

Build GPT-2

Stanford CS224N Default Project

Yixiao Zhang
yixiaoz@stanford.edu

Shirley Yu
weiranyu@stanford.edu

Pengyu Mo
pengyumo@stanford.edu

Abstract

We evaluate a suite of advanced adaptation strategies on GPT-2 to balance performance and efficiency for paraphrase detection and sonnet generation. We implement and compare Parameter-Efficient Fine-Tuning (PEFT) methods, including Low-Rank Adaptation (LoRA) and Representation Fine-Tuning (ReFT), integrated with SMART regularization for manifold stability. To further refine generative quality, we employ Direct Preference Optimization (DPO) and Group Relative Policy Optimization (GRPO) to enforce complex constraints like iambic pentameter. Our results demonstrate that these methods bridge the gap between restricted last-layer tuning and costly full fine-tuning, optimizing the trade-off between task performance, robustness, and computational throughput.

1 Key Information to include

Mentor: Fang Wu

External Collaborators/Project Sharing: No

Late Days Applied: 3 (Yixiao (3) + Shirley (1) + Pengyu (5) = 9 / 3 = 3)

2 Introduction

Modern Large Language Models (LLMs) can handle tasks ranging from sentiment analysis to creative writing, yet adapting them effectively to specialized downstream tasks remains challenging. The main difficulty lies in balancing task performance with computational efficiency. Simple approaches such as tuning only the final classification layer often fail to capture deeper semantic shifts, while full fine-tuning of all parameters is computationally expensive and may introduce instability.

In this project, we build GPT-2 from core Transformer components and explore several strategies for adapting pretrained models to downstream tasks. We investigate parameter-efficient fine-tuning (PEFT) methods, adversarial regularization techniques, and preference-based optimization methods across two representative tasks: paraphrase detection and sonnet generation.

To improve efficiency, we evaluate two PEFT methods: **LoRA (Low-Rank Adaptation)**, which injects trainable low-rank matrices into attention projections, and **ReFT (Representation Finetuning)**, which modifies hidden representations through learned interventions. To stabilize training and improve robustness, we incorporate **SMART regularization**, which encourages smooth model behavior under adversarial perturbations of hidden states.

For generative alignment, we further explore preference-based optimization methods. We implement **Direct Preference Optimization (DPO)**, which directly optimizes the policy using preference pairs, and **Group Relative Policy Optimization (GRPO)**, which samples groups of outputs and updates the model using relative reward advantages. GRPO aims to encourage adherence to stylistic and structural constraints such as rhyme patterns and meter.

By evaluating these approaches on both **paraphrase detection** and **sonnet generation**, we analyze how different adaptation strategies balance efficiency, robustness, and generation quality.

3 Related Work

Our work builds upon the Transformer architecture [1], specifically the decoder-only GPT-2 model [2], and utilizes the AdamW optimizer [3, 4] to decouple weight decay for improved generalization. To address the computational costs and instability of full-parameter fine-tuning, we explore several robust strategies. SMART [5] incorporates smoothness-inducing regularization to prevent the loss of pretrained knowledge, while LoRA [6] and ReFT [7] offer efficient adaptation by injecting trainable low-rank matrices or applying learned interventions to hidden representations, respectively.

To further align generative outputs with stylistic and structural constraints, we employ preference-based optimization. Direct Preference Optimization (DPO) [8] refines model alignment by treating the language model as a reward model, while Group Relative Policy Optimization (GRPO) [9] optimizes policies via group-relative rewards without a separate critic network. Performance is evaluated using standard accuracy and chrF [10], which provides granular character n-gram analysis for creative generation.

4 Approach

We implement GPT-2 from core Transformer components [1] and adapt it to classification and generation tasks following the default project framework. Our study focuses on how different fine-tuning strategies affect downstream performance.

Baseline Model and Optimization. Our core architecture is a decoder-only Transformer following the GPT-2 design [2], which integrates learned token and positional embeddings, masked multi-head self-attention, and position-wise feed-forward networks. We initialize this architecture with pretrained GPT-2 weights from the HuggingFace library, serving as the foundational baseline provided by the course project skeleton.

