

Direct Preference Optimization for Constrained Generation and Classification in GPT-2

Stanford CS224N Default Project

Weining Li

Department of Computer Science
Stanford University
w422li@stanford.edu

Jingxiong Zhao

Department of Computer Science
Stanford University
jxzhao@stanford.edu

Abstract

This project optimizes a GPT-2 model for two distinct downstream tasks: paraphrase detection and Shakespearean sonnet generation. To overcome computation cost and GPU resource limitations, we used Low-Rank Adaptation (LoRA) and Automatic Mixed Precision (FP16) training, which greatly saved memory and training time. For sonnet generation, standard fine-tuning often produces bad formats and repeating words. To fix this, we used a repetition penalty and a hard rule to force the model to output exactly 14 lines.

Importantly, because the original sonnet dataset is small, we used state-of-the-art (SOTA) models to generate positive synthetic data for our Supervised Fine-Tuning (SFT) round, which successfully improved our model's performance. Later, we also explored Beam Search and Direct Preference Optimization (DPO) to further improve the Sonnet quality. However, while our SFT data and 14-line rules worked very well, DPO and Beam Search did not improve the final scores. We analyze these results and think that our DPO setup could be limited by the quality of the negative data pairs and evaluation approach.

1 Key Information to include

- Mentor: Shicheng Liu
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any):

Team member	Late days remaining
Weining Li	5
Jingxiong Zhao	4
Total pooled	9
Team size	2
Late days applied for the report	3

2 Introduction

Large language models like GPT-2 possess strong general text capabilities, but adapting them to highly constrained tasks—such as strict binary classification (paraphrase detection) and strictly formatted creative writing (sonnet generation)—exposes the limitations of standard training paradigms.

In our early experiments, standard Supervised Fine-Tuning (SFT) presented several challenges. For sonnet generation, the model frequently terminated sequences prematurely and suffered from

repetitive phrasing. Furthermore, our project was bound by GCP budget and a single L40 GPU, making full-parameter fine-tuning computationally difficult due to the long training time and high cost. To address these hardware limitations, we integrated parameter-efficient techniques, including LoRA, FP16 mixed precision, and a cosine learning rate scheduler, which allowed us to train our models smoothly and efficiently.

To address the formatting issues in sonnet generation, we moved beyond standard nucleus sampling by implementing a repetition penalty and hard structural constraints to enforce the 14-line rule. Another major milestone was the introduction of synthetic data augmentation. By utilizing SOTA models to generate high-quality synthetic sonnets, we enriched our SFT dataset, which noticeably improved the model performance.

We also explored pushing the boundaries of text generation by integrating Beam Search and Direct Preference Optimization (DPO). In this report, we detail our system architecture for both tasks. We also present a critical analysis of our empirical findings, specifically examining why our SFT synthetic data and heuristic constraints succeeded, while advanced alignment techniques like DPO and Beam Search struggled in the highly subjective domain of sonnet generation.

