

Efficient Adaptation and Structure-Aware Post-Training of GPT-2 for Paraphrase Detection and Sonnet Generation

Stanford CS224N Default Project

Koa Chang

Department of Computer Science
Stanford University
koachang@stanford.edu

Brandon Kunitzer

Department of Computer Science
Stanford University
kunitzer@stanford.edu

Abstract

We study how far we can push a from-scratch CS224N GPT-2 implementation on two downstream tasks from the default final project: cloze-style paraphrase detection and Shakespearean sonnet continuation. Our main question was which extensions improve final submission quality under a single-GPU compute budget. For paraphrase detection, we implemented prompt variants, LoRA, and bidirectional augmentation. The best saved dev predictions came from a LoRA-adapted GPT-2 large model with prompt tuning, improving dev accuracy from 89.6% for our retained baseline to 90.8%. For sonnet generation, the largest gains came from reducing train/test mismatch: instead of unconditional language modeling, we trained on the actual continuation objective, masked prompt tokens out of the loss, and augmented the tiny training set with multiple prompt lengths. We then added DPO with heuristic negative continuations, structure-aware GRPO, and best-of- N reranking. The strongest exported dev score was 43.37 chrF, compared with 42.34 chrF for the unconditional full fine-tuning baseline. High-rank LoRA and more aggressive hard-negative DPO were less successful than expected.

1 Key Information

- **Mentor:** Wei Liu
- **External collaborators:** None
- **Sharing project:** No
- **Late days used:** 2

2 Introduction

The CS224N default project asks students to both implement GPT-2 and extend it on two very different downstream tasks: paraphrase detection and sonnet generation. In paraphrase detection, the model is evaluated on a single cloze-style decision, so prompt wording, label verbalization, and model size matter disproportionately. In sonnet generation, the model must maintain structure and tone over many generated tokens despite training on only 131 Shakespeare sonnets. In both settings, a naïve full fine-tuning baseline works, but it leaves clear room for better objective design and more compute-efficient adaptation.

Our project centered on two hypotheses. First, parameter-efficient fine-tuning should make it practical to scale from GPT-2 small to GPT-2 large on limited hardware, especially for paraphrase detection where the output space is tiny. Second, for sonnet generation, methods that more directly optimize the actual evaluation setting should matter more than generic language-model fine-tuning. That led

us to explore LoRA for both tasks, prompt engineering and prompt augmentation, DPO and GRPO post-training, curriculum-style conditioning, and decoding-side reranking.

Three findings stood out. For paraphrase detection, the best saved dev predictions in the repository improved accuracy from 89.6% to 90.8% by combining a larger backbone with LoRA and a better prompt. For sonnet generation, the biggest single gain came from matching training to evaluation by conditioning on the first few lines and scoring only the continuation. Finally, the final sonnet score depended not only on training but also on decoding: seed and candidate-count sweeps changed dev chrF by more than half a point, large enough to alter which checkpoint we submitted.

3 Related Work

Our work begins from GPT-2 fine-tuning in the standard pretrain-then-adapt paradigm introduced by Radford et al. [1]. The default project already supplies a full fine-tuning baseline for both cloze-style classification and autoregressive generation, so our work is best viewed as a set of targeted extensions to that baseline rather than a new architecture.

For efficient adaptation, we use LoRA [2], which freezes pretrained weights and learns low-rank updates for selected linear layers. This is attractive in our setting because it reduces optimization cost while still allowing us to experiment with larger GPT-2 checkpoints. We insert LoRA into attention and MLP projections rather than only the attention blocks.

For post-training, we explored both preference optimization and reinforcement learning. DPO [3] provides a stable way to fine-tune a model to prefer one continuation over another without training a separate reward model. GRPO [4], a PPO-style method related to policy optimization [5], removes the critic and instead uses group-relative advantages. These methods are particularly attractive for sonnet generation because chrF is non-differentiable and because stylistic constraints such as line count and rhyme can be expressed as reward features. Finally, our sonnet curriculum is motivated by curriculum learning [6]: on tiny data, ordering examples from easier style learning to harder conditioned continuation can improve optimization.

