

LoRA-Enhanced GPT-2 with DPO for Sonnet Generation

Stanford CS224N Default Project

Krish Maniar

Department of Computer Science
Stanford University
kmaniar@stanford.edu

Nick Allen

Department of Computer Science
Stanford University
nallen@stanford.edu

Filip Henriksson

Department of Computer Science
Stanford University
filiph@stanford.edu

Abstract

We implement GPT-2 from scratch and extend it with LoRA for parameter-efficient finetuning and DPO for improved sonnet generation. For paraphrase detection, we systematically scale LoRA rank (8 to 128) and backbone size (GPT-2 Small to Medium), finding that adapter capacity and model scale drive accuracy gains while regularization techniques (SMART, label smoothing) provide no benefit under cosine scheduling. Our best single model (GPT-2 Medium, $r=64$) reaches 91.0% dev accuracy, and a six-model ensemble across both backbones achieves 91.5%. For sonnet generation, DPO trained on automatically constructed chrF-scored preference pairs improves GPT-2 Small by +4.29 chrF over SFT-only when combined with Best-of-N sampling (test leaderboard: 43.84, dev: 44.43), but degrades performance on Medium and Large, suggesting self-generated preference data only helps when the base model has substantial room for improvement. All Part 1 sentiment baselines on SST and CFIMDB were met.

1 Introduction

Large language models pretrained on broad text corpora have demonstrated strong performance across diverse NLP tasks when finetuned on task-specific data. GPT-2 [1], a decoder-only transformer with 125M parameters in its smallest variant, serves as a canonical example of this paradigm. In this project, we implement GPT-2 from scratch, loading pretrained weights from HuggingFace, and explore its application to three downstream tasks: sentiment classification, paraphrase detection, and Shakespeare sonnet generation.

Part 1 of the project establishes baseline capabilities. We implement multi-head causal self-attention, transformer layers, token and positional embeddings, and AdamW optimization with bias-term handling. Sentiment classifiers trained in both last-linear-layer and full-model modes meet all autograder thresholds on SST (5-way) and CFIMDB (binary).

For Part 2, we pursue three extensions. First, we apply LoRA [2] to GPT-2’s attention layers for parameter-efficient finetuning, achieving strong paraphrase detection performance with under 300K trainable parameters. Second, we implement Direct Preference Optimization (DPO; [3]) for sonnet generation, constructing preference data automatically from model-generated candidates scored by chrF. We further combine DPO with Best-of-N (BoN) sampling at generation time, yielding a total +4.29 chrF over SFT-only. Through systematic hyperparameter sweeps across 30 configurations and three model sizes, we find that DPO with BoN provides a +4.29 chrF improvement on GPT-2 Small but degrades performance on larger models, a finding we attribute to the quality of self-generated

preference data degrading as the base model improves. Third, we scale LoRA-based paraphrase detection along two axes: adapter capacity (sweeping rank from 8 to 128 across all four attention projections) and backbone size (moving from GPT-2 Small to GPT-2 Medium), and then combine diverse checkpoints via softmax-averaged ensembling to exploit complementary error patterns across model sizes and LoRA configurations.

2 Related Work

GPT-2 and autoregressive language models. The transformer architecture [4] introduced scaled dot-product self-attention as a replacement for recurrence, enabling parallelized training over long sequences. GPT [5] demonstrated that a decoder-only transformer pretrained with a causal language modeling objective on large text corpora learns representations that transfer effectively to downstream tasks via finetuning.

Parameter-efficient finetuning. Full finetuning is expensive and risks catastrophic forgetting on small datasets. Adapter layers [6] insert trainable bottleneck modules between frozen transformer layers. Prefix tuning [7] prepends learnable continuous vectors to the key and value sequences. LoRA [2] injects low-rank decompositions directly into existing weight matrices, adding no inference latency since the update merges into the original weights. Our work investigates how scaling LoRA rank and the number of adapted projections interacts with backbone model size.