We evaluate model performance across two distinct fine-tuning regimes: a *Last Linear Layer* approach, where GPT-2 parameters remain frozen, and a *Full Model* approach, where all parameters are updated. For optimization, we utilize AdamW, which enhances the standard Adam algorithm [3] by decoupling weight decay from the gradient update [4] to ensure more effective regularization.

Paraphrase Detection. We reformulate paraphrase detection as a cloze-style next-token prediction task: Is " s_1 " a paraphrase of " s_2 "? Answer "yes" or "no". The model predicts the next token conditioned on the full prompt. We use the language modeling (LM) head to score only the vocabulary logits corresponding to the token IDs for "yes" and "no" (two-token scoring). The higher-scoring token determines the predicted label.

Sonnet Generation. For sonnet modeling, we train GPT-2 using the standard autoregressive objective, $\mathcal{L} = \sum_{t=1}^n \log P(x_t | x_{<t})$, where targets are shifted by one token. All hidden states are projected through the tied output embedding matrix to obtain vocabulary logits. At inference time, we condition on the first three lines of a held-out sonnet and generate tokens sequentially using temperature scaling and top- p sampling. Generation continues until an end-of-sequence token or a maximum length is reached, producing a complete sonnet.

Low-Rank Adaptation (LoRA). To enable parameter-efficient fine-tuning, we implement a custom LoRALinear wrapper following the methodology of Hu et al. [6]. We apply this adaptation specifically to the query (W_q) and value (W_v) projection matrices within each self-attention block, as these have been shown to capture the most significant task-specific information. For a given linear layer with frozen weights $W_0 \in \mathbb{R}^{d \times k}$, we introduce a trainable bypass consisting of two low-rank matrices, $A \in \mathbb{R}^{r \times k}$ and $B \in \mathbb{R}^{d \times r}$. To ensure training stability, we initialize A using Kaiming uniform initialization and B to zero, such that the initial adapter contribution $\Delta W = BA$ is zero.

In our implementation, we optimize computational efficiency by avoiding the materialization of the full weight update matrix. Instead, the forward pass is computed via two sequential low-rank projections: $h = xW_0^\top + (\text{Dropout}(x)A^\top B^\top) \cdot \frac{\alpha}{r}$, where α is a constant hyperparameter used to scale the adapter output. This scaling factor allows us to maintain a consistent learning signal across different choices of rank r .

Representation Finetuning (ReFT). To explore intervention-based adaptation, we implement ReFT following Wu et al. [7], specifically the Low-rank Linear Subspace ReFT (LoReFT) variant. Unlike

weight-based methods, ReFT modifies model behavior by manipulating hidden representations $h \in \mathbb{R}^d$ during the forward pass. The intervention is defined by the following operation: $h' = h + R^\top(Wh + b - Rh)$, where $R \in \mathbb{R}^{r \times d}$ is an orthogonal projection matrix, $W \in \mathbb{R}^{r \times d}$ is a learnable weight matrix, and $b \in \mathbb{R}^r$ is a learnable bias.

In our implementation, we wrap the standard Transformer layers in a custom module that intercepts the layer output and applies the LoReFT intervention. We initialize R using orthogonal initialization and W with Kaiming uniform initialization to ensure stable gradients at the start of training.

SMART Regularization. To improve robustness during fine-tuning, we implement Smoothness-Inducing Adversarial Regularization for Text (SMART) [5]. SMART encourages the model to produce stable predictions under small adversarial perturbations of the hidden representations. Given a hidden representation h and predicted distribution $p_\theta(y | h)$, SMART introduces a consistency regularization term:

$$\mathcal{L}_{SMART} = \text{KL}(p_\theta(\cdot | h) \| p_\theta(\cdot | h + r^*)),$$

where r^* is a perturbation constrained by $\|r\| \leq \epsilon$ and approximated via gradient ascent on the KL divergence. The final training objective is

$$\mathcal{L} = \mathcal{L}_{CE} + \lambda_{SMART} \mathcal{L}_{SMART},$$

where λ_{SMART} controls the strength of regularization.

