

Data Efficient Fine-Tuning and Alignment of GPT-2

Stanford CS224N Default Project

Aryaman Gupta
Aeronautics and Astronautics
Stanford University
aryamann@stanford.edu

Zeyuan Feng
Aeronautics and Astronautics
Stanford University
zeyuanf@stanford.edu

Joseph Lee
Aeronautics and Astronautics
Stanford University
jjlee25@stanford.edu

Abstract

Large language models (LLMs) often require substantial task-specific data and human-annotated preferences to achieve high performance and alignment, limiting their applicability in data-constrained settings. In this project, we investigate data-efficient fine-tuning strategies for GPT-2 on sonnet generation and paraphrase detection tasks. For sonnet generation, we address data constraints by first pre-training on large datasets, followed by supervised fine-tuning (SFT) and preference alignment using synthetic data generated with a comprehensive scoring mechanism. We also explore other avenues, such as adding special prefix tokens for explicit output conditioning and different output sampling strategies. For paraphrase detection, we perform parameter-efficient fine-tuning (PEFT) with different parameter subsets and ranks, and full fine-tuning across different GPT-2 model variants to identify the best adaptation strategies. In the sonnet generation task, our proposed method achieved a **test ChrF score of 42.218**. For the latter, we present all training comparisons and ultimately obtain the best performance with PEFT on the GPT-2 (large) model, yielding a **test accuracy of 0.862**.

1 Key Information to include

- Mentor: Advit Deepak External Collaborators: No Sharing project: No
- Late Days:

Team member	Late days remaining
Aryaman Gupta, Zeyuan Feng	0
Joseph Lee	3
Total pooled	3
Team size	3
Late days applied for the report	1

2 Introduction

Large Language Models (LLMs) are fundamentally constrained by the volume and quality of training data. The modern paradigm of training in natural language processing (NLP) involves a two-stage process: pre-training a base model on massive, diverse datasets for general linguistic understanding, followed by fine-tuning on a domain-specific dataset for downstream tasks. However, this paradigm faces two critical bottlenecks: First, fine-tuning LLMs requires retraining on large datasets, which consumes significant GPU resources and time. Second, resource limitations exist in specific domains where high-quality open-source data is scarce, leading to limited adaptation and ultimately lower performance. The NLP community has prioritized research in this "efficiency-performance" trade-off, discovering techniques that lower training costs while boosting performance.

Popular fine-tuning methods include PEFT, such as Low-Rank Adaptation (LoRA) [1], which freezes the pre-trained model weights and injects trainable rank-decomposition matrices into the transformer layers. This significantly reduces the number of training parameters and computational requirements

during fine-tuning, while maintaining comparable performance to full fine-tuning methods. LLMs are further aligned with human preferences through preference optimization methods such as reinforcement learning from human feedback (RLHF) and direct preference optimization (DPO) [2]. These approaches typically rely on human-labeled preference datasets to guide optimization. However, obtaining large-scale, high-quality preference annotations is costly and limits scalability.

Our work evaluates different fine-tuning methods for two specific downstream tasks: paraphrase detection and sonnet generation. For paraphrase detection, we leverage a large-scale dataset to adapt different model parameter subsets using LoRA and compare the performance with full fine-tuning methods across different model sizes and LoRA ranks. For sonnet generation, (i) we first pre-fine-tune on a large poetry dataset to mitigate the small data constraint, (ii) develop a synthetic preference data generation framework using rule-based scoring mechanisms, and perform DPO for further refinement. (iii) We add special prefix tokens to the sonnets for better rhyme consistency through explicit conditioning, and (iv) evaluate various output sampling and constraint decoding methods.

3 Related Work

Parameter-Efficient Fine-Tuning (PEFT). Full fine-tuning (Full FT) is the brute-force method of simply retraining every single parameter, which can be billions for larger models, thus has a clear bottleneck of high memory and compute costs of updating all these weights. PEFT methods operate on the principle that the observations that highly optimized models typically lie on a low-dimensional intrinsic manifold [3]. LoRA [1] is an exemplary work that leverages this hypothesis by adding low-rank matrices to the model parameters and updating only these matrices.

Alignment and Preference Optimization. Reinforcement learning from human feedback (RLHF) aimed at aligning LLMs with human preferences by training a reward model with human feedback [4]. Proximal Policy Optimization (PPO) algorithm is widely used in RLHF to update models using human preferences to generate higher-reward responses without shifting too far from the original model [5]. RLHF has major limitations as it requires learning a reward model from a labeled dataset. DPO [2] is considered an advancement in LLM alignment because it eliminates the need to train a reward model, making it much more scalable, but it still requires labeled preference datasets.

Output Sampling Strategies. Constrained text generation involves enforcing specific structural linguistic requirements during the autoregressive generation process, such as adding special control tokens that condition the further generation [6, 7], or logit masking, which prunes the model’s output space based on predefined rules [8]. While these methods are effective for rigid syntactic constraints, they often encounter a "fluency-constraint trade-off" in creative domains. For tasks like sonnet generation, procedural constraints must contend with byte-pair encoding (BPE) fragmentation, but tokenization creates a misalignment between the phonetic requirements of a rhyme and the model’s token distribution. Our work evaluates different unconstrained and constrained sampling schemes to perform a comprehensive evaluation.