3 Related Work

- **Efficient Fine-Tuning and Optimization:** Full-parameter fine-tuning for models like GPT-2 [1] is heavily memory-constrained. Micikevicius et al. (2018) demonstrated that Mixed Precision Training [2] accelerates computation without sacrificing numerical stability. To further reduce memory footprint, Hu et al. (2022) introduced Low-Rank Adaptation (LoRA) [3], which injects trainable rank-decomposition matrices while freezing pre-trained weights. Beyond memory efficiency, training stability is crucial. We utilized decoupled weight decay and cosine learning rate schedules [4], which have become the industry standard for smoothly converging Transformer models and preventing catastrophic forgetting during early training steps.
- **Decoding Strategies and Controllable Generation:** Text generation quality relies heavily on the decoding algorithm. Deterministic approaches like Beam Search tend to produce safe, highly probable, but often generic text, which is suboptimal for creative poetry. Holtzman et al. (2019) highlighted this issue and popularized Nucleus (Top- p) sampling [5] to maintain stochastic diversity. However, smaller models still frequently fall into degenerate repetitive loops. To explicitly mitigate this, researchers introduced dynamic logit modifications, such as the repetition penalty used in the CTRL model [6]. We adapted these penalization concepts alongside hard structural masking to enforce sonnet constraints.
- **Synthetic Data Augmentation:** A major bottleneck in specialized domain adaptation (like sonnets) is data scarcity. Advancements have shown that smaller models can effectively mimic the capabilities of much larger models using synthetic data. Frameworks like Self-Instruct [7] demonstrated that prompting State-of-the-Art (SOTA) models to generate high-quality instructional or formatted data is a highly effective augmentation strategy. We drew inspiration from this paradigm to expand our original Shakespearean corpus, allowing our model to learn structural priors more robustly during the SFT phase.
- **Preference Alignment:** Aligning language models with specific human intents or aesthetics was traditionally dominated by Reinforcement Learning from Human Feedback (RLHF), popularized by InstructGPT [8]. However, RLHF requires training a separate reward model and managing unstable PPO algorithms. Rafailov et al. (2023) proposed Direct Preference Optimization (DPO) [9], which elegantly bypasses reward modeling by reparameterizing the objective directly into a classification loss. We extend this line of work by applying DPO to strict poetic generation, exploring how preference learning interacts with highly constrained, creative formatted sonnets.

4 Approach

4.1 Shared Approaches

- **Baseline (GPT-2):** We use the provided GPT-2 [1] architecture as our baseline model.

- **Mixed Precision Training (FP16):** To reduce the extremely long training times with limited GPU resource, we implemented FP16 mixed-precision training [2] using PyTorch’s native AMP module . It reduces GPU memory usage and speeds up backpropagation while maintaining numerical stability.
- **LoRA:** To train a larger GPT2 model within our computational limits, we implemented LoRA [3] using Hugging Face’s PEFT code [10]. We injected the attention layer matrices (‘query’, ‘key’, ‘value’) as the baseline and explored different config for rank size and injection of additional attention and feed-forward matrices (‘attention_dense’, ‘interm_dense’, ‘out_dense’). By freezing the pre-trained model weights and injecting trainable rank decomposition matrices into the Transformer layers, we can efficiently fine-tune GPT-2 Medium with a fraction of the original parameters.

4.2 Paraphrase Detection

- **Customized Learning Rate Schedules:** Since Automatic Mixed Precision (FP16) is prone to numerical instability, we replaced static learning rates with a cosine scheduler featuring a linear warmup. The warmup phase gradually increases the learning rate, preventing destructive gradient updates to pre-trained weights early on. The subsequent cosine decay ensures smooth and stable convergence during the final epochs.