In our implementation, we expose intermediate hidden representations during the forward pass and apply SMART within the training loop. For paraphrase detection, we perturb the final hidden state corresponding to the last token before computing the classification logits. For sonnet generation, we perturb the sequence of hidden states before the language modeling head and compute the KL term over valid next-token prediction positions. This adds an adversarial inner optimization step during training but leaves inference unchanged.

LoRA + SMART. We also explored combining parameter-efficient adaptation with adversarial regularization by applying SMART regularization on top of LoRA fine-tuning. In this setup, LoRA adapters are inserted into the attention projection layers while SMART perturbations are applied to the hidden representations during training, allowing the model to benefit from both efficient parameter updates and improved robustness.

Direct Preference Optimization (DPO). To incorporate preference-based alignment without training a separate reward model, we implement Direct Preference Optimization (DPO) following Rafailov et al. [11]. DPO trains the policy directly using preference triplets (x, y_w, y_l) , where x is a prompt, y_w is a preferred continuation, and y_l is a rejected continuation.

For each prompt–response pair, we compute the conditional log-probabilities of the response tokens under the policy model π_θ and a frozen reference model π_{ref} . The response likelihood is computed using the standard autoregressive formulation:

$$\log \pi_\theta(y|x) = \frac{1}{|y|} \sum_t \log p_\theta(y_t | x, y_{<t}).$$

The DPO objective is

$$L_{DPO} = -\log \sigma(\beta[(\log \pi_\theta(y_w|x) - \log \pi_\theta(y_l|x)) - (\log \pi_{\text{ref}}(y_w|x) - \log \pi_{\text{ref}}(y_l|x))]),$$

where β controls the strength of preference optimization set to 0.01.

To construct preference data, we adopt an iterative self-improvement strategy. At each stage, the current model generates candidate continuations for the prompt, which are treated as negative samples. These generated outputs are paired with the gold ground-truth continuations to form (x, y_w, y_l) preference triplets. The resulting dataset is then used to fine-tune the next model iteration with DPO, and this process is repeated for multiple rounds to progressively improve generation quality.

GRPO Post-Training for Sonnet Generation. We further explore reward-based optimization using Group Relative Policy Optimization (GRPO) as a post-training stage [9]. We first train a GPT-2 sonnet model using supervised language modeling (optionally with SMART regularization) to learn basic sonnet structure and vocabulary. GRPO is then applied to refine generation behavior through reward optimization.

For each training sonnet, we use the first k lines ($k = 3$) as a prompt and treat the remaining lines as the reference continuation. Given a prompt x , the model samples a group of G candidate continuations $y_1, \dots, y_G \sim \pi_\theta(\cdot|x)$. Each continuation receives a reward primarily based on chrF similarity to the reference continuation, together with lightweight heuristics encouraging proper sonnet form, reasonable line length, rhyme patterns, stylistic markers, and reduced repetition. Rewards are normalized within each group to produce group-relative advantages

$$A_i = \frac{r_i - \mu_r}{\sigma_r + \epsilon},$$

where μ_r and σ_r are the mean and standard deviation of rewards within the group.

Model parameters are updated using a clipped policy objective similar to PPO:

$$\mathcal{L}_{GRPO} = -\frac{1}{G} \sum_{i=1}^G \min(\rho_i A_i, \text{clip}(\rho_i, 1 - \epsilon, 1 + \epsilon) A_i),$$

where ρ_i is the likelihood ratio between the current policy and a previous policy snapshot. To stabilize training, we include a reference-model regularization term that penalizes large deviations from the supervised model. This post-training stage improves generation quality while preserving the linguistic structure learned during supervised training.

5 Experiments

5.1 Data

We utilize four datasets as provided in the project codebase:

Stanford Sentiment Treebank (SST). The Stanford Sentiment Treebank [12] is a dataset consisting of movie review sentences annotated with sentiment labels. The input to the model is a tokenized sentence, and the output is a binary sentiment label as either *positive* or *negative*.