4 Approach

4.1 Paraphrase Detection

For paraphrase detection, we fine-tune the GPT-2 model using both full fine-tuning (full SFT) and LoRA SFT to compare their performance. We add LoRA adapters to different subsets of parameters in the set $W_q, W_k, W_v, W_o, W_{mlp}$, using different LoRA ranks, and with different GPT-2 models (base, medium, and large), to perform a complete comparison of fine-tuning methods across different axes. We format the model input as a question asking whether the two sentences are paraphrases of one another, and minimize the cross-entropy (CE) loss on "yes" and "no" using the provided labels. Having access to a large dataset, suitable for efficiently adapting more parts of the model, we want to compare performance differences across all methods and pick the best-performing one.

4.1.1 Low-Rank Adaptation (LoRA)

LoRA [1] operates on the principle that during fine-tuning, the changes to model’s weights are confined to a low-dimensional space, despite the model containing a large number of parameters. This concept is supported by the observations that highly optimized models typically lie on a low-dimensional intrinsic manifold [3]. More concretely, if the pre-trained weight matrix $W_0 \in R^{d \times k}$, and ΔW is the gradient update, then it is restricted to being a matrix of

lower rank: $W_0 + \Delta W = W_0 + BA$, where $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$ (Fig. 1). Here, the rank r is generally much smaller than the rank of the full pre-trained weight matrix W_0 . The matrices A and B are initialized as $A_{ij} \sim \mathcal{N}(0, \sigma^2)$ and $B_{ij} = 0$, ensuring $\Delta W = 0$ at the beginning of training. Throughout training, W_0 remains unchanged while A and B include tunable parameters. ΔW is scaled by $c = \alpha/r$, where α is the scaling factor and c controls the contribution of the low-rank adaptation to the overall model.

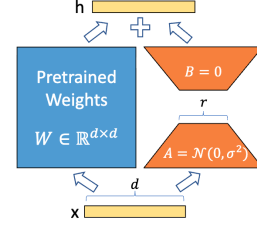


Figure 1: LoRA design

Sonnet generation requires the model to learn to generate multiple sonnet sequences while maintaining correct rhyme, from small datasets, making it an extremely complex task that needs effective solutions. We perform two complementary paradigms: (1) pretraining the model on large-scale poetry corpora [9], followed by full SFT [1] on the Shakespeare sonnet dataset; and (2) DPO [2] by augmenting training data via a synthetic preference data generation approach inspired by the "West-of-N framework" [10].

We (i) compare the performance gains and (ii) further enhance performance by encouraging rhyme scheme consistency using special prefix tokens [6, 7]. We also explore other avenues, (i) different output sampling strategies for generating creative text, and (ii) a dictionary search for generating sets of possible words with correct rhyme to constrain output generation.

4.2.1 Prefix/Control Tokens

To better instruct the model to generate correct rhymes from the given small dataset, we add special control tokens at the beginning of each line and at the start and end of each sonnet. Learning on a formatted training dataset that includes these control tokens enables explicit conditioning on specific tokens while autoregressively generating further output [7]. They act as instructions at each step, constraining the model w.r.t the control token while generating the next output token. By defining special tokens for each rhyme and line, we aim to get enhanced rhyme structure in the generated sonnets.

4.2.2 Output Sampling Strategies

LLMs use different sampling strategies while generating an output token, depending on the task at hand. Different methods have strengths in generating creative outputs, which can be controlled by tuning hyperparameters. We compare three popular methods (i) top-p nucleus sampling [11]: selects tokens with cumulative probability greater than a threshold, (ii) top-k sampling [12]: samples randomly among the "k" highest probability tokens, and (iii) beam search [13]: keeps the most probable output sequences, generates next token, filters again to keep top sequences, and repeats.

4.2.3 Dictionary-Aided Search

To enforce the rhyme structure of the sonnet, we implement a post-hoc logit masking mechanism to constrain generation to rhyme. This method is embedded in between the raw logit calculation and the token selection of the decoding pipeline. We utilize a phonetic dictionary to map the final word of a line to its corresponding rhyme class based on the final phoneme. For a subsequent line that is designated to a rhyme class, we implement a constraint such that the tokens in vocabulary $\mathcal{V}$ belong to this rhyme class. At the final position of the line, we apply a mask to the logits such that only rhyming tokens are candidates for the downstream sampling.

4.2.4 DPO Variants

For preference alignment, we explored four DPO variants, aiming to address the performance limitations of standard DPO in structured sonnet generation:

- **Standard DPO:** The baseline objective trains the (SFT) model by maximizing the likelihood of a chosen completion y_+ over a rejected completion y_- , relative to a reference model π_{ref} . The loss function is defined as follows

$$\mathcal{L}_{DPO}(\pi_\theta; \pi_{ref}) = -\mathbb{E}_{(x, y_+, y_-) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_+ | x)}{\pi_{ref}(y_+ | x)} - \beta \log \frac{\pi_\theta(y_- | x)}{\pi_{ref}(y_- | x)} \right) \right],$$

where β controls how much the trained model deviates from the reference model.

