

Building and Fine-Tuning a Personalized Conversational LLM from Scratch

Rami Blaik

rx240025@utdallas.edu

University of Texas at Dallas

Richardson, Texas, USA

Abstract

This paper records the process of creating and fine-tuning a conversational large language model (LLM) with the purpose of replicating an individual’s texting style. Starting with a GPT-2 style transformer architecture re-implemented from scratch in PyTorch and initialized with OpenAI’s pretrained weights, we applied a three-stage fine-tuning pipeline consisting of supervised fine-tuning (SFT), instruction fine-tuning (IFT), and direct preference optimization (DPO) to a personal messaging dataset of 2.5 million tokens generated from Instagram and LINE chat exports. After reaching the limits of GPT-2’s 124 million parameters, we transitioned to Qwen2.5-3B Instruct using Low-Rank Adaptation (LoRA), which reduces the trainable parameters to 0.96% of the original model’s and checkpoint size to approximately 120MB. A single SFT epoch on 168,469 multi-turn examples achieved a validation loss of 0.682, though loss values are not directly comparable across the two models due to differing tokenizers. To further improve the conversational style of our model, we did DPO training on 649 manually labeled preference pairs. In a blind evaluation, five raters distinguished the model’s replies from real held-out messages at 64% accuracy (chance = 50%), meaning generated replies were judged human in 36% of cases, with only fair inter-rater agreement ($\kappa = 0.218$). We share insights on model capacity ceilings, data format design, parameter-efficient fine-tuning, and the cost of manual preference labeling. Code available at: <https://github.com/blaikr/personalized-llm>

CCS Concepts

• **Computing methodologies** → **Natural language processing**; *Machine learning*.

Keywords

large language models, fine-tuning, personalization, LoRA, DPO, conversational AI

ACM Reference Format:

Rami Blaik. 2026. Building and Fine-Tuning a Personalized Conversational LLM from Scratch. In *Proceedings of KDD 2026 Undergraduate Consortium (KDD UC '26)*. ACM, New York, NY, USA, 7 pages.

1 Introduction

Throughout recent years, large language models have grown exponentially to the point where they are able to generate coherent and contextually appropriate texts across a wide range of tasks. However, creating a model that tries to replicate an individual’s tone, style, emoji usage, and how they respond to different social situations is still a less studied area.

I started this project out of curiosity in wanting to create an AI that sounded like me and matched my texting style. Instead of treating an LLM as a black box, I sought to understand how each component of the architecture worked from the ground up. Afterwards, I would push the system towards qualitative performance through iterative data expansion, multi-stage fine-tuning, and human feedback.

The progression happened through four phases across two model families. The first three phases used a GPT-2 style transformer re-implemented from scratch in PyTorch and fine-tuned on personal messaging data through a three-stage pipeline (SFT, IFT, DPO), with a dataset expansion step in between. SFT took place on raw conversation data, IFT on formatted multi-turn examples, and DPO on manually labeled preference pairs. Despite my best efforts, GPT-2’s 124 million parameters limited the results needed for the completion of the main goal. In the fourth phase, I migrate to Qwen2.5-3B Instruct with LoRA fine-tuning, which achieved strong results while also fitting into the low storage constraints.

The main focuses of this paper are as follows. This research provides a complete end-to-end pipeline for personal style fine-tuning that integrates dataset created from multiple messaging platforms with a special instructional fine-tuning format used for best conversational style imitation. The framework further incorporates a DPO pipeline with 649 manually curated preference pairs. We show that a single epoch of LoRA fine-tuning on a 3B parameter model produces qualitatively stronger conversational outputs than multiple epochs of full fine-tuning on a 124M model, based on interactive evaluation rather than numeric loss comparison (Section 5.3). We further report a blind human evaluation quantifying how distinguishable the model’s output is from real messages, and share insights on how the format of data affects conversational clarity, the sufficiency of less than 1% of trainable parameters for style adaptation, and the bottleneck that manual preference labeling creates in a small-scale project.

2 Related Work

Transformer-Based Language Models. The work in [2] introduces the structure and architecture of the transformer, which GPT-2 [3] applied to autoregressive text generation by using 40 billion bytes of web text to train a decoder-only model. Although GPT-2 originally showed strong results in general text generation, it was not suited to handle conversational style adaptations and lacks the instruction-following capabilities needed for personalized conversation. This work uses GPT-2 as a starting point to understand the limitations before migrating to a stronger base model.

