimensions.

4.3 Mathematical Formulation

To efficiently adapt TinyLlama to the 8 436-example wizard-dialogue dataset, I employed Low-Rank Adaptation (LoRA) [1]. In each attention layer, the original weight matrix $W \in \mathbb{R}^{d\times d}$ is frozen, and we learn two much smaller matrices $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times d}$ (with $r = 8$ in this context) so that

$$W' \;=\; W + B\,A,$$

where $r \ll d$ reduces the number of trainable parameters from $\mathcal{O}(d^2)$ to $\mathcal{O}(2dr)$. For TinyLlama's hidden size $d = 2048$, this meant updating only $\sim 0.8\%$ of the original weight count, allowing me to fine-tune a free Colab T4 in less than an hour without GPU memory overflow [1, 2].

After training, I evaluated how well each model assigns likelihood to held-out NPC interactions using perplexity, defined as

$$\mathrm{PPL}(\mathbf{x}) = \exp\Big(-\tfrac{1}{N}\sum_{i=1}^{N}\log p(x_i)\Big),$$

where x_i are the token IDs in the model's generated response and N is the total number of these tokens. Here we compute PPL only over the *completion*

tokens—masking out the prompt—so that the metric reflects the adapter's ability to produce coherent fantasy dialogue rather than simply reconstructing the input [3]. This is discussed further in the next section.

5 Results

5.1 Perplexity on Held-Out Responses

We evaluate perplexity only over the completion tokens (masking prompts) on our 843-example held-out split. Table 1 shows that fine-tuning reduced PPL by over 75%.

Model	PPL (completions only)
TinyLlama-1.1B-Chat (base)	154.2
TinyLlama-1.1B-Chat + LoRA adapter	36.3

Table 1: Perplexity on held-out wizard interaction data.

As noted previously, perplexity was measured as

$$\text{PPL} = \exp\Big(-\tfrac{1}{N}\sum_{i=1}^{N}\log p(x_i)\Big),$$

computed only over the generated response tokens.

5.2 Output Demonstration

Prompt: Player interacts with a glowing dragon

Fine-tuned TinyLlama:
I'm not sure what you want. But if you don't ask, I may never answer. And that would be unfortunate for all. Or so it seems. *snort* *shakes head* *dragon flaps wings and disappears into the night sky.*

Base TinyLlama:
"Wow, that sounds incredible! I'm so excited to see what this game can do. But before we dive in, could you tell me more about the different types of players? Are there any who prefer single-player experiences or those who thrive on competition?"

Figure 1: Comparison of base vs. fine-tuned responses for a "glowing dragon" prompt.

5.3 Qualitative Outputs

Comparing base vs. fine-tuned responses to held-out prompts shows that only the adapter retains format adherence and fantasy tone—base outputs frequently

hallucinated JSON errors or off-topic chatter. Although both responses in 5.2 are somewhat coherent, Base TinyLlama often resorted to hallucinations that would ultimately consume far too large a portion of this report to fully showcase.

6 Failures and Lessons Learned

Initial DialoGPT-Small attempts under CPU constraints produced incoherent, repetitive replies even when using established dialogue datasets. Originally, I tried creating a dataset involving the video game EarthBound's dialogue lines, which included manually typing out context for 878 lines of dialogue. This was ultimately a failure, resulting in slightly stylized gibberish. DialoGPT-Medium was also used later on, and similarly failed to converge, highlighting GPU necessity and the advantage of instruction-tuned bases, along with not using models that were over four years old.

7 Conclusion and FutureWork

This project has demonstrated that small, parameter-efficient adapters can successfully imbue LLMs with rich, stylistic NPC dialogue when combined with clean schemas and GPU-accelerated LoRA. Future work could include quantizing merged adapters for desktop 'llama.cpp' deployment and A/B user studies for subjective style evaluation.

References

[1] Edward Hu, Yelong Shen, Philip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Weizhu Chen, and Yuanzhuo Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[2] IBM. What is lora (low-rank adaptation)? IBM THINK, 2024.

[3] Daniel Jurafsky and James H. Martin. *Speech and Language Processing*. Pearson, 2023.

[4] Microsoft Corporation. Lora: Low-rank adaptation implementation for pytorch. `https://github.com/microsoft/LoRA`, 2021.