4 Approach

4.1 Paraphrase Detection

We formulate paraphrase detection as next-token prediction. Given a sentence pair (s_1, s_2) , we build a prompt and ask GPT-2 to generate either yes or no. Our implementation computes the final-token hidden state and projects it back into vocabulary space with GPT-2’s tied output embeddings. Training uses full-vocabulary cross-entropy at the answer position, with labels set to the token IDs for yes and no.

We implemented two prompt templates in code, prompt A,

Is " s_1 " a paraphrase of " s_2 "? Answer "yes" or "no":

and prompt B,

Do " s_1 " and " s_2 " have the same meaning? Answer yes or no:.

If we used a prompt to train, then we also used that same prompt to generate the test labels. We also implemented and tested bidirectional augmentation, which duplicates each training pair in reversed order.

For parameter-efficient adaptation, we implemented LoRA from scratch. For a frozen weight matrix W , LoRA learns

$$\Delta W = \frac{\alpha}{r} BA,$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$ and $B \in \mathbb{R}^{d_{\text{out}} \times r}$. In our code, adapters can be inserted into all attention projections and both MLP layers.

Finally, we added a GRPO stage. The GRPO implementation samples yes/no candidates, computes group-relative advantages, and regularizes to a frozen reference model. We kept this code path in the

repository, but did not end up using GRPO for the final paraphrase submission because we exhausted compute before completing a careful sweep.

As a note, we also experimented with changing the random seed.

4.2 Sonnet Generation

The starter baseline fine-tunes GPT-2 as an unconditional language model on full sonnets. However, evaluation conditions on the first three lines and scores only the continuation. We therefore changed the supervised objective to match the actual task: each training example is a full sonnet together with a prompt prefix, and prompt tokens are masked out of the loss so the model is only trained on the continuation. We also augment the tiny training corpus by using 2-, 3-, and 4-line prefixes, turning 131 training sonnets into 393 conditioned examples.

This gives the masked completion loss

$$\mathcal{L}_{\text{SFT}} = - \sum_{t \in \text{completion}} \log p_{\theta}(x_t \mid x_{<t}),$$

where prompt tokens do not contribute to the sum. We then explored three families of extensions.

DPO with heuristic negatives. Because the sonnet data contains only gold continuations, we synthesize rejected continuations by corrupting the gold suffix with operations such as line shuffling, dropping, repetition, line reversal, prompt repetition, and splicing in another sonnet’s suffix. DPO then prefers the gold continuation y^+ over a corrupted continuation y^- :

$$\mathcal{L}_{\text{DPO}} = - \log \sigma \left(\beta \left[(\log \pi_{\theta}(y^+) - \log \pi_{\text{ref}}(y^+)) - (\log \pi_{\theta}(y^-) - \log \pi_{\text{ref}}(y^-)) \right] \right).$$

GRPO with structure-aware reward. For GRPO, we combine chrF with structural heuristics:

$$R = 0.60 R_{\text{chrF}} + 0.15 R_{\text{prompt}} + 0.10 R_{\text{lines}} + 0.05 R_{\text{rhyme}} + 0.05 R_{\text{repeat}} + 0.05 R_{\text{length}}.$$

The reward features are implemented directly in our code: prompt preservation, closeness to 14 lines, approximate rhyme on Shakespearean line pairs, repetition penalties, and line-length regularity.

Best-of- N decoding and reranking. We found that output variance remained high even after better training. We therefore generate multiple candidates per prompt and rerank them with a weighted score combining average model log-probability, prompt preservation, line-count accuracy, rhyme proxy, repetition control, and line-length regularity. This reranker was part of our strongest final sonnet submission.

What we implemented ourselves. All LoRA, DPO, GRPO, corruption, curriculum, and reranking logic described above was implemented by us inside the provided starter framework, while pretrained GPT-2 weights were loaded from HuggingFace as required by the assignment.

5 Experiments

5.1 Data

For implementation validation we previously matched the expected trend on SST and CFIMDB, which gave us confidence that our GPT-2 and AdamW implementations were correct before moving to the extension-heavy downstream tasks.

For paraphrase detection, we use the Quora question-pairs data distributed with the project [7]: 283,003 train examples, 40,429 dev examples, and 80,858 test examples.

For sonnet generation, we use 131 Shakespeare sonnets for training and the provided 12-sonnet held-out dev set for evaluation. Each evaluation prompt consists of the first three lines of a sonnet, and the model must generate the remaining lines.