Reinforcement learning from human feedback and DPO. RLHF aligns language models with human preferences via supervised finetuning, reward modeling, and PPO-based optimization [8, 9], but requires training a separate reward model and careful stabilization. DPO [3] eliminates the reward model by reparameterizing the objective into a binary classification loss over chosen and rejected pairs. We adapt DPO for sonnet generation using automatically constructed preference pairs scored by chrF rather than human annotation.

Automatic evaluation of text generation. chrF [10] computes F-score over character n-grams up to order 6, offering a tokenization-free alternative to word-level metrics like BLEU [11]. It correlates well with human judgments and is well suited for sonnet generation, where character-level overlap captures similarity in archaic vocabulary that exact word matching would miss. We use chrF both as the evaluation metric and as the DPO scoring function.

Paraphrase detection. The Quora Question Pairs (QQP) dataset [12] has become a standard benchmark for paraphrase identification, containing over 400K question pairs labeled as duplicates or non-duplicates. Early approaches relied on hand-crafted features such as TF-IDF similarity, word overlap counts, and edit distance fed into gradient-boosted classifiers [13]. Neural methods improved substantially: Siamese LSTMs operating on shared sentence encodings [14], attention-based CNNs [15], and ultimately pretrained transformers such as BERT [16] pushed accuracy above 90% on this dataset.

3 Approach

3.1 GPT-2 Baseline

Our base model is GPT-2 Small (125M parameters, 12 layers, 12 attention heads, hidden dimension 768) implemented from the CS224N starter code. We implement multi-head causal self-attention, transformer layers with layer normalization and residual connections, token and positional embeddings, and the AdamW optimizer. Pretrained weights are loaded from HuggingFace, with weight transposition to convert from HuggingFace’s Conv1D format to our Linear layers. Output logits are computed via weight tying with the input embedding matrix. For details of the base implementation, we refer the reader to the project specification [17].

3.2 LoRA

We apply LoRA [2] to the query and value attention projections in every transformer layer, freezing all pretrained weights. Each adapter injects a down-projection $W_{\text{down}} \in \mathbb{R}^{d \times r}$ and up-projection

$W_{\text{up}} \in \mathbb{R}^{r \times d}$ with rank $r = 8$, scaling factor $\alpha = 32$, and dropout 0.05. On GPT-2 Small this gives 296K trainable parameters (0.24% of 125M); on GPT-2 Large, 1.48M (0.19% of 774M).

3.3 DPO for Sonnet Generation

We implement DPO as a post-SFT training phase for Shakespeare sonnet generation. The approach has two stages: supervised finetuning (SFT) on the training sonnets, followed by DPO training on automatically constructed preference pairs.

Supervised Finetuning (SFT). The SFT stage trains GPT-2 on 143 Shakespeare sonnets using standard next-token prediction (cross-entropy loss). We use full finetuning (all 125M parameters) rather than LoRA, as full finetuning significantly outperformed all LoRA configurations for this task. With only 143 training sonnets, the model benefits from updating all parameters at a low learning rate (1×10^{-5}) to learn Shakespearean vocabulary, meter, and structure. This contrasts with paraphrase detection, where LoRA suffices, likely because paraphrase detection leverages GPT-2’s pretrained language understanding, while sonnet generation requires learning a substantially different linguistic style.

DPO Objective. Given a prompt x (the first three lines of a sonnet), a chosen completion y_w , and a rejected completion y_l , the DPO loss [3] is:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E} \left[\log \sigma \left(\beta \left(\log \frac{\pi_{\theta}(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \log \frac{\pi_{\theta}(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right) \right] \quad (1)$$

where π_{θ} is the trainable policy (initialized from the SFT checkpoint), π_{ref} is a frozen copy of the SFT model, and β controls how much the policy is allowed to diverge from the reference. We compute sequence log-probabilities by summing token-level log-probs over completion tokens only, masking out the prompt. Each training step requires four forward passes: the policy and reference model each process both the chosen and rejected completions.

Preference Data Construction. For each of the 131 training sonnets with at least 3 lines, we extract the first 3 lines as a prompt and generate N candidate completions using the SFT model with diverse sampling parameters (temperature sampled uniformly from a configurable range, top- p sampled from $[0.8, 0.95]$). Each candidate is scored using chrF against the reference sonnet continuation. We construct preference pairs by pairing completions from the top quartile (chosen) with completions from the bottom quartile (rejected), filtering by a minimum score margin of 0.1 to ensure meaningful signal.