- **DPO-Positive (DPOP):** A variant designed to mitigate the “model collapse” issue often seen in standard DPO. DPOP penalizes log-probability drops on the preferred completions compared to the reference model. This encourages the model to maintain the probability of generating high-quality outputs rather than merely widening the probability gap between generating y_+ and y_- while allowing both to drop significantly.

$$\mathcal{L}_{\text{DPOP}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_+, y_-) \sim D} \left[\log \sigma \left(\beta \left(\log \frac{\pi_\theta(y_+ | x)}{\pi_{\text{ref}}(y_+ | x)} - \log \frac{\pi_\theta(y_- | x)}{\pi_{\text{ref}}(y_- | x)} - \lambda_{\text{DPOP}} \cdot \max \left(0, \log \frac{\pi_{\text{ref}}(y_+ | x)}{\pi_\theta(y_+ | x)} \right) \right) \right) \right]$$

- **Co-training with DPO(P) + SFT:** As an alternative to maintain quality during training, we co-train the model using a multi-objective loss: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{DPO(P)}} + \lambda_{\text{SFT}} \mathcal{L}_{\text{SFT}}$. Importantly, the SFT loss is computed w.r.t. the preferred completions y_+ , which are not necessarily human-authored sonnets from the training dataset. This forces the model to learn preferences while modeling the ground-truth sonnet distribution to prevent quality drops.

4.2.5 Data Generation

The performance of DPO is highly sensitive to the quality and diversity of the preference pairs. We therefore collected our dataset using a hybrid approach:

Ground Truth (GT) Preference Dataset. The GT dataset was constructed using the training dataset that consists of human-authored sonnets. For each data point, the human sonnet serves as the preferred sample y_+ while the corresponding rejected sample y_- is generated by the reference model. These pairs provide a clear signal for the model to learn high-quality outputs, especially given the reference model’s performance is significantly lower than the human standard.

Synthetic Preference Dataset. To increase diversity and prevent the model from memorizing fixed strings, we implemented a synthetic generation pipeline. The goal is to generate diverse prompts and responses without relying on advanced external LLMs (e.g., GPT-5.2) for prompt generation.

1. **Prompt Sampling:** 75% of the prompts are derived from the first three lines of the GT dataset to ensure a high prompt quality, while the remaining prompts are novel single-line starts. The single-line prompts are generated by sampling a full sonnet from the reference model (using a GT prompt) and extracting the most linguistically valid line (excluding the first three lines).
2. **Automated Labeling:** For each prompt x , the reference model generates four candidate sonnets. Candidates are ranked using a custom reward function evaluating rhyme schemes, line counts, word diversity, and indentation. The highest-scoring completion is labeled as preferred y_+ while the lowest-scoring completion is labeled as y_- .

5 Experiments

5.1 Data

We directly leverage the provided Quora and Shakespeare sonnets datasets for the downstream tasks. Particularly for sonnet generation, for pre-fine-tuning on general internet poetry datasets, we use the Kaggle Poems Dataset [9], containing around 20.7K poems across different categories and styles, to train the base model to generate rhyming poetic outputs. This internet poetry dataset consisted of 66 Shakespeare sonnets, including two that were present in the provided testing dataset and were filtered out before training to prevent data leakage, using a combined manual and automated filtering process.

5.2 Evaluation method

For paraphrase detection, the primary test metric of interest is accuracy on both the training and validation datasets. We also present the cross-entropy loss during training on both datasets. For sonnet generation, the test metric of interest is chrF (character-level n-gram F-score).

5.3 Experimental details

Table 1 contains all the hyperparameters and model details as used in our experiments for both downstream tasks. For LoRA implementation, we used the HuggingFace PEFT library. Everything

else in our project is self-developed, though we referred to open-source codebases for easier and more efficient implementation.

Model	Embedding d	Layers l	Heads h
GPT-2 Base (125M)	768	12	12
GPT-2 Medium (357M)	1024	24	16
GPT-2 Large (777M)	1280	36	20

Training Setup	
Batch size	32
Learning rate	1×10^{-5}
Epochs	20
GPU (Base)	1 $\times$ NVIDIA RTX 5090 (32GB)
GPU (Medium, Large)	2 $\times$ NVIDIA A6000 (96GB)
GPT-2 Large training time	$\sim$ 15 hours (10 epochs)

Hyperparameter	Value
Epochs	200
Batch Size	8
Learning Rate	1×10^{-5}
Top-p	0.9
Temperature	1.2
Top-k	50
Beam Size	5
Training Time (GPT (large))	$\sim$ 1.5 hr

LoRA Adapter Placement (GPT-2 Large)	Parameters (%)
(W_q, W_k, W_v)	0.284%
(W_q, W_k, W_v, W_o)	0.379%
$(W_q, W_k, W_v, W_o, W_{mlp})$	0.754%

Table 1: Model configurations, training setup, and decoding hyperparameters for the experiments for paraphrase detection (left) and sonnet generation (right).

Control Tokens. We define new control tokens, update the tokenizer to recognize them as individual tokens, and adjust the embedding size to incorporate them. The used control tokens are:

<SONNET_START>, <SONNET_END>, <LINE_X>, and <R_Y>.

where "X" in "<LINE_X>" is the line number $X \in \{1, 2, \dots, 14\}$, and "R_Y" are rhyme tokens defined for the individual rhyme $Y \in \{A, B, C, D, E, F, G\}$. The line and rhyme tokens are added in beginning of every new line in the sonnet dataset. The metrics were computed after removing prefix tokens from the generated sonnets for fair comparison.