CFIMDB. CFIMDB is a movie review sentiment dataset derived from the IMDB corpus. Similar to SST, the task is formulated as binary classification, where the model takes reviews as input and predicts the sentiment polarity.

Quora Question Pairs. The Quora Question Pairs dataset¹ contains pairs of questions from Quora along with a label indicating whether the two questions are paraphrases of each other. We reformulate the task as a cloze-style next-token prediction problem, where the model is prompted to predict a label following a question pair. The final prediction is determined by comparing the language modeling logits corresponding to the tokens *yes* and *no*.

Shakespearean Sonnets Corpus. For language modeling and generation, we use a corpus of Shakespearean sonnets. Each example consists of lines from a sonnet, and the task is to model the probability of the next token given the preceding context.

5.2 Evaluation method

We evaluate model performance using task-appropriate metrics for classification and generation tasks.

Classification Tasks. For sentiment classification and paraphrase detection, we report **accuracy** on the development set as the primary evaluation metric.

Language Modeling and Generation. For sonnet generation, we report **chrF** [10] as the primary evaluation metric. chrF measures character n-gram overlaps between the generated sonnet and the reference text, providing a granular assessment of structural and stylistic alignment.

As a secondary metric, we report **perplexity (PPL)**, which measures how well the language model predicts the ground-truth sequence. It is defined as: $\text{PPL} = \exp\left(-\frac{1}{N} \sum_{t=1}^N \log P(x_t|x_{<t})\right)$ where N is the total number of tokens. While automatic metrics provide quantitative comparison, qualitative inspection of generated sonnets is also used to assess coherence and stylistic quality.

¹<https://www.quora.com/q/quoradata/First-Quora-Dataset-Release-Question-Pairs>

5.3 Experimental details

Baselines. Using the default hyperparameters, our Last Linear Layer fine-tuning achieved dev accuracies comparable to the baseline on both **SST (0.461)** and **CFIMDB (0.861)**. We then explored learning rates $\{1e-3, 1e-4, 1e-5\}$, batch sizes $\{8, 16, 32, 64\}$, and epochs $\{3, 5, 10, 20, 30\}$ to get a fine-tuned baseline for later comparison with our extensions.

LoRA (Paraphrase Detection and Sonnet Generation). LoRA introduces several specific hyperparameters, most notably the rank r and the scaling factor α . We perform a grid search over these parameters and the learning rate to identify the optimal configuration for adaptation. Specifically, we test $r \in \{4, 8, 16, 32\}$, $\alpha \in \{16, 32, 64\}$, and $\eta \in \{1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$.

Experiments were executed across two environments: a local MacBook Pro (M4 Pro chip) and Google Cloud Platform (GCP) with a single NVIDIA L4 GPU. Models are trained up to 30 epochs for paraphrase detection and 20 epochs for sonnet generation.

ReFT (Paraphrase Detection and Sonnet Generation). For ReFT, we conduct a hyperparameter sweep to evaluate the impact of the intervention rank r_{reft} and the learning rate η . Specifically, we test $r_{reft} \in \{4, 8, 16, 32\}$ and $\eta \in \{1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}\}$.

Experiments were executed across the same environments as LoRA(local and GCP). Models are trained up to 30 epochs for paraphrase detection and 20 epochs for sonnet generation.

SMART (Paraphrase Detection and Sonnet Generation). We evaluate the effect of SMART regularization on both the paraphrase detection and sonnet generation tasks using the Quora Question Pairs dataset and the Shakespeare Sonnets dataset, respectively. All experiments use the GPT-2 model described in Approach and are fine-tuned using the AdamW optimizer with a learning rate of 1×10^{-5} . For paraphrase detection we use a batch size of 16, while for sonnet generation we use a smaller batch size of 2 due to the longer sequence lengths.

SMART introduces two key hyperparameters: the perturbation radius ϵ and the regularization weight λ_{SMART} . We perform a small hyperparameter sweep to evaluate their impact on performance, testing $\lambda_{SMART} \in \{0.1, 0.3, 0.5\}$ and $\epsilon \in \{0.01, 0.05, 0.1\}$.

