

# Accelerated DPO Finetuning GPT-2 with Constructed Data

Stanford CS224N Default Project

**Jessie Ou**

Department of Computer Science  
Stanford University  
jessieou@stanford.edu

**Weixin Yu**

Department of Computer Science  
Stanford University  
weixinyu@stanford.edu

## Abstract

We implement the GPT-2 decoder architecture and fine-tune the model for two downstream tasks: cloze-style paraphrase detection and Shakespearean sonnet generation. To better align generated sonnets with desirable stylistic and structural properties of sonnets, we apply Direct Preference Optimization (DPO) using synthetically constructed preference pairs. To improve training efficiency, we incorporate FlashAttention into the attention module and evaluate Low-Rank Adaptation (LoRA) for parameter-efficient fine-tuning. We further examine task-aware decoding strategies for sonnet generation, including nucleus sampling, beam search, random sampling, and structure-enforcing generation constraints. Our experiments show that the effectiveness of these techniques is highly task-dependent. FlashAttention consistently accelerates training without affecting model quality, while LoRA improves paraphrase detection but degrades performance for creative generation tasks such as sonnet writing. DPO with carefully constructed diverse negative examples yields modest improvements in sonnet generation, though gains remain limited due to the mismatch between DPO objective and chrF evaluation metric. Overall, our results demonstrate that the effectiveness of modern fine-tuning techniques is highly task-dependent, and that careful consideration of task structure, evaluation metrics, and data quality is essential for achieving strong performance across diverse NLP tasks.

## 1 Key Information

**Mentor:** Ahmed Ahmed; **External Collaborators:** No; **Sharing project:** No

## 2 Introduction

Large language models (LLMs) based on the transformer architecture [1] have become the dominant paradigm for a wide range of natural language processing tasks. In this project, we build on the GPT-2 architecture and evaluate it on three tasks: sentiment analysis, paraphrase detection, and sonnet generation. This project investigates three questions: whether accelerated attention mechanisms can improve wall-clock training efficiency without sacrificing performance; whether parameter-efficient fine-tuning with LoRA can match or even outperform full model fine tuning across different tasks; and whether DPO fine-tuning with automatically constructed preference data can improve sonnet generation beyond standard cross-entropy supervised tuning.

We experimented most intensively with our approaches on the sonnet generation task. Sonnet generation is a particularly challenging task for language models. Unlike discriminative tasks such as sentiment analysis and paraphrase detection, which require the model to learn a compact decision boundary, sonnet generation requires the model to create structurally constrained text while ensuring the semantic coherence and authentic style.

Another challenge arises from the limited size of the dataset: the available corpus contains only 154 Shakespearean sonnets. Training solely on positive examples makes it difficult for the model

to learn what distinguishes a good sonnet from an invalid or poorly structured one. To address this limitation, we explore Direct Preference Optimization (DPO) [2], which learns from pairwise preference data rather than only positive examples. Because human preference annotations are unavailable, we construct synthetic negative examples by corrupting real sonnets, enabling the model to learn preferences between valid and corrupted outputs.

We make three main contributions. First, we implement FlashAttention [3] as a drop-in replacement for the standard masked self-attention used in GPT-2, achieving up to a  $1.26\times$  training speedup for GPT-2 Medium. Second, we evaluate LoRA [4] across tasks and find that while it slightly improves performance on paraphrase detection, it significantly degrades sonnet generation, suggesting that low-rank adaptation may be less suitable for creative generation tasks. Third, we investigate the use of structural constraints and DPO fine-tuning for sonnet generation. We observe that structural constraints slightly improve performance ( $42.111 \rightarrow 42.471$  chrF), and DPO fine-tuning also improves performance ( $41.596 \rightarrow 42.349$  chrF). Our results suggest that the effectiveness of modern fine-tuning and alignment methods is highly task-dependent, and that high-quality preference data is crucial for DPO in low-resource creative generation settings.

### 3 Related Work

**Reinforcement Learning from Human Feedback and DPO.** Pretrained language models by itself, using encoder/decoder architectures, are not very aligned with user intent. The performance can be significantly improved by supervised fine-tuning, including PPO [5], using examples paired with human-labeled scores. To avoid the high cost and inaccuracy in obtaining human-labeled scores, RLHF methods fit a reward model with pairwise comparison data, using models like Bradley-Terry [6] [7]. However, this is costly due to training multiple LMs and sampling in-the-loop. In addition to high computational cost, PPO-based RLHF introduces training instability and sensitivity to KL regularization, and decouples reward modeling from policy optimization, potentially leading to objective mismatch. Direct preference optimization (DPO) streamlined the process of training the separate reward model while optimizing with the same binary cross entropy objective.

**Synthetic Preference Data.** A key bottleneck in applying DPO outside of large-scale human annotation settings is the construction of preference data when human annotations are unavailable. Prior work has explored both model-generated negatives [2] and rule-based corruptions for tasks such as summarization and translation. We took inspiration from data augmentation strategies in contrastive learning [8] to construct synthetic negatives. These synthetic corruptions aim to approximate common failure modes in generation while avoiding the need for manual annotation.

**Parameter-Efficient Fine-tuning.** Full fine-tuning in large language models can be expensive. Low-Rank Adaptation (LoRA) [4] addresses this by constraining weight updates to a low-rank subspace. LoRA has been shown to be effective and sometimes prevent data overfitting, and our work contributes by comparing LoRA against full fine-tuning on tasks of varying creativity.