DPO. We use $\beta = 0.1$, $\lambda_{DPOP} = 0.5$, and $\lambda_{SFT} = 0.2$ for all DPO variants. Each variant is trained separately on the full preference dataset and the synthetic-only dataset. A total of 3000 GT preference data points and 5000 synthetic data points were generated, which took around 4 hours to complete. No iterative preference dataset generation was performed since there is no clear improvement after the first DPO iteration.

Dictionary-aided Search. We developed a RhymeFilter class utilizing the CMU Pronouncing Dictionary (CMUDict) to extract and store the required phonetic mappings [14]. The filter performed an exhaustive computation of the full GPT-2 vocabulary to identify all tokens matching the phonetic ending.

5.4 Results

5.4.1 Paraphrase Detection

Fig. 2 presents all the training comparisons with the dev (validation) dataset accuracy metric. For an extensive evaluation of fine-tuning methods, we train all possible combinations and compare their performance on dev accuracy. From (a), LoRA works better as more and more parameters are adapted, performing best while adapting both attention and MLP weights, but still doesn't match full SFT performance. From (b), LoRA adapters added to both attention and MLP weights with increasing ranks bring no significant change, as they all show very similar behavior. From (c), larger GPT-2 models work better when used with LoRA adapters on more parameters. From (d), as expected, full SFT works best when done with larger models. Appendix A.1 presents a more extensive evaluation using the loss metric as well, on both training and validation datasets.

The obtained results provide new insights into LoRA performance. While LoRA matches the performance of full SFT in many cases, here the latter clearly surpasses it, achieving an accuracy of ~ 0.88 , whereas LoRA achieves ~ 0.86 when applied to both attention and MLP weights. LoRA performing the same across increasing ranks suggests that the task at hand is low-complexity and resides in low-intrinsic ranks. Lastly, larger GPT-2 models perform better, both when adapted with

LoRA and full SFT. We achieve accuracies of **0.895** on **Dev** dataset and **0.862** on **Test** dataset using **LoRA model on GPT-2 large with rank=8** (higher than TA baselines of 0.882 and 0.856), which was chosen for its smoother convergence behaviour over full fine-tuned models (Appendix A.1). Full FT on GPT-2 (large) yielded a Dev accuracy of 0.907 but only 0.66 on Test set, signaling overfitting.

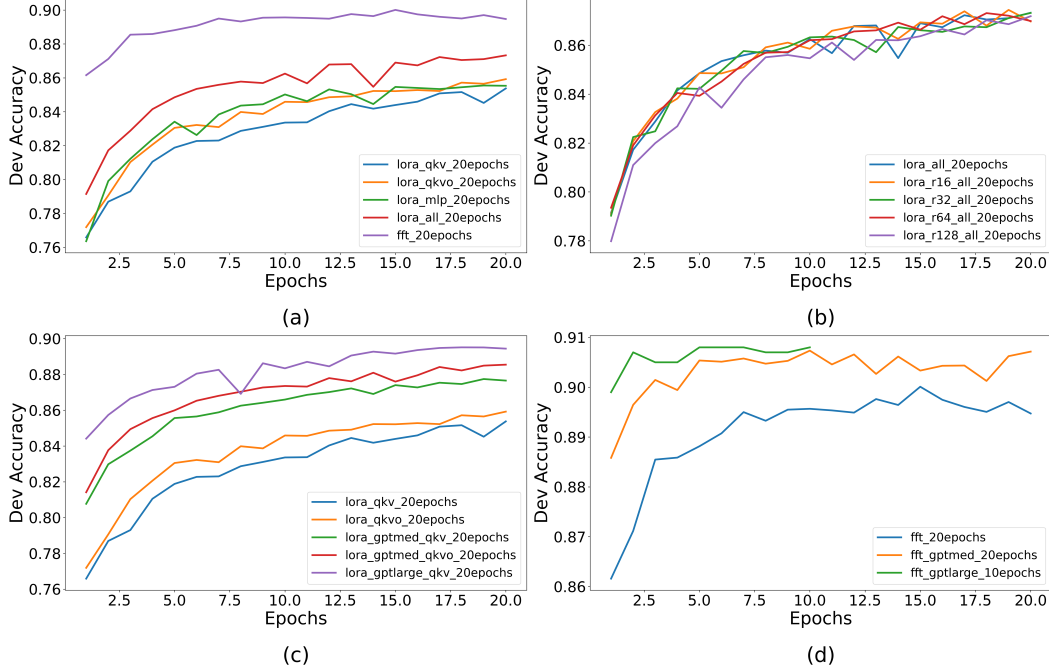


Figure 2: Paraphrase Detection training comparisons with dev dataset accuracy metric. (a) LoRA across different parameter subsets $\{ (W_q, W_k, W_v), (W_q, W_k, W_v, W_o), (W_{mlp}), (W_q, W_k, W_v, W_o, W_{mlp}) \}$ and full SFT for GPT-2 base rank=8, (b) LoRA on both attention and MLP weights for GPT-2 base across different ranks, (c) LoRA on attention weights with different GPT-2 sizes (base, medium, large) with rank=8, (d) full SFT across different GPT-2 sizes.