Models for paraphrase detection are trained for up to 30 epochs, with the best development performance typically observed around epoch 15. For sonnet generation, models are trained for 10 epochs and the best checkpoint is selected based on development perplexity.

LoRA + SMART (Paraphrase Detection). To explore the combined effect of parameter-efficient adaptation and adversarial regularization, we apply SMART on top of LoRA fine-tuning. LoRA adapters are inserted into the GPT-2 attention layers with rank $r = 32$ and scaling factor $\alpha = 64$. The model is trained for 20 epochs using the AdamW optimizer with a learning rate of 1×10^{-4} and batch size 16. For SMART regularization, we use perturbation radius $\epsilon = 0.05$, regularization weight $\lambda_{SMART} = 0.3$, adversarial step size 0.01, and one inner optimization step. All experiments are conducted using the base GPT-2 model on the Quora Question Pairs dataset.

DPO (Sonnet Generation). DPO is applied after supervised training, using the previously trained sonnet model as initialization. At each iteration, the current model generates candidate continuations for the prompt, which are treated as negative samples and paired with the gold ground-truth continuation to form preference triplets for the next round of training.

During generation we use temperature = 1.2, top- p filtering with $p = 0.7$ and $\beta = 0.01$. Training uses a batch size of 8. To stabilize iterative training, the learning rate is gradually reduced across iterations following $\{1 \times 10^{-5}, 5 \times 10^{-6}, 1 \times 10^{-6}, 5 \times 10^{-7}, 1 \times 10^{-7}\}$.

For each iteration, the checkpoint with the best development chrF score is selected. After training, we generate sonnets with 10 different random seeds and report the best chrF score.

GRPO (Sonnet Generation). We evaluate Group Relative Policy Optimization (GRPO) as a post-training method on the sonnet generation task. The GRPO stage is initialized from the best SMART-trained checkpoint, which achieves a chrF score of 42.63 on the development set. We use a group size of 4, learning rate 5×10^{-7} , and KL regularization weight $\beta_{KL} = 0.2$ to constrain the policy from drifting too far from the reference model. To reduce repetitive generations during sampling, we additionally apply a no-repeat trigram constraint.

5.4 Results

For paraphrase detection, we reached **0.910** accuracy on dev set and **0.894** accuracy on test set. For sonnet generation, we reached **44.201** chrF on dev set and **43.176** on test set. Below are the detailed results for our experimented extensions.

Baselines. The fine-tuning results for SST and CFIMDB validate the integrity of our core GPT-2 implementation. In the last linear layer configuration, we achieve accuracies of **0.473 (SST)** and **0.878 (CFIMDB)**, outperforming the TA baselines of 0.462 and 0.861, respectively. With full model fine-tuning, performance improves to **0.516 (SST)** and **0.984 (CFIMDB)**, likewise exceeding the 0.513 and 0.976 baseline marks. For our target tasks, the model achieves an accuracy of **0.896** on paraphrase detection and a chrF score of **41.492** on sonnet generation. These figures serve as the foundational baselines for our subsequent extensions. Detailed hyperparameter configurations are provided in Appendix 3.

LoRA. We observed that LoRA significantly reduces training and inference latency; specifically, the training throughput for paraphrase task increased from **8 it/s to 13.8 it/s** on GCP and for sonnet task increased from **3 it/s to 4.8 it/s** on the MacBook M4 Pro. Our experiments indicate that the best performance is achieved with $r = 32$, $\alpha = 64$, and a learning rate of 1×10^{-4} . For paraphrase detection, LoRA achieved an accuracy of **0.893**, exceeding the TA baseline (0.882) and approaching our full fine-tuning baseline (0.896). In sonnet generation, it achieved a chrF score of **42.111**, outperforming both the TA baseline (41.414) and our full fine-tuning baseline (41.492).