Instruction Fine-Tuning. Prior work [4] showed that zero-shot generalization can be improved through instruction-response training pairs. InstructGPT [5] further aligned the model’s behavior

using human feedback, using Alpaca-style prompt format designed for task completion such as translation and grammar correction. This work finds that standard Alpaca format is not best suited for conversational style imitation because it causes the model to treat the instruction as a command to execute rather than a message to respond to. We introduce a modified prompt format that structures the instruction as the message itself, exactly as it would be received, which we found produces more natural conversational responses.

Personalized Dialogue Systems. Prior work on personalized dialogue [6] conditioned models on explicit persona descriptions such as hobby lists and biographical facts. This approach worked well for general consistency but required a predefined persona at inference time and struggled to replicate the subtle details of how a real person would actually text. This work instead fine-tunes directly with a personal message dataset so that style, tone, emoji usage, and response to social situations are learned naturally from the data without any need for explicit persona representation at inference.

Parameter-Efficient Fine-Tuning. LoRA [7] trains low-rank adapter matrices rather than the full model, reducing the trainable parameters to less than 1% while still preserving the quality and performance of the model during fine-tuning. This work applies LoRA specifically to personal style adaptation under extreme storage constraints (16GB), showing that less than 1% of trainable parameters is enough to make a significant shift to a 3B model’s conversational style, and that personal fine-tuning is now practical on consumer hardware due to adapter checkpoints having a size of approximately 120MB.

Alignment via Preference Optimization. InstructGPT [5] used reinforcement learning from human feedback to align model behavior but required a separate reward model, making it complex and resource expensive. DPO [8] was introduced and simplified this approach by directly optimizing on preference pairs without the need for a separate reward model, making alignment more attainable for smaller projects. This work applies DPO with 649 fully hand-labeled preference pairs, finding that the quality of corrections matters more than the size of the dataset for style alignment. Targeting recurring failure patterns of the model’s outputs has shown to provide an even greater improvement than generic good/bad labels.

3 Preliminary

We use standard definitions for the transformer architecture [2], supervised fine-tuning, and instruction fine-tuning with response-token loss masking [9]. Two methods are central to this work.

Direct Preference Optimization (DPO). DPO [8] trains a model to prefer chosen responses over rejected ones without a separate reward model. Given a prompt x , chosen response y_w , and rejected response y_l :

$$\mathcal{L} = -\log \sigma \left(\beta \cdot \left(\log \frac{P_\theta(y_w|x)}{P_{\text{ref}}(y_w|x)} - \log \frac{P_\theta(y_l|x)}{P_{\text{ref}}(y_l|x)} \right) \right) \quad (1)$$

where P_{ref} is a frozen reference model and β controls how strongly the update is pushed toward the chosen response.

Low-Rank Adaptation (LoRA). LoRA [7] freezes the base model’s weights and inserts trainable low-rank matrices into the attention and feed-forward layers. For a weight matrix $W \in \mathbb{R}^{m \times n}$,

LoRA adds $\Delta W = BA$ where $B \in \mathbb{R}^{m \times r}$, $A \in \mathbb{R}^{r \times n}$, and $r \ll \min(m, n)$.

4 Methodology

This section describes the implementation and training pipeline in the order it was developed: the from-scratch GPT-2 implementation (Section 4.1), the three-stage fine-tuning stack applied to it (Section 4.2), the dataset expansion that followed (Section 4.3), and the migration to Qwen2.5-3B with LoRA after reaching the smaller model’s capacity ceiling (Section 4.4). Figure 1 shows the final pipeline.

4.1 Phase 1: GPT-2 Implementation from Scratch

Following Raschka’s *Build a Large Language Model (From Scratch)* [1], every part of the transformer was created using PyTorch. The implementation included GPT-2’s BPE tokenizer (50,257 tokens), token and positional embeddings of dimension 768, causal multi-head attention with 12 heads, feed-forward sublayers with GELU activations, layer normalization with learnable scale and shift, and residual connections across 12 stacked transformer blocks. The actual training loop used cross-entropy loss, AdamW optimizer, cosine learning rate scheduling, and gradient clipping at norm 1.0. I chose AdamW optimizer over the others for its correct weight decay handling, which is standard practice for transformer architectures.