5.4.2 Sonnet Generation

Fig. 3 shows the performance comparison of all the fine-tuning methods, and Table 2 (left) shows the best chrF scores on the Dev set from each model, while using top-p nucleus sampling. Training loss converges smoothly across all models, and they achieve very similar final Dev set chrF scores after 200 epochs. Pre-fine-tuning on the internet poetry dataset for 50 epochs (to prevent overfitting on this data) provides marginal benefit over direct SFT using the GPT-2 (base) model. Training with extra control tokens is unstable at the beginning, as expected, because the model needs to understand those new tokens as single instruction entities and learn to generate them as they appear in the dataset. However, contrary to our hypothesis, it does not provide any performance gains here. Lastly, GPT-2 large achieves the best overall Dev set metrics, demonstrating that large high-capacity models perform such creative tasks better.

Table 2 (right) compares output sampling strategies for GPT-2 (large). Top-p sampling works best, preserving the most likely output tokens while maintaining sufficient stochasticity with a threshold of 0.9 and a temperature of 1.2, but this evaluation is based on the chosen hyperparameters. A hyperparameter search across some range of values could provide a more holistic evaluation. Also, metrics during evaluations vary slightly from those during training due to some inherent randomness in PyTorch and GPU operations, even after fixing the seeds.

We further attempted to improve the pretrained SFT model using DPO variants. As shown in the Fig. 4, none of the DPO variants provided a substantial improvement on the Dev set ChrF scores, indicating a limited ability to generalize to OOD prompts. With a very small margin, the DPOP+SFT (Synth) baseline performs the best in balancing training performance with ChrF stability, and achieved the highest **chrF score of 42.218** and dev chrF of 42.65 (both higher than TA baselines of 41.208 and

Method	Max chrF
Direct SFT (GPT-2)	42.78
Pre-FT + SFT (GPT-2)	42.79
Control Tokens SFT (GPT-2)	41.8
Direct SFT (GPT-2 Medium)	42.53
Direct SFT (GPT-2 Large)	42.9

Method	Dev chrF
top-p	41.63
top-k	40.96
beam search	41.27

Table 2: **(Left)** Maximum chrF scores for direct SFT for all GPT sizes, pre-fine-tuned GPT-2 (base) model on internet poetry dataset followed by SFT, and SFT with special prefix tokens with GPT-2 (base). **(Right)** Comparison among different output token sampling methods with GPT-2 (large), with a slight difference in both tables due to inherent randomness in PyTorch and GPUs.

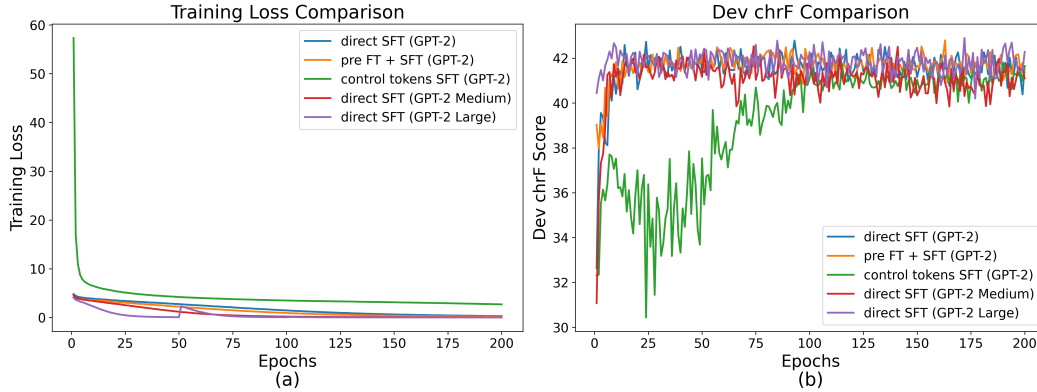


Figure 3: Comparison of different fine-tuning methods. (a) Training loss for 200 epochs for each run and (b) chrF score on the Dev dataset along with training.

41.414) for sonnet generation test among all our models, including GPT-2 (large), which performed very well on the Dev set but got a test chrF of 42.05.

6 Analysis

6.1 Paraphrase Detection

When manually evaluating the dev dataset samples on which the fine-tuned model’s predictions are wrong, we found that in some cases, when phrases are lexically similar but semantically different, the model gets confused and marks them as similar. Shared words between actual paraphrase training samples introduce bias, leading it to give "yes" outputs for samples that share some words but have different meanings. However, in general, it works very well when the difference between the two phrases is clearer. Some of the incorrect prediction examples are shown below.

1. How do I come out of a menstrual cycle stomachache?; What can I take for delayed menstrual cycle?
2. Did Baba Farid live in a jungle and crows ate his flesh?; When the god man, Sathya Sai Baba, was lying helpless in the hospital, whom did his devotees pray to?

6.2 Sonnet Generation

DPO Result Analysis. As shown in the blue lines in Fig. 4, Standard DPO led to a catastrophic drop in both training and development ChrF scores, despite quickly converging to the highest preference selection accuracy. In contrast, the DPOP and DPO(P)+SFT variants maintained stable ChrF scores throughout training. We attribute this failure of DPO to its tendency to exploit the objective by drastically reducing the absolute probability of both preferred and rejected completions to greedily maximize the margin between them. On the other hand, DPOP explicitly penalizes drops in log-probability for preferred completions, preventing the model from reducing their log-probability too much. Similarly, SFT co-training prevents model collapse by anchoring it to the ground-truth distribution through the supervision signal.