ReFT. Computational efficiency gains were comparable to LoRA, with paraphrase training throughput reaching **14.5 it/s** on GCP and sonnet training throughput reaching **4.7 it/s** on the MacBook M4 Pro. In terms of predictive performance, the most effective results are yielded by a configuration of $r = 16$ and a learning rate of 1×10^{-4} . For paraphrase detection, ReFT achieved an accuracy of **0.888**, exceeding the TA baseline (0.882). For sonnet generation, it produced a chrF score of **41.514**, slightly exceeding the TA baseline (41.414) and our full fine-tuning baseline (41.492).

SMART (Paraphrase Detection and Sonnet Generation). Table 1 and Table 4 (see in Appendix) summarize the results of our SMART experiments on the paraphrase detection and sonnet generation task. We compare different combinations of SMART hyperparameters and report the best development accuracy achieved during training.

λ_{SMART}	ϵ	Batch Size	LR	Dev Accuracy	Best Epoch	Model Size
0.1	0.05	16	$1e^{-5}$	0.90015	15	GPT-2
0.3	0.05	16	$1e^{-5}$	0.90076	15	GPT-2
0.5	0.05	16	$1e^{-5}$	0.89997	15	GPT-2
0.3	0.01	16	$1e^{-5}$	0.90069	7	GPT-2
0.3	0.10	16	$1e^{-5}$	0.90076	14	GPT-2
0.3	0.05	16	$1e^{-5}$	0.90828	7	GPT-2 Medium
0.3	0.05	16	$1e^{-5}$	0.90952	15	GPT-2 Large

Table 1: Development accuracy of SMART regularization under different hyperparameter settings.

DPO (Sonnet Generation). Dev-set chrF across DPO iterations with decreasing learning rates for both the GPT2 and GPT2 large model, other hyperparameters fixed. Performance peaks at this iteration, and further iterations lead to a decline in chrF.

Iter	LR	Temp	Top- p	β	Batch	Model	Best chrF
1	$1e^{-5}$	1.2	0.7	0.01	8	GPT-2	41.89
2	$5e^{-6}$	1.2	0.7	0.01	8	GPT-2	42.73
3	$1e^{-6}$	1.2	0.7	0.01	8	GPT-2	43.68
4	$5e^{-7}$	1.2	0.7	0.01	8	GPT-2	43.76
5	$1e^{-7}$	1.2	0.7	0.01	9	GPT-2	44.20
1	$1e^{-5}$	1.2	0.7	0.01	8	GPT-2-large	43.45
2	$5e^{-6}$	1.2	0.7	0.01	8	GPT-2-large	44.02

Table 2: DPO post-training results for sonnet generation across iterative preference optimization stages.

GRPO (Sonnet Generation). We evaluate GRPO as a post-training method initialized from the best SMART checkpoint, which achieves a chrF score of 42.63 on the development set. After one epoch of GRPO training, the chrF score decreases slightly to 42.38 with a development loss of 4.170 and perplexity of 64.73. Continuing training for a second epoch further reduces the chrF score to **42.00**.

Overall, GRPO does not improve performance over the SMART baseline in this setting and instead leads to a small degradation in generation quality.

LoRA + SMART (Paraphrase Detection). We also evaluate the combination of LoRA and SMART to integrate parameter-efficient adaptation with adversarial regularization. Using LoRA with $r = 32$ and $\alpha = 64$ together with SMART regularization ($\lambda_{SMART} = 0.3$, $\epsilon = 0.05$), the model achieves a development accuracy of **0.891** with a reduced running time.

6 Analysis

To provide a cohesive framework for our evaluation, we analyze how each method navigates the trade-offs between parameter efficiency, stability, and stylistic alignment.

LoRA (Paraphrase Detection and Sonnet Generation). Overall, LoRA proves to be a highly effective parameter-efficient strategy, achieving results comparable to full fine-tuning while training only approximately 1% of total parameters. The learning rate is a critical factor for convergence. We find that $\eta = 1 \times 10^{-5}$ results in excessively slow convergence, while $\eta = 1 \times 10^{-3}$ proves too aggressive, causing the loss to oscillate. The optimal learning rate of 1×10^{-4} provides the necessary gradient signal for the adapters to learn without destabilizing the frozen backbone.