4.3 Sonnet Generation

- **Synthetic Data Augmentation:** The original Shakespeare sonnet training set contains only 131 examples, which is insufficient for reliably adapting GPT-2 to a highly constrained poetic generation task. To address that, we used knowledge distillation[11] to expand the supervised fine-tuning corpus with 3,000 synthetic sonnets generated by a stronger teacher model (GPT-5.3), resulting in a total training set of 3,131 sonnets. Our goal was to expose the model to substantially more sonnet-shaped text so that it could better internalize structural priors such as line-level formatting, poetic diction, and overall sonnet organization.
- **Nucleus Sampling (Top- p):** As our primary decoding strategy, we adopted nucleus sampling. At each generation step, we sort the vocabulary by next-token probability and retain only the smallest prefix of tokens whose cumulative probability exceeds a threshold p , then sample the next token from this truncated distribution. We tuned both the top- p threshold and the sampling temperature on the development set to balance fluency, diversity, and structural stability.
- **Beam Search:** We also evaluated Beam Search as a stronger deterministic decoding baseline. Beam Search maintains the top k partial hypotheses at each decoding step and expands them according to cumulative sequence likelihood, returning the highest-scoring completed sequence at the end of generation. We used beam width $k = 3$ in our experiments. The motivation for including Beam Search was to test whether more aggressive likelihood maximization could improve the fluency and global consistency of generated sonnets. Beam Search may improve local probability optimization, but it can also reduce diversity by repeatedly favoring safe, high-probability continuations.
- **Early Stopping:** To prevent overfitting and select the most effective checkpoint during training, we applied early stopping based on dev-set result. After each training epoch, we generated sonnets on the development set and computed the corresponding chrF score as our model selection criterion. We introduced a patience threshold so that training would terminate when the dev chrF failed to improve for several consecutive epochs. In addition, we saved a checkpoint at the end of every epoch and retained the checkpoint with the best dev chrF as the final model used for evaluation.
- **Repetition Penalty:** To prevent the repetitive loops common in smaller GPT models, we implemented a dynamic repetition penalty [6]. During generation, we track all previously generated token IDs. For the next-token prediction, we dynamically penalize the logits of these historical tokens: dividing by 1.2 if positive, or multiplying by 1.2 if negative. This mathematically reduces their softmax probabilities, strongly enforcing vocabulary diversity.
- **DPO:** We tried to apply DPO to align outputs with sonnet aesthetics, bypassing the complexity of RLHF reward modeling. We maintained both a frozen Reference Model (our best SFT checkpoint) and a trainable Policy Model. We optimized the standard DPO loss

($\beta = 0.1$) using SOTA-generated synthetic sonnets as "chosen" targets and a GPT-2 SFT outputs with high temperature (1.2) as "rejected" targets.

5 Experiments

5.1 Data

We utilize the datasets provided in the default project repository, supplemented by an augmented dataset for our extension tasks:

- Stanford Sentiment Treebank (SST): Standard benchmark for sentiment analysis.
- CFIMDB Dataset: A large-scale movie review dataset for binary sentiment classification.
- Quora Dataset: Used for identifying duplicate questions and semantic similarity.
- Shakespeare Sonnet Dataset: The original training set contains only 131 data points, which is insufficient for fine-tuning a large-scale GPT-2 model effectively. To address this, we used knowledge distillation from a larger teacher model(GPT-5.3) to increase the training set to 3,131 sonnets.
- Pairwise Preference Data for DPO: We constructed 1000 "preferred" vs. "rejected" pairs to support DPO, the preferred responses are from Sonnets generated by GPT 5.3, and for on-policy alignment, we used our SFT checkpoint of GPT2 with slightly higher temperature to generate rejected responses.

5.2 Evaluation Metrics

We evaluate our model quantitatively across all three tasks. For the classification tasks—Sentiment Analysis and Paraphrase Detection—we employ **Accuracy** as the primary metric. To assess the Sonnet Generation task, we utilize the **chrF score**. We chose chrF over word-level metrics because its character n-gram approach better captures the archaic morphology and stylistic nuances characteristic of Shakespearean English. .

5.3 Experimental details

All experiments were conducted on Google Cloud using a single NVIDIA L40 GPU. Because full-parameter fine-tuning of GPT-2 was computationally expensive under our budget, we adopted parameter-efficient fine-tuning throughout our extension experiments. In particular, we used LoRA to reduce the number of trainable parameters and FP16 automatic mixed precision to lower memory usage and accelerate training. Unless otherwise noted, optimization was performed with AdamW and a cosine learning rate scheduler. For each experiment, we tuned the batch size, learning rate, and number of epochs on the development set, and selected the checkpoint with the best dev performance.

Sentiment classification

For sentiment classification, we follow the default project setup and fine-tune GPT2-small on SST and CFIMDB. Since this task serves primarily as a baseline classification setting, we use the default configuration provided in the project codebase. Specifically, we use a batch size of 64 and set the hidden dropout probability to 0.3. We evaluate two training regimes: updating only the final linear classification layer, and full-model fine-tuning. For both settings, model selection is based on development accuracy.