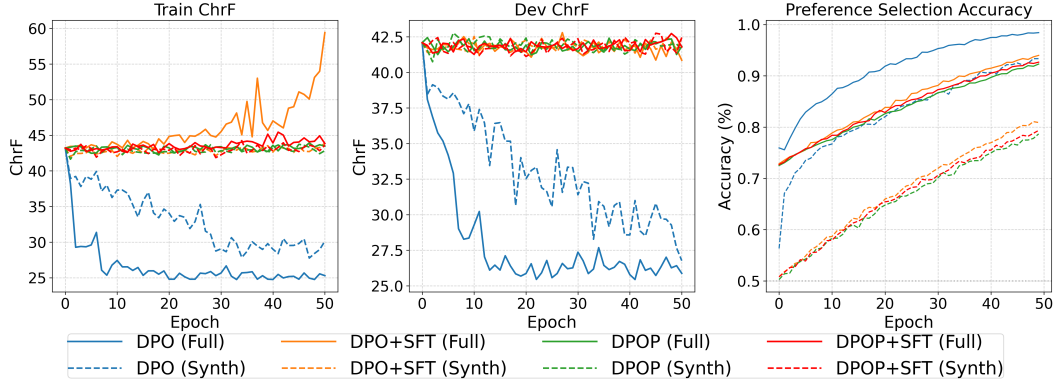


Figure 4: Quantitative results on the DPO variants.

Incorporating GT data effectively improves the quality of preferred completions, thereby providing a steep gradient for rapid preference alignment. Hence, models trained on the full dataset (including the GT preference data) generally achieve higher preference accuracy faster than those trained only on the synthetic preference data. However, the DPO (Full) variant experienced a significantly faster and more severe collapse than the DPO (Synthetic) variant. This is possibly because the high-contrast, low-diversity nature of the GT pairs provided more room for the model to hack the objective. Interestingly, training the DPO with an auxiliary SFT loss allowed the model to overfit to the small training dataset, achieving a much higher training ChrF than the pre-finetuning + SFT model, without sacrificing generalization too much on the development set.

Beam Search Sampling: Frequent failure modes included confidently outputting successive line breaks and repetitive phrasing. Additional structural constraints, such as 3-gram blocking and prohibiting line breaks with less than 3 words, did not improve the beam search performance.

Dictionary-aided Search: ChrF scores were 0.31-0.32, showing clear diminished performance. There exists a structural mismatch between token-level and word-level representations for words with prefixes and suffixes when encoding and decoding, causing incomplete logit masking.

7 Conclusion

In this project, we explored data-efficient strategies for fine-tuning and aligning GPT-2 models for paraphrase detection and sonnet generation tasks. For paraphrase detection, we compared full supervised fine-tuning (SFT) with parameter-efficient methods such as LoRA across different GPT-2 model sizes, parameter subsets, and LoRA ranks. While full SFT achieved the highest raw accuracy, LoRA provided competitive performance while updating less than 1% of the model parameters. In particular, LoRA applied to both attention and MLP layers on GPT-2 Large with $rank = 8$ demonstrated smoother learning behavior than full FT and achieved a test accuracy of 0.862, outperforming the provided baselines while maintaining smoother training behavior. For sonnet generation, we evaluated several approaches to improve performance under limited data. Pre-fine-tuning on a larger poetry dataset provided a small improvement over direct SFT, while control tokens designed to enforce rhyme structure did not lead to measurable gains. Among decoding strategies, top-p nucleus sampling produced the best results. Preference optimization using DPO variants improved training stability but did not significantly outperform the SFT baseline. Overall, our results highlight the effectiveness of parameter-efficient adaptation and the challenges of aligning models for structured creative generation tasks.

Team contributions

All team members actively contributed to the project discussions, design, experimentation, and write-ups. In particular, Aryaman led the LoRA & full SFT comparisons for paraphrase detection, and pre-fine-tuning, full SFT, and prefix-token SFT comparisons for sonnet generation. Joseph implemented the base GPT-2 architecture, performed the sentiment analysis task, and conducted dictionary-aided constrained decoding for sonnet generation. Zeyuan developed and evaluated all the DPO variants and tested different output-sampling strategies for sonnet generation.

References

- [1] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.
- [2] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- [3] Armen Aghajanyan, Sonal Gupta, and Luke Zettlemoyer. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In *Proceedings of the 59th annual meeting of the association for computational linguistics and the 11th international joint conference on natural language processing (volume 1: long papers)*, pages 7319–7328, 2021.
- [4] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [6] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.
- [7] Jordan Clive, Kris Cao, and Marek Rei. Control prefixes for parameter-efficient text generation. In *Proceedings of the Second Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, pages 363–382, 2022.
- [8] Jiayu Qin, Jianchao Tan, Kefeng Zhang, Xunliang Cai, and Wei Wang. Maskprune: Mask-based llm pruning for layer-wise uniform structures. *arXiv preprint arXiv:2502.14008*, 2025.
- [9] Kaggle poems dataset. <https://www.kaggle.com/datasets/michaelarman/poemsdataset>.
- [10] Alizée Pace, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. West-of-n: Synthetic preference generation for improved reward modeling. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*, 2024.
- [11] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.
- [12] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, 2018.
- [13] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.
- [14] Carnegie Mellon University. The cmu pronouncing dictionary, 2014. Version 0.7b.