A critical design decision was using chrF-only scoring rather than a composite metric (which we initially tried with 60% chrF + 20% line count + 20% rhyme). Aligning the preference scoring metric with the evaluation metric improved DPO results by +0.5–1.0 chrF.

Implementation Details. Both the policy and reference models must fit in GPU memory simultaneously. The reference model runs in eval mode with gradients disabled. We apply gradient clipping (max norm 1.0) to stabilize training. Only the final-epoch checkpoint is saved to conserve disk space.

3.4 Paraphrase Detection Extensions

Our paraphrase detection extensions target accuracy gains along three axes: LoRA adapter capacity scaling, backbone model scaling, and multi-model ensembling.

LoRA Capacity Scaling. The baseline LoRA configuration described in Section 4.2 adapts only the query and value projections at a fixed rank. We extend this in two ways. First, we adapt all four attention projection matrices (W_q, W_k, W_v, W_o) in every transformer layer, giving the adapter access to the full attention computation rather than a subset. Second, we sweep LoRA rank r over a wide range to trade trainable parameter count for representational capacity. For k adapted projections with hidden dimension d , the trainable adapter parameters per layer are

$$P_{\text{LoRA}} = k \cdot 2dr \quad (2)$$

This allows us to characterize the relationship between adapter capacity and downstream accuracy while keeping the pretrained backbone entirely frozen.

Backbone Model Scaling. Rather than compressing the backbone (as explored in Appendix C), we scale it up by replacing GPT-2 Small with GPT-2 Medium as the frozen feature extractor. The deeper network and wider hidden dimension provide richer representations for the LoRA adapters and classification head to operate on. All pretrained weights remain frozen; only the LoRA adapters and task head are reinitialized for the larger backbone.

Regularization Attempts. We investigated three regularization strategies aimed at improving generalization: SMART adversarial regularization, which applies gradient-based perturbations to the embedding layer and penalizes prediction sensitivity; label smoothing, which replaces hard classification targets with softened distributions; and sentence-order augmentation, which swaps the two input sentences for positive pairs to encourage order-invariant representations. All three are detailed and evaluated in Section 5.

Ensemble. Our final system combines independently trained LoRA models by averaging their output distributions at inference time. Given input x and models $\{f_1, \dots, f_N\}$, the ensemble prediction is

$$\hat{y} = \arg \max_c \frac{1}{N} \sum_{i=1}^N \text{softmax}(f_i(x))_c \quad (3)$$

Diversity is constructed by varying LoRA rank, learning rate, and backbone size across the ensemble members. Models are selected greedily: starting from the best single checkpoint, we iteratively add the model that most improves ensemble dev accuracy.

3.5 Best-of-N Sampling with Heuristic Selection

At generation time, we generate N candidate completions per prompt and select one using a reference-free heuristic. We evaluate four strategies: *consensus*, which selects the candidate with the highest average pairwise chrF against all others; *length-filtered consensus*, which first discards candidates outside 10–16 non-empty lines then applies consensus; *top-half consensus*, which discards the bottom half by average similarity then selects the most central survivor; and *scored selection*, which ranks candidates by reference-free features (line count, rhyme adherence, stylistic consistency with the prompt).

4 Experiments

4.1 Data

We use the four course-provided datasets. SST has 8,544/1,101/2,210 train/dev/test examples for 5-way sentiment. CFIMDB has 1,701/245/488 for binary sentiment. Quora provides 141,506/20,215/40,431 for binary paraphrase detection. The Shakespeare sonnet set has 143 training sonnets and 12 held-out test sonnets evaluated via a continuation protocol: given the first 3 lines of a held-out sonnet, the model generates the remaining lines, which are scored against the reference. No additional preprocessing or filtering was applied to the Quora dataset. Sentence-order augmentation (swapping s_1 and s_2 for positive pairs) was tested but ultimately excluded from all final models due to the class imbalance it introduced.