Paraphrase Detection

For the paraphrase detection task, with LoRA and FP16 mentioned above, we were able to train the GPT-2 models with a batch size of 32. To maximize semantic comprehension capabilities within our parameter-efficient setup, we aggressively expanded the LoRA target modules beyond the standard attention projections. Specifically, adapters were injected into both the attention mechanisms (query, key, value, attention_dense) and the deeper feed-forward network layers (interm_dense, out_dense), configured with rank $r = 32$, scaling factor $\alpha = 64$, and a dropout rate of 0.1. Furthermore, to stabilize the fine-tuning process and prevent catastrophic forgetting of pre-trained weights, we implemented a cosine learning rate scheduler with a linear warmup phase. The warmup spanned the

first 10% of the total training steps, gradually scaling the learning rate to its peak before smoothly decaying it along a cosine curve.

Sonnet Generation

For sonnet generation, we first performed supervised fine-tuning on the Shakespeare sonnet corpus augmented with GPT-5.3-generated synthetic data, increasing the effective training set to 3,131 sonnets. We then ran a second-stage preference optimization experiment using DPO on 1,000 winning-losing pairs constructed from the sonnet data and automatically generated negative samples. The SFT stage used batch size 8, learning rate 5e-5, and 10 epochs, while the DPO stage used batch size 8, learning rate 1e-5, and 10 epochs. For decoding, our primary setting used nucleus sampling with temperature 0.5, top-p 0.9, and a repetition penalty to reduce degenerative repetition. Because sonnet generation requires strict structure, we additionally enforced a hard constraint that each output contain exactly 14 non-empty lines; generations that terminated too early or exceeded the line budget were truncated or regenerated. We also compared this decoding strategy against beam search with beam width 3 under the same token budget. Model selection for sonnet generation was based on development chrF, and we also tracked qualitative failure modes such as repetition and line-count violations.

5.4 Results

Sentiment Classification

Fine-tune Mode	SST Dev Accuracy	CFIMDB Dev Accuracy
Last Linear Layer	0.457	0.865
Full Model	0.507	0.984

Table 1: Sentiment Classification Accuracy

Paraphrase Detection

Model Size	Approaches	Training Time / Epoch	Dev Acc	Test Acc
GPT2-small	Baseline GPT-2	50 min	0.897	n/a
GPT2-small	FP16	16 min	0.897	n/a
GPT2-medium	FP16	48 min	n/a	n/a
GPT2-medium	FP16 + LoRA	27 min	0.896	0.880
GPT2-medium	FP16 + tuned LoRA & lr	29 min	0.907	0.898

Table 2: Paraphrase Detection Training Time & Accuracy

Sonnet Generation

For all sonnet experiments, we use LoRA and FP16 mixed precision by default, since both techniques were validated in earlier development as effective for reducing memory cost while maintaining strong performance. We compare decoding strategies, model sizes, data augmentation, and preference optimization under this shared setup.

Model	Approach	Dev chrF Score	Test chrF score
GPT2-medium	Beam Search ($k = 3$)	39.584	n/a
GPT2-medium	Top- p	40.032	n/a
GPT2-medium	Top- p + Synthetic Data (+3000)	42.744	n/a
GPT2-large	Beam Search ($k = 3$) + Synthetic Data (+3000)	41.165	n/a
GPT2-large	Top- p + Synthetic Data (+3000)	43.231	42.728
GPT2-large	DPO + Top- p + Synthetic Data (+3000)	42.175	n/a

Table 3: Sonnet generation chrF scores with different settings. All experiments use LoRA and FP16 by default.

6 Analysis

Beyond the aggregate automatic metrics, we conducted a qualitative analysis to better understand which design choices improved paraphrase detection and sonnet generation and which introduced new

failure modes. Our main takeaway is that, parameter-efficient fine-tuning (LoRA) and mixed precision (FP16) not only solved our computational bottlenecks but also acted as implicit regularizers to prevent overfitting. In addition, in this low-resource and highly structured setting, simple task-specific interventions were more effective than more sophisticated decoding and preference-optimization methods. In particular, synthetic data augmentation, repetition control, and explicit enforcement of sonnet structure yielded more reliable improvements than Beam Search or DPO.