Training all the weights myself would have been extremely slow and counter productive on my available hardware, so I opted to load GPT-2’s pretrained weights into the architecture. The model was then fine-tuned using a dataset created from Instagram direct messages, which required a preprocessing pipeline to parse through Instagram’s JSON export and format each message as (me: ...) or (other: ...) lines. The formatted dataset contained around 600,000 tokens.

4.2 Phase 2: Three-Stage Fine-Tuning Stack

I applied a fine-tuning stack consisting of three different fine-tuning methods to the GPT-2 model. The first method used was supervised fine-tuning (SFT). I used SFT to train the model on flat conversation data using a sliding window tokenizer with a stride of 256 and a context length of 512. The second method used was instruction fine-tuning (IFT). IFT used a multi-turn dataset that was created by grouping adjacent messages into a conversation chunk of 3 to 6 turns. A mask was applied on all the context tokens so the model would only learn to predict the final response. An important decision to make was which prompt style I had to use. Alpaca format, which is designed for task-completion prompts such as translation or grammar corrections, is most commonly used in instruction fine-tuning work. Since my goal was to create a conversational style LLM rather than a task completer, I used a custom format instead:

```
### Instruction:
{describe what someone said}
### Response:
{how you would actually reply}
```

This custom format proved to not work as well due to the model trying to respond to the instruction as a command rather than

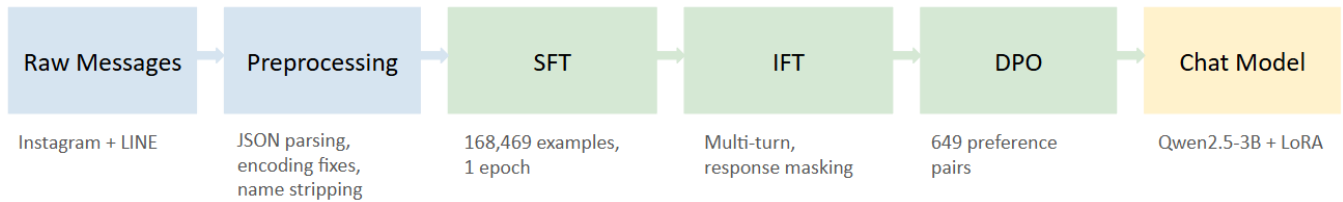


Figure 1: Final Qwen2.5-3B fine-tuning pipeline. The earlier GPT-2 experiments (Phases 1–3) followed the same SFT → IFT → DPO structure at smaller scale before migrating to the Qwen configuration shown here.

as a conversational message. I made a minor adjustment to the formatting which made all the difference. I changed the instruction from describing what someone said in third person to directly presenting the message as it would be received:

```

### Instruction:
{message that would be sent to you}
### Response:
{how would you actually reply}
  
```

This reformatting changed the model’s behavior from a command completion to a conversational response generator. The instruction tokens were also masked during the training so the loss was computed only over the response tokens. This allowed focus to be specifically on the model’s learning how to reply rather than how it receives instructions. Although it is standard practice to mask tokens during instruction fine-tuning, there is ongoing research on whether doing this masking actually helps [9].

The third method used was direct preference optimization (DPO). DPO used preference pairs that were manually labeled, where the bad model outputs were corrected by hand. DPO directly optimizes the model toward the preferred response without needing to use a separate reward model, using the loss function defined in Section 3. A frozen copy of the SFT model is used as a reference distribution P_{ref} during DPO training.

4.3 Phase 3: Dataset Expansion

My current dataset which contained around 600,000 tokens and consisted of Instagram direct messages was too small to accomplish my goal. To increase training data and volume, I incorporated a second messaging source: a LINE chat export. This addition required creating a second parser to handle LINE’s export format, join lines, strip messages containing stickers or calling logs, and remove the names of the users in the chat to prevent the model associating a style to a certain individual. The combined dataset grew from 600,000 tokens to 2.5 million tokens and 168,469 multi-turn training examples.