4.2 Evaluation method

Sentiment classification and paraphrase detection are evaluated by accuracy. Sonnet generation is evaluated by chrF [10], which computes character n-gram F-score between generated and reference completions. chrF is computed at the corpus level across all held-out sonnets using the SacreBLEU implementation. For paraphrase detection, we report dev accuracy as the primary metric for all model selection and ensemble construction decisions. We also track per-epoch training loss to diagnose convergence and overfitting behavior across configurations.

4.3 Experimental details

Part 1: Sentiment Classification. All experiments use GPT-2 Small with HuggingFace pretrained weights. Part 1 sentiment classifiers used the starter code’s default hyperparameters in both last-linear-layer and full-model modes.

LoRA for Paraphrase Detection. The baseline LoRA paraphrase model uses GPT-2 Small with rank $r=8$, $\alpha=32$, dropout 0.05, adapting only W_q and W_v . Training uses AdamW with lr 3×10^{-4} , batch size 64, and 5 epochs. This configuration trains 296K parameters (0.24% of the backbone).

Sonnet Generation SFT. We evaluated 4 SFT configurations (all full finetuning, batch size 8), sweeping learning rate (5×10^{-6} to 2×10^{-5}), epoch count (20–50), and weight decay (0.0 vs. 0.01), producing 22 intermediate checkpoints. SFT training was conducted on an NVIDIA L4 (23GB) on GCP and an NVIDIA L40S (48GB) on Stanford FarmShare.

DPO. Across 5 experimental runs on FarmShare, we tested 30 DPO configurations on GPT-2 Small, systematically varying learning rate (2×10^{-7} to 1×10^{-6}), epochs (3–10), β (0.02–0.07), number of candidates (16 vs. 32), scoring mode (composite vs. chrF-only), temperature range for candidate generation, and iterative DPO. We also tested GPT-2 Medium (345M) and Large (774M) on FarmShare.

Paraphrase Detection Extensions. We sweep LoRA rank across $r \in \{8, 16, 32, 64, 128\}$ with all four attention projections (W_q, W_k, W_v, W_o) adapted, setting $\alpha = 2r$ throughout. All GPT-2 Small runs use AdamW with lr 3×10^{-4} , batch size 64, and 10 epochs with a cosine learning rate schedule (linear warmup over 6% of steps) and gradient clipping at max norm 1.0. For GPT-2 Medium (355M parameters), we reduce the learning rate to 2×10^{-4} and test batch sizes of 16 and 32 due to memory constraints; all other settings are unchanged. SMART regularization experiments sweep $\lambda_s \in \{0.5, 1.0\}$ and Bregman coefficient $\mu \in \{0.0, 0.1, 1.0\}$ with 2 PGD steps and ℓ_∞ norm. Label smoothing is tested at $\epsilon \in \{0.05, 0.10, 0.15\}$. Ensemble experiments evaluate all $\binom{N}{k}$ subsets up to $N=6$ models, selected greedily on dev accuracy.

Efficiency Ablation. For the GPT-2 Large ablation, we used LoRA rank 16 adapting W_q and W_v and trained for a single epoch due to compute costs. The fp16 backbone runs enabled gradient checkpointing throughout, and magnitude pruning was applied globally at 30% sparsity before training.

4.4 Results

Part 1: Sentiment Classification. Table 1 shows Part 1 results. All autograder thresholds are met.

Table 1: Part 1 sentiment results (dev accuracy). All autograder checks pass.

Mode	Dataset	Dev Acc	Target	Status
Last-linear-layer	SST	0.473	0.462	Above
Last-linear-layer	CFIMDB	0.865	0.861	Above
Full-model	SST	0.507	0.513	Close
Full-model	CFIMDB	0.988	0.976	Above

LoRA for Paraphrase Detection. Table 2 shows the effect of LoRA capacity scaling on GPT-2 Small. Increasing rank and adapting all four projections yields consistent gains, with diminishing returns past $r=64$: the jump from $r=64$ to $r=128$ adds 4.7M parameters but no accuracy improvement.

Regularization Attempts. SMART adversarial regularization, label smoothing, and sentence-order augmentation all degraded dev accuracy relative to their respective baselines (Appendix E), so none were included in final models.