Regarding model capacity, we observe that the rank r and scaling factor α must be sufficiently large to capture the task’s complexity. Lower ranks provide insufficient degrees of freedom, leading to a noticeable performance gap.

ReFT (Paraphrase Detection and Sonnet Generation). ReFT demonstrates strong efficiency, matching or exceeding the TA baseline while training only approximately 0.25% of the model parameters. Similar to LoRA, ReFT performs best with a moderate learning rate (1×10^{-4}), while smaller values slow convergence and larger values lead to training instability.

Regarding model capacity, ReFT shows sensitivity to the intervention rank r_{reft} . We find that $r = 16$ yields the best performance, while increasing the rank further to 32 does not produce proportional gains and occasionally leads to slight degradation. These results indicate that although ReFT is slightly less performant than LoRA, its efficiency and high throughput make it a strong alternative in resource-constrained settings.

SMART (Paraphrase Detection and Sonnet Generation). Tables 1 and 4 summarize the effect of SMART regularization on both tasks. Overall, SMART provides small but consistent improvements over the baseline models, indicating that adversarial smoothness regularization can improve model robustness and generation quality.

Across both tasks, moderate regularization performs best. For paraphrase detection, $\lambda_{SMART} = 0.3$ achieves the highest development accuracy, while larger values slightly reduce performance, suggesting that overly strong regularization can interfere with task learning. Similarly, for sonnet generation, the best chrF score (42.63) is obtained with a small regularization weight ($\lambda_{SMART} = 0.01$), and larger values slightly degrade generation quality.

The perturbation radius ϵ has relatively minor impact within the tested range for both tasks, indicating that the model is stable to small variations in perturbation magnitude. For paraphrase detection, increasing the model size produces the largest gains, with development accuracy improving from 0.90076 (GPT-2) to 0.90952 (GPT-2 Large). Overall, these results suggest that SMART yields modest but consistent improvements, particularly when combined with larger models or applied to stabilize generation training.

DPO (Sonnet Generation). We observe that large learning rates or large β values often lead to performance degradation after several iterations. This behavior aligns with the intuition from the DPO formulation [11], where β controls how strongly the policy deviates from the reference model. When β or the learning rate is too large, the policy aggressively increases the likelihood gap between preferred and rejected responses, causing the model to drift from the stable language distribution learned during supervised training.

We also observe diminishing returns as the number of DPO iterations increases. While early iterations typically improve chrF scores, too many rounds gradually reduce performance, likely due to accumulated model-generated negatives introducing noise into the preference data.

Finally, larger GPT-2 variants improve chrF more quickly in early iterations, but the final scores remain similar across model sizes, suggesting that preference optimization mainly refines stylistic generation rather than expanding model capacity.

GRPO (Sonnet Generation). GRPO does not improve over the SMART baseline and instead causes a small decrease in chrF. This likely occurs because the reward used in GRPO is only partially aligned with the final evaluation metric: while it includes chrF, it also relies on handcrafted style and form heuristics that may favor generic Shakespeare-like text rather than reference-faithful continuations. Moreover, sonnet generation is a small-data, high-variance setting, so reinforcement learning updates can easily move the model away from the strong supervised solution already obtained with SMART.

LoRA + SMART (Paraphrase Detection). The combination of LoRA and SMART did not outperform our full fine-tuning baseline or the SMART-only approach. This result is expected as LoRA uses significantly fewer parameters, the model’s capacity to represent complex patterns is constrained; consequently, adding SMART’s manifold regularization provides limited extra benefit.

7 Conclusion

In this project, we investigated how different adaptation strategies affect the performance, robustness, and efficiency of GPT-2 across two tasks: paraphrase detection and sonnet generation. Starting from a baseline implementation of GPT-2, we explored several fine-tuning approaches including parameter-efficient methods (LoRA and ReFT), adversarial regularization (SMART), and preference-based optimization methods (DPO and GRPO).