6.1 Efficiency and Performance Gains

For the paraphrase detection task, our primary success was overcoming severe computational bottlenecks while achieving strong classification performance. The integration of Automatic Mixed Precision (FP16) and Low-Rank Adaptation (LoRA) proved highly effective and essential for our constrained environment.

First, FP16 training significantly accelerated computation and reduced the memory footprint. This optimization allowed us to utilize a larger batch size of 32 and train the model much faster and smoothly without encountering Out-Of-Memory (OOM) errors on a single L40 GPU.

Second, applying LoRA comprehensively across both the attention mechanisms and the feed-forward network layers allowed the model to learn deep semantic equivalences using only a tiny fraction of the total parameters. Interestingly, we observed that this parameter-efficient strategy acted as an implicit regularizer. By freezing the base weights, LoRA prevented catastrophic forgetting and preserved the robust linguistic priors of the pre-trained GPT-2 model. Rather than overfitting to the syntactic quirks of the Quora dataset, the model successfully leveraged its pre-trained knowledge to generalize and identify true semantic paraphrases.

Ultimately, our results demonstrate that dense LoRA injections, when stabilized by a cosine learning rate scheduler, can match the representational power of full fine-tuning under computation resource limitations.

6.2 Beam search reduced quality and increased repetition

One of the most interesting findings during development was that Beam Search did not improve generation quality. In fact, Beam Search with beam size 3 underperformed our top- p sampling baseline, achieving a chrF score of 41.165 compared to 43.231 for top- p decoding. Although Beam Search is often effective for tasks where the objective is closely aligned with likelihood maximization, our results suggest that this assumption does not hold as well for creative generation tasks such as poetry.

Qualitatively, Beam Search frequently produced outputs with repetitive phrasing, semantically redundant lines, and overly conservative continuation patterns. In many cases, the model reused similar sentence structures across multiple lines, which made the resulting sonnets feel rigid and unnatural. This behavior is consistent with a known tendency of beam-based decoding to overfavor high-probability continuations, even when those continuations reduce diversity and stylistic richness. For sonnet generation, where lexical variety and figurative language are important aspects of quality, this tendency appears to be harmful rather than beneficial.

By contrast, top- p sampling produced outputs that were more varied and less repetitive while still remaining stylistically coherent. This suggests that, for structured creative generation, a controlled stochastic decoding strategy is better suited than aggressive likelihood maximization. In our setting, decoding diversity was not merely a stylistic preference; it was directly related to better automatic performance and more natural qualitative outputs.

6.3 DPO did not outperform the SFT baseline

Despite the theoretical advantages of Direct Preference Optimization (DPO), our empirical results showed that DPO did not improve the CHRF score compared to our best SFT baseline. We attribute this to three fundamental challenges in applying preference alignment to creative, strictly formatted domains.

First, the data quality of synthetic preference pairs. DPO is notoriously sensitive to the contrastive margin between the chosen and rejected responses. Because a large-scale, human-annotated preference dataset for sonnets does not exist, we relied on intermediate, flawed SFT outputs as our rejected responses. If the rejected data is simply disjointed text rather than "structurally correct but stylistically poor" sonnet because of the high temperature we used here, the model learns a trivial decision boundary. It fails to capture the nuanced differences in poetic aesthetics, limiting the effectiveness of the optimization objective.

In addition, aligning language models often incurs an "alignment tax", a degradation in base generation capabilities. In sonnet poetry, the model must maintain delicate, long-range token dependencies to preserve meter and complex rhyme schemes. By aggressively shifting the log-probability distributions to maximize the DPO margin, the model disrupts these fragile dependencies learned during the SFT phase. This results in outputs that occasionally lose their coherent narrative flow or break rhyming structures, demonstrating a trade-off between preference alignment and sequence stability.