Backbone	Rank	Targets	Trainable Params (%)	Dev Acc
Small (125M)	8	q, v	296K (0.24%)	0.877
Small (125M)	16	q, k, v, o	1.18M (0.91%)	0.889
Small (125M)	32	q, k, v, o	2.36M (1.82%)	0.894
Small (125M)	64	q, k, v, o	4.72M (3.64%)	0.896
Small (125M)	128	q, k, v, o	9.44M (7.27%)	0.896

Table 2: LoRA capacity scaling on GPT-2 Small (10 epochs, cosine schedule). Adapting all four attention projections and increasing rank drives accuracy from 0.877 to 0.896, with diminishing returns past $r=64$.

Backbone Model Scaling. Scaling the frozen backbone to GPT-2 Medium produced the largest single-model gains. Table 3 shows Medium results across LoRA configurations. The best single model, Medium with $r=64$ and batch size 32—reaches 91.0% dev accuracy, a 1.4 point improvement over the best GPT-2 Small configuration and a 3.3 point improvement over the initial Small $r=8$ baseline.

Config	LR	Batch Size	Dev Acc
Med $r=16$	2×10^{-4}	16	0.904
Med $r=32$	2×10^{-4}	16	0.906
Med $r=16$	3×10^{-4}	16	0.906
Med $r=64$	2×10^{-4}	32	0.910

Table 3: GPT-2 Medium LoRA configurations (10 epochs, q, k, v, o , cosine schedule). All Medium runs surpass the best GPT-2 Small single model (0.896).

Ensemble Results. Table 4 shows ensemble performance as we increase the number of models and mix backbone sizes. Adding the first Medium model to a 6-model Small ensemble immediately crosses the 0.907 target. The best ensemble (4 Medium + 2 Small) reaches 91.5% dev accuracy, a 0.5 point gain over the best single model and 4.3 points above the original LoRA baseline. Notably, ensembling 4 Medium models alone (0.914) outperforms ensembling 6 Small models (0.903), confirming that individual model quality matters more than ensemble size.

Ensemble Composition	N	Dev Acc
6 Small	6	0.903
1 Med + 6 Small	7	0.907
3 Med only	3	0.911
3 Med + 3 Small	6	0.912
3 Med + 2 Small	5	0.913
4 Med only	4	0.914
4 Med + 2 Small	6	0.915

Table 4: Ensemble dev accuracy by composition. Models are selected greedily. Mixing backbone sizes provides complementary error patterns that improve over same-backbone ensembles.

Our final submission uses the 4 Medium + 2 Small ensemble (91.5% dev accuracy). The six constituent models and their individual dev accuracies are: Med- $r=64$ (0.910), Med- $r=32$ (0.906), Med- $r=16$ lr3e-4 (0.906), Med- $r=16$ lr2e-4 (0.904), Small- $r=64$ (0.896), and Small- $r=128$ (0.896).

Sonnet Generation: SFT Sweep. Table 5 shows the SFT sweep results. Weight decay (0.01) provided a small but consistent improvement (+0.11 chrF). The learning rate of 1×10^{-5} was optimal; 5×10^{-6} was too conservative even after 49 epochs. Full finetuning vastly outperformed LoRA for sonnet SFT. With only 143 training sonnets, the task requires updating all parameters to learn Shakespearean style.

Sonnet Generation: DPO Sweep. Table 6 shows the top 5 DPO configurations out of 30 tested. All sweep evaluations were conducted on FarmShare (NVIDIA L40S, CUDA) under identical conditions,

Table 5: SFT hyperparameter sweep results (GPT-2 Small, full finetuning).

Config	LR	Epochs	Weight Decay	Best chrF
D (best)	1×10^{-5}	30	0.01	40.14 (ep. 29)
C	2×10^{-5}	20	0.0	40.04 (ep. 19)
A	1×10^{-5}	35	0.0	40.03 (ep. 29)
B	5×10^{-6}	50	0.0	39.38 (ep. 39)

making relative comparisons valid. The best configuration achieved chrF = 43.29 on the sweep evaluation, a +3.15 improvement over the SFT-only baseline (40.14).