A Appendix

A.1 Paraphrase Detection

The following figures present fine-tuning performance comparisons across different axes, using loss and accuracy metrics on both the training and validation datasets. One notable result in Fig. 8 is that full SFT quickly overfits the training dataset, and the validation loss starts to diverge, though we only save checkpoints when the Dev dataset performance improves. Thus, for our evaluation, we select the LoRA model on GPT-2 Large with rank=8 due to its smoother behavior overall.

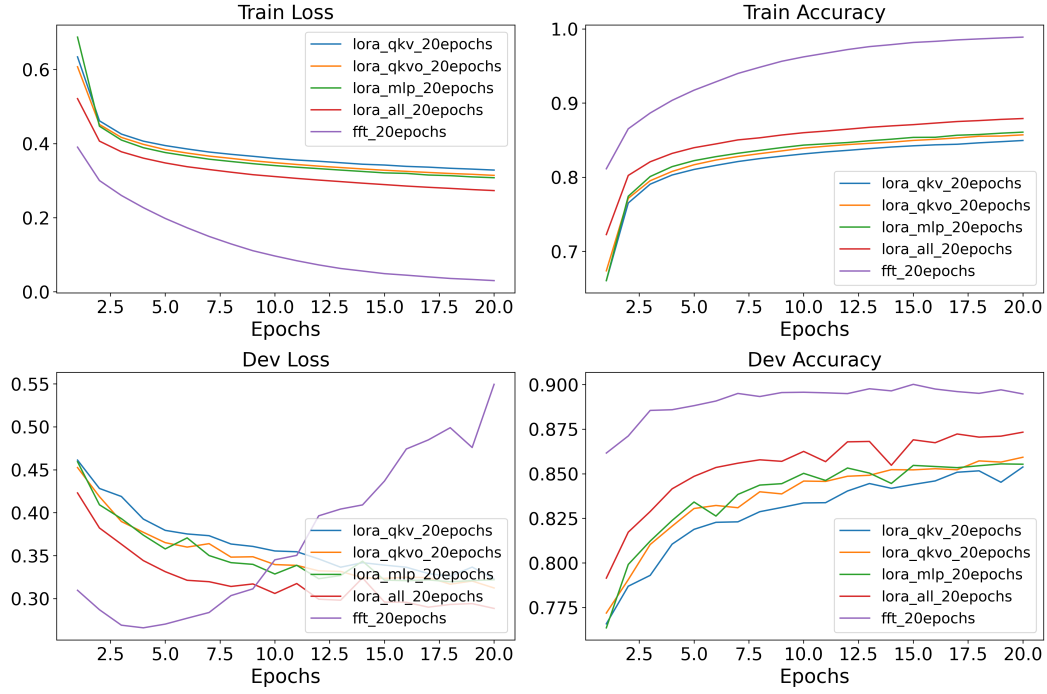


Figure 5: LoRA comparisons across different parameter adapters and full SFT.

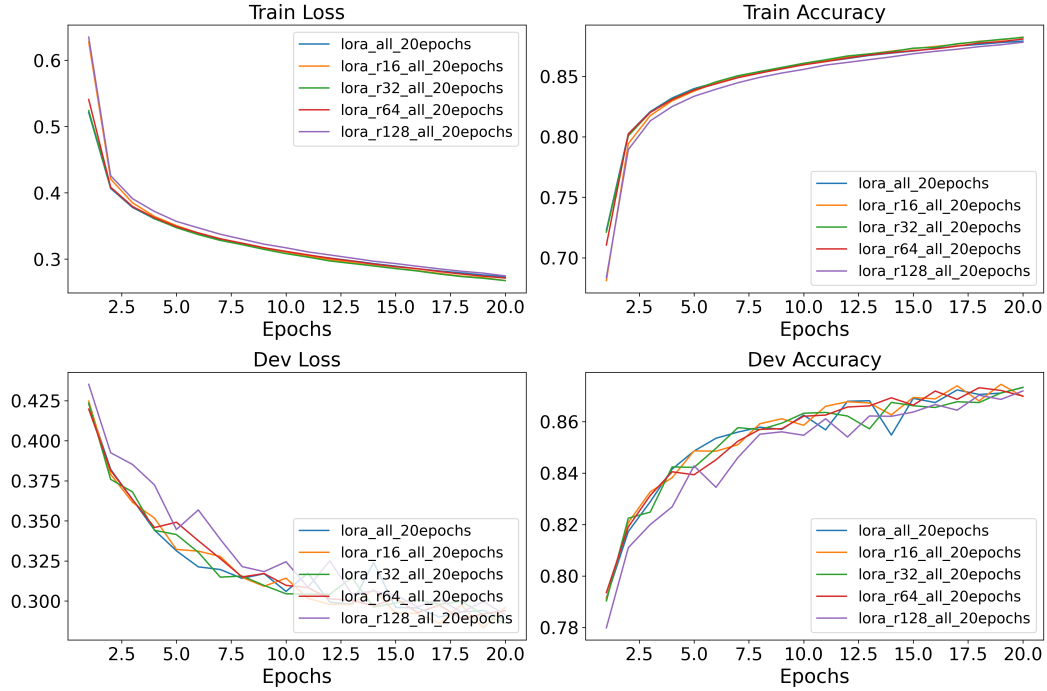


Figure 6: LoRA comparisons across different LoRA ranks for rank=8 and GPT-2 base.