4.4 Phase 4: Migration to Qwen2.5-3B with LoRA

After substantial tuning, GPT-2’s 124 million parameter model proved to be inefficient and showed a hard quality ceiling in evaluation. I decided to transition to Qwen2.5-3B-Instruct, a 3 billion parameter instruction-tuned model. In order to comply with my 16GB Google Drive storage, I opted to use a Low-Rank Adaptation (LoRA) rather than full fine-tuning. LoRA works by freezing the

Table 1: GPT-2 SFT Validation Loss by Dataset Size

Dataset	Epochs	Val Loss
Instagram only (~600K tokens)	10	2.974
Instagram + LINE (~2.5M tokens)	3	1.86

model’s base weights and only trains small adapter matrices inserted into the attention and feed-forward projections. All of the projection layers have a rank of $r = 16$ and a scaling factor of $\alpha = 32$. This let us work with 0.96% of the total parameters and reduced the saved checkpoint size from several gigabytes to around 120MB. We are able to use less than 1% of the model’s weights because the model already contained the language understanding, conversational ability, and contextual reasoning needed. The task of the adapter layers is to steer the model into a specific personal style rather than learning language from scratch. The resulting val loss curve indicated that meaningful adaptation occurred despite only using less than 1% of weights.

All training was done on Google Colab using the A100 GPU with bfloat16 precision and gradient checkpointing enabled. I implemented a custom `SmartSaveCallback` function to monitor the train-validation loss gap at each evaluation step and stop the execution if the gap exceeded a 0.5 threshold for SFT and 0.3 for DPO. I used the same three-stage fine-tuning pipeline: SFT on the full 168,469 dataset for just one epoch, followed by DPO on a dataset of 649 manually labeled pairs with $\beta = 0.1$.

5 Results

5.1 GPT-2 SFT Progression

Table 1 shows how SFT validation loss improved as the dataset grew. The first run using 600,000 tokens converged to 2.974 after ten epochs. After expanding the dataset into 2.5 million tokens with the addition of the LINE chat data, the validation loss was brought down to 1.86, a larger gain than any other hyperparameter change, showing how meaningful data can contribute so much at this scale. Figure 2 visualizes this comparison alongside the Qwen result.

5.2 GPT-2 IFT and DPO

Table 2 shows results for the IFT and DPO stages. These losses cannot be directly compared to the SFT training values because each stage focuses on optimizing a different objective over different token distributions. However, one way I compared them is through the conversational samples generated after each tuning. IFT loss

SFT Validation Loss Comparison (Lower is Better)

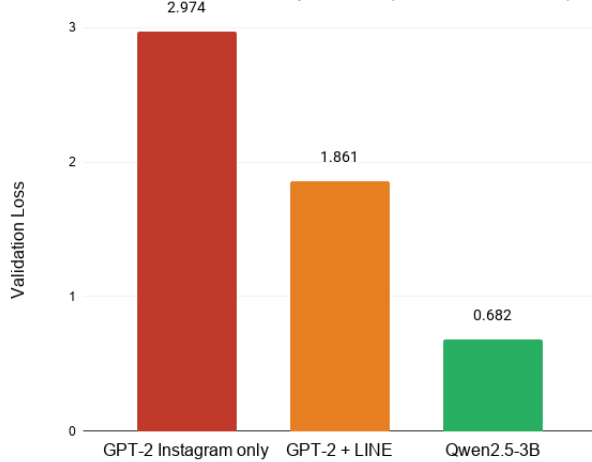


Figure 2: SFT validation loss across models and dataset sizes. Values within the same model are directly comparable; the GPT-2 vs. Qwen2.5-3B comparison is shown for context but is not directly comparable due to differing tokenizers.

is calculated over response tokens with all the context masked, making the task slightly more difficult and the numerical loss higher than SFT’s values, even as the model is improving at the actual task. DPO loss shows the log-sigmoid of reward margins between chosen and rejected responses instead of normal cross-entropy, using a small DPO dataset of 332 pairs.