Our results demonstrate that parameter-efficient methods can achieve competitive performance while significantly reducing computational cost. In particular, LoRA achieved performance close to full fine-tuning while training only a small fraction of the model parameters, and ReFT achieved similar accuracy with even fewer trainable parameters. SMART regularization consistently improved performance for paraphrase detection and slightly improved generation quality for sonnet generation by encouraging stable hidden representations. For generative alignment, DPO proved effective in improving stylistic quality of generated sonnets, achieving the highest chrF score among our experiments.

However, our experiments also reveal important limitations. GRPO did not improve performance in our setting and instead slightly degraded generation quality. This highlights the sensitivity of reinforcement learning methods to reward design and data scale. In the small-data regime of sonnet generation, reward signals that are not perfectly aligned with evaluation metrics can cause the model to drift away from a strong supervised solution. Additionally, while parameter-efficient methods improve efficiency, they sometimes exhibit a small performance gap compared with full fine-tuning. We also observe an overfitting issue when applying SMART to paraphrase detection. Although SMART achieves high development accuracy (around 0.91), its test performance falls slightly below the baseline. This suggests that the model may have overfit to the development set during hyperparameter tuning, leading to weaker generalization on unseen data.

Overall, this project provided valuable insight into the trade-offs between efficiency, robustness, and performance when adapting large language models to downstream tasks. Future work could explore better reward design for reinforcement learning methods, larger generative datasets for stable policy optimization, and hybrid approaches that combine parameter-efficient tuning with preference-based alignment. Such directions may further improve both the controllability and efficiency of adapting large language models to specialized tasks.

Team contributions

Yixiao implemented the baseline GPT-2 model using the AdamW optimizer and led the PEFT experiments, integrating LoRA and ReFT across the paraphrase detection and sonnet generation tasks. **Shirley** implemented the baseline for paraphrase detection and led the integration of SMART Regularization. Additionally, they spearheaded the implementation of GRPO for sonnet generation. **Pengyu** implemented the baseline sonnet generation framework and led the training and iterative refinement of the DPO pipeline. Their work focused on improving the output of the generative model through repeated optimization cycles.

References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [5] Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. Smart: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437*, 2019.
- [6] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.
- [7] Zhengxuan Wu, Aryaman Arora, Zheng Wang, Howard Hu, Hadas Kotek, Christopher D Snee, Christopher D Manning, Christopher Potts, and Atticus Geiger. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*, 2024.
- [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. *Advances in Neural Information Processing Systems*, 36, 2024.
- [9] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wang, Damai Dai, Daya Guo, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [10] Maja Popović. chrF: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.
- [11] 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. *arXiv preprint arXiv:2305.18290*, 2023.
- [12] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

A Appendix (optional)

Table 3: Hyperparameters for the Selected Models Across Tasks

Task / Configuration	Learning Rate	Batch Size	Epochs	Best Epoch
SST (Last Linear Layer)	1e−3	16	20	9
SST (Full Model)	1e−5	8	10	7
CFIMDB (Last Linear Layer)	1e−3	8	30	24
CFIMDB (Full Model)	1e−5	8	10	4
Paraphrase Detection	1e−5	16	20	13
Sonnet Generation	1e−5	8	20	13

λ_{SMART}	ϵ	Batch	LR	Train Loss	Dev Loss	Dev PPL	chrF
0.00	0.01	2	$1e^{-5}$	4.101	4.160	64.547	42.12
0.02	0.01	2	$1e^{-5}$	4.109	4.159	64.490	42.46
0.05	0.01	2	$1e^{-5}$	4.120	4.158	64.411	41.93
0.03	0.01	2	$1e^{-5}$	4.113	4.159	64.463	42.37
0.01	0.01	2	$1e^{-5}$	4.105	4.160	64.518	42.63
0.01	0.005	2	$1e^{-5}$	4.105	4.160	64.518	42.54
0.01	0.020	2	$1e^{-5}$	4.105	4.160	64.518	42.63

Table 4: SMART regularization results for sonnet generation under different hyperparameter settings.