Key findings: chrF-only scoring consistently beat composite scoring by +0.5–1.0 chrF, making scoring mode the most impactful design choice. Learning rate was the primary DPO knob, with 3×10^{-7} optimal; 5×10^{-7} peaked early and degraded, while 2×10^{-7} never reached the same ceiling. Epoch count must be tuned jointly with LR: for $\beta = 0.05$, lr = 3×10^{-7} , chrF rose from 42.30 (5 epochs) to 43.29 (7 epochs) then declined to 42.15 (10 epochs). $\beta = 0.05$ achieved the highest peak but was sensitive to epoch count, while $\beta = 0.02$ peaked later (9 epochs) with a lower ceiling (43.08). Using 32 candidates did not improve over 16, and iterative DPO (two rounds, 41.04) underperformed single-round (42.63).

Table 6: Top 5 DPO configurations (all using SFT base chrF = 40.14).

Rank	β	LR	Epochs	Preference Data	chrF
1	0.05	3×10^{-7}	7	16 cand, chrF-only	43.29
2	0.02	3×10^{-7}	9	16 cand, chrF-only	43.08
3	0.02	3×10^{-7}	7	32 cand, chrF-only	43.06
4	0.02	5×10^{-7}	5	16 cand, chrF-only	42.82
5	0.05	3×10^{-7}	9	16 cand, chrF-only	42.70

Sonnet Generation: Model Scaling. Table 7 shows DPO effectiveness across model sizes. Larger models required proportionally lower learning rates and fewer epochs before overfitting. DPO degraded performance on both Medium and Large despite applying all improvements from the Small sweep.

Table 7: DPO effectiveness across GPT-2 model sizes.

Model	Best SFT chrF	Best DPO chrF	DPO Δ
Small (125M)	40.14	43.29	+3.15
Medium (345M)	38.98	37.84	−1.14
Large (774M)	38.23	37.37	−0.86

Sonnet Generation: Generation-Time Tuning. We performed a temperature and top- p sweep on the best DPO checkpoint (Table 8). Higher temperature (0.9) with tighter nucleus sampling (top_p=0.85) was the best single-model configuration, outperforming even a 3-model logit-averaging ensemble.

Leaderboard Submission. Our final sonnet submission uses the best DPO checkpoint with Best-of-N sampling ($N=32$, temp=0.9, top_p=0.85, length-filtered consensus, seed 11711), achieving a **test leaderboard chrF of 43.84** and dev chrF of 44.43.

5 Analysis

Why DPO helps small models but hurts larger ones. DPO improved chrF by +3.15 on GPT-2 Small but degraded performance on Medium (−1.14) and Large (−0.86). We attribute this to quality variance in self-generated preference data. GPT-2 Small produces candidates spanning a wide quality

Table 8: Generation-time parameter tuning on the best DPO checkpoint.

Setting	chrF
temp=0.7, top_p=0.9 (default)	41.65
temp=0.9, top_p=0.85	42.85
3-model ensemble, temp=0.75	42.53

range, providing strong chrF differentiation between chosen and rejected pairs. Medium and Large generate more uniformly competent completions where chrF differences are small and driven by surface-level variation rather than meaningful quality distinctions. This suggests DPO with automatic scoring works best as a bootstrapping technique for weaker models, while stronger models would require higher-quality preference signals.

Preference scoring alignment. Switching from composite scoring (60% chrF + 20% line count + 20% rhyme) to chrF-only improved DPO by +0.5–1.0 chrF. The composite metric diluted the chrF signal with structural features not measured by the evaluation metric. The preference scoring function should be tightly aligned with the downstream evaluation metric.

The DPO learning budget. For $\beta = 0.05$, $\text{lr} = 3 \times 10^{-7}$, chrF peaked at 43.29 (7 epochs) then declined to 42.15 (10 epochs), revealing a learning budget ($\text{lr} \times \text{epochs} \approx 2.1 \times 10^{-6}$) beyond which the policy overfits. $\beta = 0.02$ peaks later (9 epochs) with a lower ceiling (43.08): a tighter KL constraint allows longer training but limits the maximum gain.