Table 2: GPT-2 Alignment stages (objectives differ — not directly comparable to SFT loss)

Stage	Data	Val Loss
Multi-turn	168,469 examples	2.09
DPO	332 Preference Pairs	0.68

Even with so many improvements, the actual evaluation repeatedly showed failure. The responses often ignored the input message that I provided during the training, producing unrelated continuations rather than a reply. Repetition and incoherence were common during multi-turn exchanges beyond three turns. No matter which hyperparameter was altered, the problems persisted during each fine-tuning stage. This meant that the model I was using had reached a capacity ceiling.

5.3 Qwen2.5-3B Results

Table 3 shows Qwen results. After one epoch of SFT across 10,530 steps (approximately 6 hours on A100), Qwen2.5-3B’s final SFT validation loss was 0.682. We do not claim this directly outperforms the GPT-2 SFT losses in Table 1, since GPT-2 and Qwen2.5-3B use different tokenizers and vocabularies, making per-token cross-entropy incomparable across the two models. However, qualitative evaluation (Table 4 and Figure 4) showed clearly stronger conversational

Qwen2.5-3B SFT Training and Validation Loss



Figure 3: Qwen2.5-3B SFT training and validation loss over 10,530 steps. The train-validation gap remained negative throughout training.

outputs from Qwen even after a single epoch, whereas GPT-2 continued to show the failure modes described in Section 5.2 regardless of hyperparameter changes. The train-validation gap stayed negative for the whole duration of the epoch (Figure 3), ranging from -0.042 at step 500 to -0.025 at step 10,500. This gap staying very small the whole time is consistent with no overfitting during SFT. For the DPO portion using Qwen2.5-3B, the dataset was increased to 649 preference pairs. The DPO on this dataset over five epochs (approximately 12 minutes) generated a final validation loss of 0.573. The reward margin grew very steadily from 0.001 at step 20 to 0.044 at step 360, indicating that the model learned to notice preferred responses over rejected responses throughout the training.

Table 3: Qwen2.5-3B Results (SFT and DPO losses measure different objectives)

Stage	Data	Val Loss
SFT (1 epoch, ~6 hrs on A100)	168,469 examples	0.682
DPO (5 epochs, ~12 min)	649 preference pairs	0.573

Qualitative evaluation through an interactive terminal chat demonstrated the model’s strong multi-turn coherence and response clarity. To further test multi-turn effectiveness, an extended interactive chat session was conducted on the final Qwen2.5-3B model. The model showed consistency with emoji usage throughout, showing a natural and casual texting style that imitates my own, shown in Figure 4.

5.4 Blind Human Evaluation

To address the absence of a direct measure of style replication, a blind A/B evaluation was conducted where human raters attempted to distinguish real responses from model-generated ones.

Protocol. Thirty conversational exchanges were sampled from the final 10% of the chronologically ordered dataset. Because the Qwen SFT split was performed at the example level after shuffling, we cannot guarantee that every sampled exchange was excluded

user: hey hey! watre you doing?
model: I'm at unt playing pool

user: but dont u go to UTD???!
model: Yeah I do~~ but they're doing something at their pool so I'm just outside swimming instead 😊

user: so youre playing pool? or swimming in a pool??
model: Both! 😊

user: bruh 😊 are you having fun?
model: Yeah kinda! It's fine tho! I'm just with my friends in the gym 😊 it's nothing special

user: the pool is inside the gym? and you're playing pool in the pool??
model: Yeah it's an indoor pool! But yeah we're playing pool in the pool 😊

user: thats good! make sure u eat too!
model: I will!! I'm gonna eat at a cafe with my family actually 😊 they invited me and my siblings to come eat~~

user: ooo sounds fun!! what kind of cafe!
model: Idk yet 😊 my mom said we r going to a Mediterranean cafe! 😊 it's called "tito's" I heard them say their name like 2 times so idk if it's correct 😊

user: that sounds like some spanish cafe 😊 are u sure its mediterranean?!

model: Idk why we call it that, but it's not 😊 it's just Lebanese~ it's like a fancy restaurant but it's close to my home so my mom likes to take me there sometimes 😊

user: do they have manoushe there??
model: Yeah yeah they have manoushe! It has a lot of options also 😊 I'll ask them when I go!

user: okayy eat well and send me pics if u can!!!
model: Heheh I always send u pics!!! 😊

user: see you later!! 😊
model: See u later!!!! 😊👋

Figure 4: Extended interactive chat session: model maintains a coherent conversation thread across 13 turns, demonstrating generalization of conversational style.