There could also be a metric misalignment. The CHRF metric evaluates sequence quality by measuring character n-gram overlap. However, sonnet generation is inherently open-ended. A model can produce a stylistically beautiful, structurally perfect sonnet that shares very few n-grams with the specific reference dataset. Consequently, DPO might successfully align the model towards our "preferred" synthetic style, but this creative divergence is heavily penalized by exact-match metrics like CHRF. CHRF could potentially fail to evaluate human preference and structural rhythm in some cases.

There are other potential factors that could affect DPO's performance, one is the hyper-parameters used are not optimized, as DPO may require a very different config on learning rate, temperature, etc. We tried some setups, but were not able to thoroughly run experiments against all combinations.

6.4 What worked best: synthetic SFT data and hard constraints

In contrast to Beam Search and DPO, the interventions that produced the most consistent gains were the simplest ones: synthetic supervised data augmentation and explicit structural control. The original sonnet dataset contains only 131 training examples, which is extremely limited for teaching a language model both poetic style and rigid sonnet form. Expanding the dataset to 3,131 sonnets provided substantially more task-specific supervision and exposed the model to a broader range of sonnet-like phrasing and structure during fine-tuning.

Similarly, enforcing the requirement that each generated sonnet contain exactly 14 non-empty lines directly addressed one of the most common failure modes of the base model. Rather than relying on the model to implicitly learn this constraint from a small dataset, we incorporated the structure explicitly at decoding time. This intervention was highly effective because it was tightly matched to the task definition itself.

Taken together, these findings suggest that in low-resource generation tasks with strong formal constraints, improvements are more likely to come from better supervision and explicit task-aware controls than from more advanced optimization techniques. For our project, the main bottleneck was not the absence of a more sophisticated learning objective, but the difficulty of getting a small model to consistently satisfy both stylistic and structural requirements. Methods that directly addressed those requirements therefore delivered the largest practical gains.

7 Conclusion

This project shows that GPT-2 can be adapted to both paraphrase detection and sonnet generation under limited compute by using parameter-efficient methods. LoRA and FP16 were especially important, reducing training cost while preserving strong performance on a single L40 GPU. For paraphrase detection, our best configuration, GPT-2 medium with FP16, LoRA, and a tuned learning rate schedule, reached the best dev accuracy. For sonnet generation, the best setup combined GPT-2 large, top-p sampling, and synthetic data augmentation.

More broadly, our results suggest that in low-resource, highly structured generation tasks, simple task-specific interventions can outperform more complex alignment methods. Synthetic SFT data, repetition control, and explicit 14-line constraints consistently improved sonnet quality, while beam search reduced diversity and DPO did not beat the strongest SFT baseline. These findings suggest that better supervision and stronger structural control mattered more than added optimization complexity.

Team contributions (Required for multi-person team)

- Jingxiong has worked on the Sonnet generation baseline setup, also synthetic data augmentation, Nucleus filtering, Beam search, AdamW optimizer, early stopping and Sonnet generate structure controlling logic.
- Weining has worked on the base GPT2 model and modules, the Paraphrase detection baseline setup, also LoRA, Mixed Precision, customized learning rate schedules, DPO and rejection data for alignment, repetition penalty.

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] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Igor Babuschkin, Gideon Ginsburg, Mark Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *ICLR*, 2018.
- [3] 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. In *ICLR*, 2022.
- [4] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.
- [5] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *ICLR*, 2020.
- [6] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*, 2019.
- [7] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *ACL*, 2023.
- [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 *NeurIPS*, 2022.
- [9] 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 *NeurIPS*, 2023.
- [10] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. Peft: State-of-the-art parameter-efficient fine-tuning methods. <https://github.com/huggingface/peft>, 2022.
- [11] Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024.