5.2 Evaluation Method

Paraphrase detection is evaluated by accuracy on the dev set. Sonnet generation is evaluated with chrF [8], which compares character n-grams between the generated continuation and the reference continuation.

5.3 Experimental Details

All substantial experiments were run on a single Google Cloud GPU. Paraphrase runs used AdamW with learning rate 10^{-5} for every trial (did briefly experiment with higher learning rate for LoRA but found it made the training unstable), batch sizes between 8 and 16 depending on model size, and 8–10 epochs. When using LoRA, we always used rank 256. The final saved paraphrase predictions came from a GPT-2 large LoRA run with prompt tuning.

For sonnet experiments, we experimented with full fine-tuning runs used GPT-2 large, batch size 2, gradient accumulation 2, cosine warmup, and gradient clipping at 1.0. The main supervised learning rates were 1.2×10^{-5} for the unconditional baseline, 8×10^{-6} for mixed-prompt curriculum stages, and 6×10^{-6} for exact three-line specialization. DPO used 3–4 epochs, learning rates around 3×10^{-6} , and β in the range 0.15–0.20. GRPO used 2 epochs, learning rate 2×10^{-7} , group size 4, and KL coefficient 0.06. Final decoding used temperature 0.85, top- p 0.92, repetition penalty 1.15, no-repeat trigrams, and 8–12 reranked candidates.

One practical detail mattered more than we expected for sonnet generation: scheduler, gradient clipping, and mixed precision were necessary for stable large-model training, but they were not the primary source of gains. The biggest improvements instead came from matching the task objective and from careful decoding.

5.4 Results

Table 1 summarizes the final models we would submit from the repository state.

Task	Submitted model to test leaderboard	Dev	Official test
Paraphrase	GPT-2 large + LoRA + prompt template A	90.7%	90.6%
Sonnet	Curriculum SFT + structure-aware GRPO + reranked best-of-12 decoding	43.37 chrF	42.923 chrF

Table 1: Final submitted configurations. The repository contains the unscored submission files for both tasks, but not the official leaderboard scores.

Here are the results of experiments run with Paraphrase detection (prompt A unless otherwise specified):

Paraphrase setting	Dev acc.	Notes
GPT-2 small full fine-tuning baseline	89.6%	retained dev output
GPT-2 medium full fine-tuning	90.5%	Submitted this to test leader board and scored 90.3%
GPT-2 large full fine-tuning	90.6%	Did not submit because large with LoRA did better
GPT-2 large LoRA	90.7%	Submitted this to test leader board and scored 90.6%
GPT-2 large LoRA + prompt template B	90.6%	very similar to default prompt A
GPT-2 large LoRA + seed tuning	90.8 %	best saved dev output, test results were 90.4%
Bidirectional training GPT-2 Medium LoRA	<89.6%	implemented but exact accuracy was poor so didn't note it

Table 2: Paraphrase settings explored during the project.

We also have graphs to show the experiments done with sonnet generation:

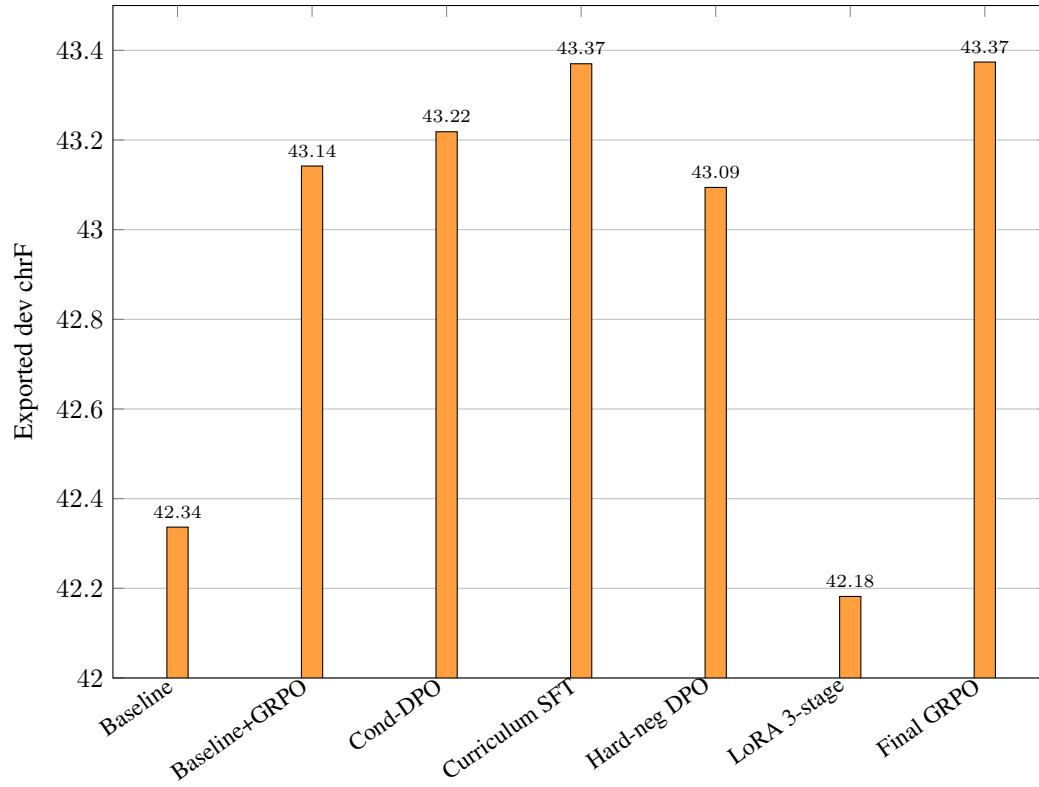


Figure 1: Representative sonnet-generation results on the held-out dev set. Conditioning the model on the actual continuation task was the biggest gain; later GRPO and reranking produced the best final score, while high-rank LoRA and harder DPO negatives underperformed.

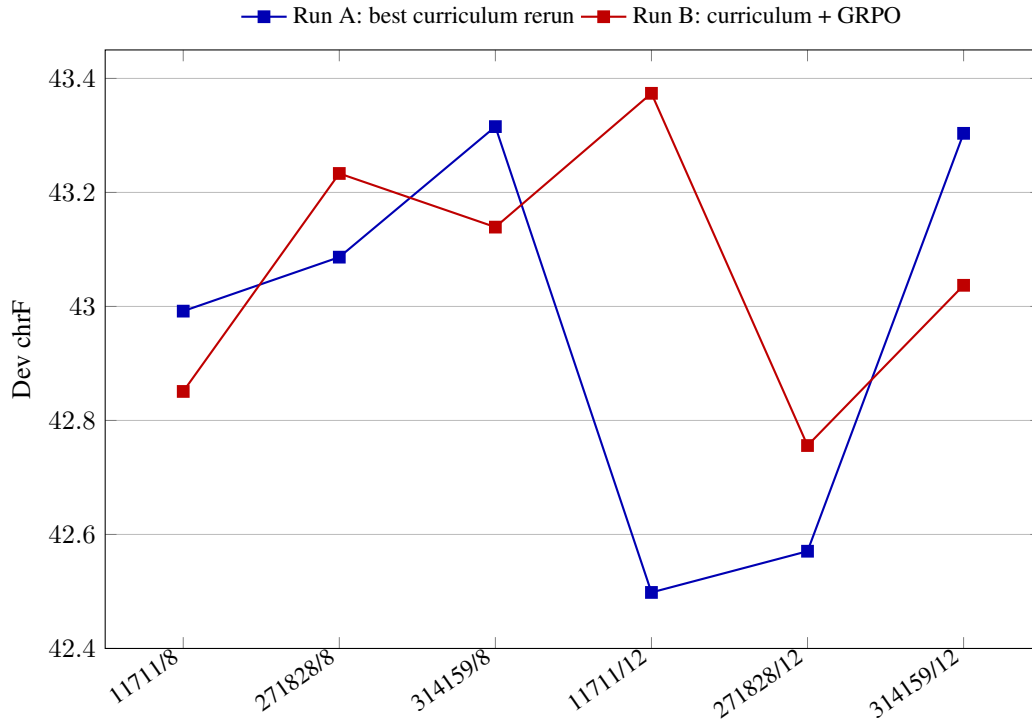


Figure 2: Decode sweeps for the final two sonnet candidates. Decoding alone changed dev chrF by more than 0.8 points, which is why our last-stage optimization emphasized reranking and seed/candidate sweeps rather than only additional training.