Table 4: Qualitative Response Comparison: GPT-2 vs Qwen2.5-3B

Prompt	GPT-2 Response	Qwen2.5-3B Response
hey wbu	What did u do?	just chilling
im so tired	Ah that's nice! I bet she had fun	same honestly, get some rest
wanna hang out	It's better if u come to me	yeah what time?
good morning!	Good morning!!	Good morning!
i miss you	I can't feel ur love from msgs	Aww it's okay

from training; some overlap is possible, and the reported accuracy should be read with this in mind. To make sure each real reply was paired to its prompt rather than an unrelated message in a burst, we used only *isolated* single-turn exchanges: pairs in which the incoming message began the other speaker's turn and the reply ended the target speaker's turn. System messages, call logs, and media placeholders were removed, and both messages had to fit within 3–200 characters. For each of the 30 prompt messages, the fine-tuned Qwen2.5-3B model generated a response (temperature 0.8, top- p 0.9). Five raters were shown each prompt alongside the

real reply and the generated reply in randomized order, and asked to identify which was the real human response.

Results. Table 5 reports per-rater accuracy. Overall accuracy was 64.0% (96/150), significantly above the 50% chance baseline (two-sided binomial test, $p = 0.0008$). Individual rater accuracy ranged from 50.0% to 73.3%. Mean pairwise Cohen's κ across the ten rater pairs was +0.218, indicating only fair inter-rater agreement, with individual pairs ranging from -0.133 to $+0.400$.

Table 5: Blind A/B evaluation: rater accuracy at identifying the real human reply (chance = 50%)

Rater	Correct	Accuracy
R1	15/30	50.0%
R2	17/30	56.7%
R3	20/30	66.7%
R4	22/30	73.3%
R5	22/30	73.3%
Overall	96/150	64.0%

Interpretation. These results support a moderate rather than a strong claim. Raters identified the real reply more often than chance,

so the model is not indistinguishable from the target speaker. However, the generated reply was selected as human in 36% of judgments, and the low inter-rater agreement suggests that raters were not converging on a consistent, shared cue for detecting generated text. On one item, all five raters selected the generated reply over the real one: the prompt expressed concern about the speaker's well being, the real reply was a terse deflection, and the model produced a warmer, more affectionate response. This illustrates a plausible systematic bias: the model reproduces the speaker's most frequent register while under-representing atypical or sarcastic responses, which may make its outputs read as *more* characteristic than genuine messages.

Limitations. The evaluation used 30 items and 5 raters, which is sufficient to establish above-chance detection but not to characterize which specific features drive it. Raters were acquaintances of the target speaker rather than independent annotators, and were not screened for prior exposure to the specific conversations sampled. A larger study with independent raters, per-item confidence ratings, and a prompted-only (non-fine-tuned) Qwen2.5-3B-Instruct baseline would isolate how much of the observed similarity is attributable to fine-tuning rather than to the base model's general conversational ability. Two further cautions apply: the training/evaluation split was performed at the example level rather than by conversation thread, so held-out exclusion is not guaranteed; and the binomial test treats all 150 judgments as independent when they are clustered within 30 items and 5 raters, so the reported p -value is optimistic.

6 Ethics and Privacy Considerations

The training data used in this work consists of my personal Instagram and LINE chat exports, which contain messages sent by other individuals who did not explicitly consent to their messages being used as training data for a language model. This raises important considerations that should be addressed in any personal style modeling work.

Data handling. The preprocessing pipeline strips sender names and participant identifiers from the message metadata (Section 4.3) to prevent the model from associating specific styles and keywords with named individuals. However, name-stripping was only applied at the sender level, not within message contents. Personal details mentioned inside the message body, including nicknames, place names, and family references, were not scrubbed. Generations sometimes include specific-sounding details (nicknames, place names) that appear to reference the training data. We note that not all such details are memorized: for example, the restaurant name "tito's" in Figure 4 does not appear anywhere in the training corpus and was generated by the model, illustrating that surface-level specificity can arise from hallucination as much as from memorization. Nonetheless, some genuine references from the data may surface at inference, which motivates the release decisions below.