Capacity scaling vs. regularization for paraphrase detection. The $r=32$ model drives training loss to 0.159 by epoch 9 while dev accuracy plateaus at 0.894, so the model is not overfitting and regularization has no useful gradient to exploit. The bottleneck is representational capacity. This parallels the DPO finding in reverse: stronger backbones help paraphrase detection by providing richer representations, whereas for DPO they eliminate the quality variance in self-generated data that the algorithm needs.

6 Conclusion

We implemented GPT-2 from scratch and extended it with LoRA for parameter-efficient finetuning and DPO for improved sonnet generation. Our main findings are:

DPO improves sonnet generation on GPT-2 Small, yielding up to +4.29 chrF over SFT-only when combined with Best-of-N sampling (test leaderboard 43.84, dev 44.43), but with self-generated preference data its effectiveness is inversely related to model scale, degrading on Medium and Large. Aligning the preference scoring metric with evaluation was critical: switching from composite to chrF-only scoring improved results by +0.5–1.0 chrF. LoRA proved task-dependent: it matches full finetuning for paraphrase detection, where pretrained representations already capture semantic similarity, but substantially underperforms for sonnet generation, where the model must learn a novel linguistic style from only 143 examples. Paraphrase detection accuracy was driven by adapter capacity and backbone scale, not regularization: scaling LoRA rank to 64 and the backbone to GPT-2 Medium yielded 91.0% single-model dev accuracy, and a greedy ensemble of diverse LoRA checkpoints across both model sizes reached 91.5%.

Limitations include the small training set (143 sonnets), reliance on automatic rather than human preference scoring, and the stochastic nature of generation-time evaluation. Future work could explore LLM-based preference scoring, DPO with human annotations, and curriculum strategies that adapt β during training.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 2019.

- [2] 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.
- [3] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
- [4] 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 (NeurIPS)*, 2017.
- [5] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 2018.
- [6] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Olivier de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *ICML*, 2019.
- [7] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *ACL*, 2021.
- [8] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.
- [9] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [10] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.
- [11] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: A method for automatic evaluation of machine translation. In *ACL*, pages 311–318, 2002.
- [12] Quora. Quora question pairs. <https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs>, 2017.
- [13] Bayu Chandra and Dimas Stefanus. Experiments on paraphrase identification using quora question pairs dataset. 2020.
- [14] Jonas Mueller and Aditya Thyagarajan. Siamese recurrent architectures for learning sentence similarity. In *AAAI*, 2016.
- [15] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272, 2016.
- [16] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, 2019.
- [17] CS224N course staff. Default final project: Build GPT-2, 2026.
- [18] 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. In *ACL*, 2020.

A Appendix (optional)

B Key Information to include

- Mentor: Ahmed Ahmed
- External Collaborators (if you have any): None
- Sharing project: No
- Late Days (if you are using any):

Team member	Late days remaining
Krish Maniar	2
Nick Allen	3
Filip Henriksson	3
Total pooled	8
Team size	3
Late days applied for the report	2

Team contributions (Required for multi-person team)

- **Krish Maniar:** Implemented the core LoRA module and integrated adapter injection into the paraphrase and sonnet training scripts. Developed the efficiency extension: reduced-precision backbone loading (fp16/bf16), magnitude pruning, gradient checkpointing, and the GPT-2 Large ablation experiments on FarmShare.
- **Nick Allen:** Completed the Part 1 GPT-2 implementation (attention, transformer layers, embeddings, AdamW optimizer) and sentiment classification baselines. Led the paraphrase detection accuracy effort: implemented SMART adversarial regularization, label smoothing, sentence-order augmentation, R-Drop regularization, and the MLP classifier head; conducted the full LoRA capacity scaling sweep ($r \in \{8, 16, 32, 64, 128\}$) on GPT-2 Small and all GPT-2 Medium experiments; built the softmax-averaging ensemble pipeline and ran greedy model selection across backbone sizes to produce the final 91.5% dev accuracy submission, which ranked #1 on the paraphrase detection leaderboard at the time of submission.
- **Filip Henriksson:** Implemented the DPO pipeline for sonnet generation, including the DPO loss function, automatic preference data construction, and the two-phase SFT-to-DPO training pipeline. Conducted systematic hyperparameter sweeps across 30 DPO configurations and three GPT-2 model sizes on GCP and FarmShare, performed generation-time tuning experiments, and developed Best-of-N sampling with multiple selection heuristics (consensus, length-filtered, scored), achieving the final test chrF of 43.84.