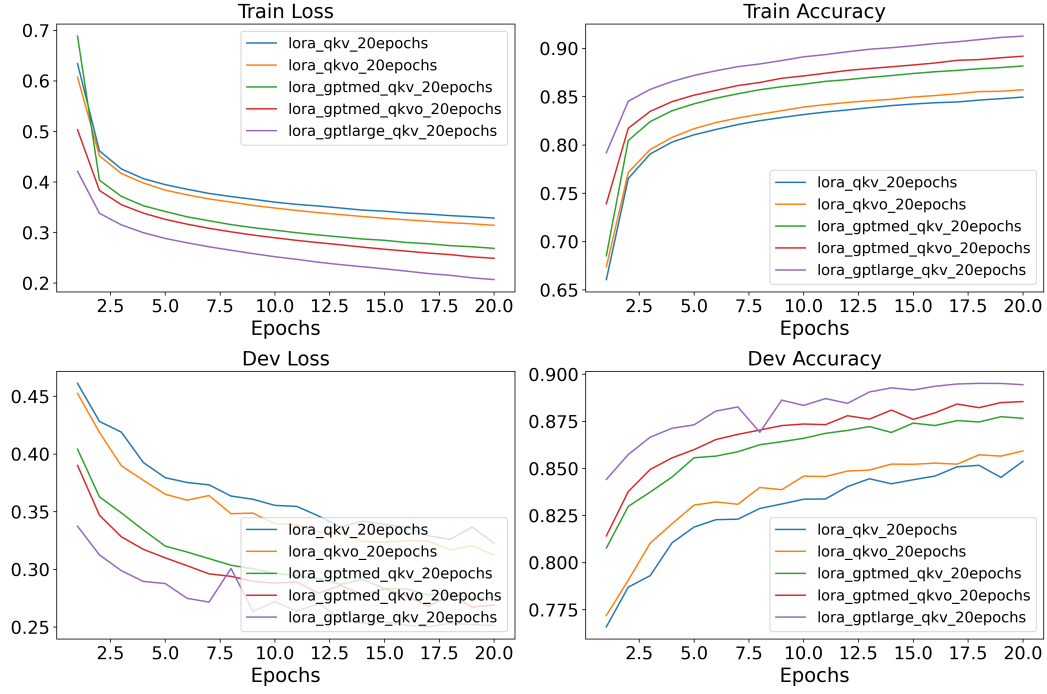


Figure 7: LoRA Comparisons for different GPT-2 sizes for rank=8 and adapters applied to (W_q , W_k , W_v , W_o).

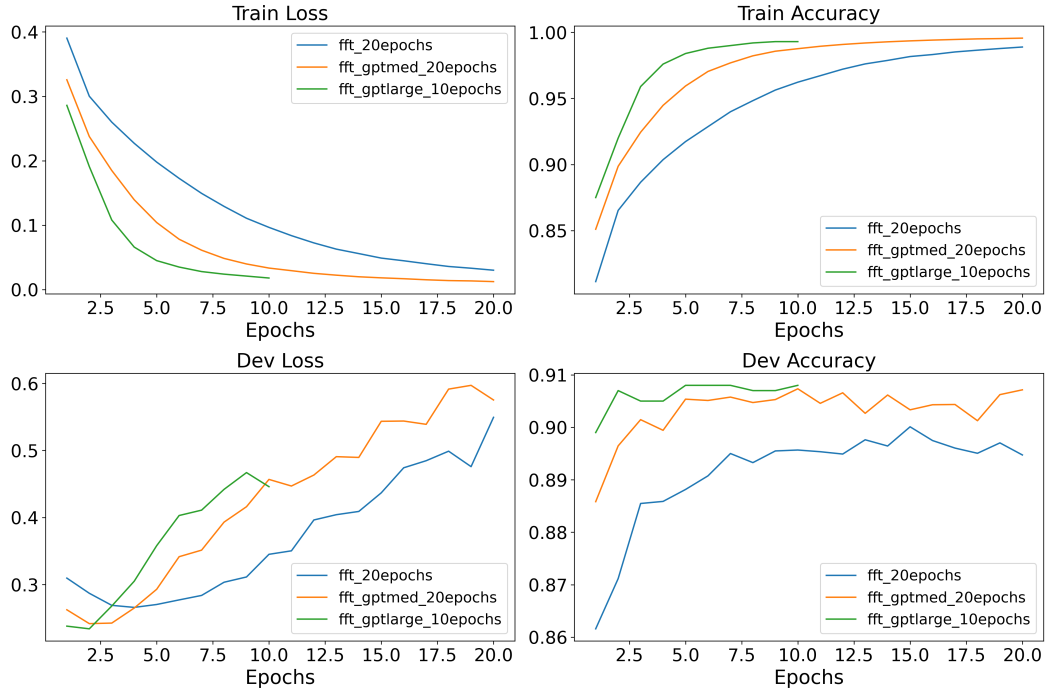


Figure 8: full SFT comparisons across different GPT-2 sizes.