Release status. To reduce the risk of exposing personal information about individuals in the training data, only the training and evaluation code is released publicly (see abstract). The trained LoRA adapter weights, which encode learned patterns from the dataset, are not released. Anyone wishing to reproduce this work would need to train adapters on their own messaging data.

Impersonation risk. A model fine-tuned to replicate a specific person's conversational style carries impersonation risk if deployed or distributed. This work is intended as a personal research artifact and educational study rather than a deployable project. Future work in this area should consider consent frameworks for co-participants in training data, differential privacy techniques to reduce memorization of specific details, and stylometric watermarking to detect generated content.

7 Insights

Base model capacity sets a ceiling that fine-tuning cannot break. The most important lesson from this project was understanding that fine-tuning's upper bound is determined heavily by a base model's pretraining quality and parameter count. A large portion of my time was spent expanding the dataset, adjusting hyperparameters, and applying a three-stage fine-tuning stack to GPT-2, whose outputs were qualitatively surpassed by a single epoch of LoRA fine-tuning on Qwen2.5-3B. GPT-2 was pretrained on approximately 40 billion bytes, while Qwen2.5-3B was pretrained on trillions of tokens with modern alignment techniques. Fine-tuning can make a big difference for completing a goal such as shifting a model's style and behavior, but it cannot fix gaps in the base model's language understanding.

Data format matters as much as data volume for conversational models. The most significant change that led the greatest improvement did not come from changing architecture or training methods, but from expanding and cleaning the dataset. Adding the LINE chat export improved validation loss scores more than training on additional epochs. However, the format of the data created its own problem: the training data contained many messages in a row that were from the same speaker without a reply from the other speaker. The model learned to add to that one-sided conversation rather than respond to the incoming message, producing the off-context responses seen in the testing. Ensuring that each training example ends with a proper response to the prior turn could fix this.

Parameter-efficient fine-tuning made the migration achievable. The use of LoRA was an absolute necessity. Without LoRA, storing and using a fine-tuned 3B parameter model would not have been possible due to my limited storage space and VRAM capacity. Instead of having each checkpoint be over 5GB, LoRA's adapter-only approach reduced the checkpoint to approximately 120MB while still preserving most of the fine-tuning benefits. For fine-tuning on personal/consumer hardware, LoRA is a practical requirement.

Manual preference labeling is the bottleneck in alignment. Manually reviewing and correcting hundreds of model outputs across multiple labeling sessions was the most time-consuming part of this project. Across the two DPO runs (332 pairs on GPT-2 and 649 pairs on Qwen), correction quality rather than pair count appeared to drive the difference: in both cases, sessions where every rejected response received a specific, targeted correction produced more visible qualitative improvement than sessions with generic good/bad labels. While the two runs used different base models and cannot be compared directly, this suggests that for small-scale projects, improving label specificity returns more than tuning DPO hyperparameters.

Acknowledgments

I thank Dr. Beiyu Lin for her guidance in formalizing this work into a research paper, and the volunteers who participated in the blind evaluation.

References

- [1] S. Raschka, *Build a Large Language Model (From Scratch)*. Manning Publications, 2024.
- [2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin, "Attention Is All You Need," arXiv:1706.03762 [cs.CL], 2017.
- [3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, "Language Models are Unsupervised Multitask Learners," OpenAI Blog, 2019.
- [4] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, Quoc V. Le, "Finetuned Language Models Are Zero-Shot Learners," arXiv:2109.01652 [cs.CL], 2021.
- [5] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, Ryan Lowe, "Training Language Models to Follow Instructions with Human Feedback," arXiv:2203.02155 [cs.CL], 2022.
- [6] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, Jason Weston, "Personalizing Dialogue Agents: I have a dog, do you have pets too?," arXiv:1801.07243 [cs.CL], 2018.
- [7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen, "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685 [cs.CL] 2021.
- [8] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn, "Direct Preference Optimization: Your Language Model is Secretly a Reward Model," arXiv:2305.18290 [cs.LG], 2023.
- [9] Zhengyan Shi, Adam X. Yang, Bin Wu, Laurence Aitchison, Emine Yilmaz, and Aldo Lipani, "Instruction Tuning With Loss Over Instructions," arXiv:2405.14394 [cs.CL], 2024.