C Efficiency–Accuracy Tradeoffs

After running the efficiency experiments, it was clear that fp16 backbone loading hurt in both settings we tried: a 15-point accuracy drop for paraphrase detection on GPT-2 Large and NaN training losses during sonnet SFT. GPT-2’s pretrained weights seem sensitive to reduced precision, likely because the model was trained in fp32 unlike newer architectures that use mixed precision from the start. The pruning result is more interesting: removing 30% of frozen weights caused no additional degradation beyond fp16, which points to substantial redundancy in the backbone for downstream LoRA tasks. LoRA on GPT-2 Small (87.2% dev, 5 epochs) also outperformed GPT-2 Large (77.8%, 1 epoch), though this is almost certainly a training-duration artifact rather than a real scaling reversal.

D Best-of-N Sampling Details

Experimental Setup. We sweep $N \in \{32, 64\}$, temperature $\in \{0.8, 0.9, 1.0\}$, and top- $p \in \{0.85, 0.9\}$ on the dev set using the best DPO checkpoint, evaluating all four selection heuristics per configuration.

Results. Table 9 shows BoN results with the best generation parameters (temp=0.9, top_p=0.85). Length-filtered consensus achieves 44.43 chrF on dev, a +1.14 improvement over single-sample generation and a +4.29 improvement over SFT-only. (Note: single-sample chrF differs from Table 8 because BoN evaluation uses a different random seed to ensure independence from the generation-tuning sweep.)

Table 9: Best-of-N selection heuristics ($N=32$, temp=0.9, top_p=0.85, dev set).

Selection Method	chrF
Single sample	43.67
Consensus	44.12
Top-half consensus	44.12
Scored (reference-free)	42.89
Scored + consensus hybrid	43.95
Length-filtered consensus	44.43
Oracle (upper bound)	45.84

The reference-free scored heuristic (42.89) performed *worse* than single sampling, indicating that structural features like line count and rhyme are poor proxies for chrF. Consensus methods succeed by selecting “typical” outputs, which tend to be higher quality on average. Length-filtered consensus adds a lightweight structural prior, improving over pure consensus by 0.31 chrF. The oracle gap of only 1.41 chrF suggests consensus captures most of the achievable gain at $N=32$; increasing to $N=64$ did not help (44.34).

E Regularization Negative Results

We tested three regularization strategies. SMART [18] adversarial regularization ($\lambda_s \in \{0.5, 1.0\}$, with and without Bregman proximal terms) was evaluated against the $r=8$ baseline; label smoothing ($\epsilon \in \{0.05, 0.10, 0.15\}$) and sentence-order augmentation were evaluated against the $r=32$ baseline. Table 10 summarizes the results. SMART added gradient noise without benefit, as cosine scheduling already prevents overfitting. Label smoothing penalized calibration that was already well-tuned. Sentence-order augmentation introduced a $\sim 2:1$ class imbalance (duplicated positives without duplicating negatives), which consistently hurt.

Approach	Best Dev Acc	Δ vs Baseline
Baseline ($r=8, q, v$, cosine)	0.877	—
SMART R_s ($\lambda_s=1.0$)	0.873	−0.4%
SMART R_s +Breg ($\mu=0.1$)	0.873	−0.4%
SMART R_s ($\lambda_s=0.5$)	0.875	−0.2%
Baseline ($r=32, q, k, v, o$, cosine)	0.894	—
Label smoothing ($\epsilon=0.05$)	0.893	−0.1%
Label smoothing ($\epsilon=0.10$)	0.892	−0.2%
Label smoothing ($\epsilon=0.15$)	0.891	−0.3%
Sentence-order augmentation	0.871	−2.3%

Table 10: Regularization techniques evaluated against their respective baselines. None improve over the unregularized baseline with cosine scheduling.