6 Analysis

Paraphrase detection. The paraphrase detection task underwent many changes with minimal improvement. Comparing the retained baseline dev predictions with the final saved dev predictions, accuracy improved from 89.6% to 90.8%. We ended up submitting predictions that scored 90.6% on the test set. By far, the biggest improvements were results of increasing the model size - all other experiments minimally improved the accuracy scores. From this, we determined that the number of parameters is the most influential aspect of the model on accuracy when working with paraphrase detection. Qualitatively, our run notes also indicate that the GPT-2 large LoRA configuration took only about three quarters of the runtime of comparable full fine-tuning, so it improved both score and iteration speed. If we had more compute, the next natural steps would be combining the final LoRA model with more thorough hyper-parameter sweeps (such as sweeping learning rate).

Sonnet generation. The clearest positive result on the sonnet task was the jump from the unconditional full fine-tuning baseline (42.34 chrF) to conditioned training recipes, which suggests that our main bottleneck on only 131 training sonnets was objective mismatch rather than lack of another optimizer. DPO and GRPO both helped relative to the baseline, but the strongest simple modeling improvement was curriculum-style SFT that first learned broader Shakespearean style and then specialized to exact three-line continuation, reaching 43.37 chrF. Two negative results were also very informative. The first was that a higher rank LoRA that updated 53.1M parameters (6.4% of GPT-2 large) still lagged full fine-tuning at 42.18 chrF, which suggests that LORA updates were too restrictive for long, structured continuation. The second was that more aggressive DPO negatives (reverse, prompt_repeat, splice) lowered exported dev chrF from 43.22 to 43.09, likely because they were too easy and encouraged the model to reject obviously broken continuations rather than improve subtle ones.

GRPO helped most as a polishing step on top of a strong supervised model rather than as a substitute for one. Baseline+GRPO beat the unconditional baseline by about 0.81 chrF, but the best final result came only after combining curriculum-trained checkpoints, structure-aware rewards, and reranked

decoding. Figure 2 shows why decoding became part of the model rather than an afterthought. The final candidate models varied from 42.76 to 43.37 chrF across generation seeds and candidate counts, so “best dev checkpoint” during training did not always produce the best exported file. Qualitatively, the unconditional baseline often remains locally fluent but drifts into generic romantic language after a few lines, whereas the stronger curriculum and GRPO systems preserve the elegiac tone and 14-line structure more reliably, even though they still hallucinate rather than reconstruct Shakespeare exactly. Taken together, these results indicate that on this benchmark, better task alignment and lower-variance decoding mattered more than simply adding more training machinery.

7 Conclusion

Across both tasks, the effective extensions greatly improved performance above the baselines. On paraphrase detection, the best saved improvement came from combining a larger GPT-2 backbone with LoRA, better prompt design, and seed tuning raising dev accuracy to 90.8%. On sonnet generation, the strongest gains came from conditioned completion training, curriculum-style prompt augmentation, and a final structure-aware GRPO plus reranked best-of- N decoding pipeline, reaching 43.37 dev chrF.

The main limitations of our project were compute and experiment tracking. We implemented more extensions than we could fully sweep. If we had more time and compute, we would complete the paraphrase GRPO, test lower-rank and selectively unfrozen LoRA variants on sonnets, build richer structure rewards using phonetic rhyme or syllable constraints, and conduct more thorough hyper-parameter sweeps.

Team Contributions

Koa Chang focused primarily on the sonnet-generation extensions, including conditioned SFT, curriculum training, DPO/GRPO experiments, reranking, run orchestration, and result analysis. Brandon Kunitzer focused primarily on paraphrase-detection extensions, including LoRA integration, prompt engineering, and downstream evaluation. Both authors implemented and validated the core GPT-2 project code, discussed experiment design, interpreted the results, and co-wrote the report.

References

- [1] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 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. *arXiv preprint arXiv:2305.18290*, 2024.
- [4] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [6] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th International Conference on Machine Learning*, pages 41–48, 2009.
- [7] Zhiguo Wang, Wael Hamza, and Radu Florian. Bilateral multi-perspective matching for natural language sentences. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 4144–4150, 2017.
- [8] Maja Popović. chrF: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, 2015.