**Efficient Attention Mechanisms.** The scaled dot-product attention mechanism [1] has quadratic time and memory complexity with respect to sequence length. Flash Attention [3] addresses this by using tiling and CUDA kernel fusion to compute exact attention in chunks, achieving wall-clock speedups without approximation. In contrast, Longformer [9] replaces full self-attention with a sliding window pattern, reducing complexity from quadratic to linear at the cost of expressiveness. We experimented the attention mechanism with a relatively short sequence length, and observed consistent speedups.

## 4 Approach

### 4.1 Baselines

We implemented the GPT-2 architecture based on the provided starter code [10], completing the multi-head masked self-attention mechanism and the remaining components of the decoder stack. For sentiment classification, used a linear and dropout layer on the hidden representation of the last tokens. For paraphrase detection, we used a simple linear layer to produce. For sonnet generation, we projected the hidden states back to the vocabulary space using the shared embedding weight matrix and generated text autoregressively using top-p decoding.

## 4.2 Direct Preference Optimization (DPO) [2]

We finetuned GPT-2 using the DPO objective following a standard cross-entropy task finetuning. We constructed the pairwise data and implemented the DPO loss function by ourselves, where the DPO loss function is given by:

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

For paraphrase detection, we generated negative samples by label-flipping the correct yes/no answer for each prompt in the Quora dataset. For sonnet generation, we employed both rule-based corruptions and randomized generation. For rule-based corruptions, we corrupted real sonnets using the following transformations to target structural and stylistic properties of sonnets: (1) rhyme violation, where selected line-final words are replaced with non-rhyming alternatives; (2) line-order permutation to disrupt discourse structure; (3) iambic meter violations via token removal/addition; and (4) intra-line token shuffling to break syntax. Finally, we combined these individual corruptions to generate negative examples with multiple corruptions.

For randomized generation, we used the baseline model to generate negative examples from the test dataset. We then evaluated these generations using chrF scores against the ground-truth sonnets and constructed positive–negative preference pairs between the ground-truth and each generated sample. In addition, we included pairs among generated samples whose chrF scores differed substantially, treating the higher-scoring generation as the preferred example.

## 4.3 Flash Attention [3]

To improve training efficiency, we integrated FlashAttention into the attention layer of our GPT-2 implementation. In our implementation, we replaced the original attention computation with the fused attention primitive provided by PyTorch, which internally uses FlashAttention kernels when supported by the hardware. This allows PyTorch to automatically dispatch to optimized FlashAttention kernels on compatible GPUs.

## 4.4 Low-rank Adaptation of Large Language Models (LoRA) [4]

To enable parameter-efficient fine-tuning, we incorporated LoRA into our GPT-2 implementation. We implemented LoRA ourselves and applied it only to the linear projection layers within the multi-head attention module while leaving the rest of the GPT-2 architecture unchanged.

We applied the following LoRA adaptation to the query, key, and value matrix in each attention layer.

$$W' = W + \frac{\alpha}{r} BA$$

where  $W$  is the original weight matrix,  $r$  is rank, and  $\alpha$  is the LoRA scaling factor,  $A \in \mathbb{R}^{r \times d}$  is the down-projection matrix, and  $B \in \mathbb{R}^{k \times r}$  is the up-projection matrix.

During fine-tuning, the original pretrained weights remain frozen, and only the LoRA matrices are updated. This significantly reduces the number of trainable parameters and lowers the computational cost of adaptation.

## 4.5 Hyperparameter Tuning

We performed targeted hyperparameter tuning to identify stable and effective settings for both supervised fine-tuning and preference optimization. Our main tuning parameters were the learning rate  $\eta$ , the  $\beta$  parameter used in DPO, the number of training epochs, and the GPT-2 model size.

## 4.6 Task-aware Decoding Techniques (Sonnet Generation)

During the random generation of DPO negative examples using the baseline model, we observed that the sentence-level chrF score varies non-trivially across different samples. This suggests that, even with the same model, selecting the “best” generated sonnet from a set of candidates could improve the chrF score. In addition, we noticed that many generated sonnets does not follow the strict structure of sonnets. Motivated by these observations, we explored several task-aware decoding strategies to improve generation quality:

- Beam Search: Generate multiple candidate sequences while maintaining the most probable ones according to the model’s likelihood.
- Random Generation with Selection: Generate multiple candidates randomly and select the one with the highest accumulated conditional probability  $p(y \mid x)$ , as estimated by the dpo-fine-tuned model.
- Structure-Enforcing Generation: Incorporate rule-based constraints to enforce sonnet-specific structural properties, such as writing exactly 14 lines, forcing a newline after prompt, and last line must have 5 words.

## 5 Experiments

### 5.1 Data

The Sentiment Analysis task will use 2 datasets: 1) Stanford Sentiment Treebank (SST) dataset (11,855 sentences from movie reviews), and 2) CFIMDB dataset (2434 highly polar movie reviews). The Paraphrase Detection task will use the Quora dataset, consisting of 400,000 question pairs with labels. The sonnet generation task will use the Sonnet Dataset, consisting of consists of 154 sonnets written by Shakespeare.

### 5.2 Evaluation Method

For Sentiment Analysis and Paraphrase Detection, we evaluated performance using the mean accuracy on class labels. For sonnet generation, we assessed performance using the chrF metric (character-level n-gram F-score) by conditioning on the first three lines of each held-out sonnet and generating the remaining lines, then comparing our generated continuation against the reference continuation. In addition, we reported wall-clock training time to quantify the benefits of accelerated attention.

We did all experiments on an NVIDIA L4 GPU, with machine type g2-standard-4 (4 vCPUs, 16 GB Memory), and 100GB boot disk storage.

### 5.3 Sentiment Analysis

The baseline accuracy of our GPT-2 model with an extra linear and dropout layer on the sentiment analysis task achieved the accuracy as shown in Table 1. We uses 30 training epochs, a batch size of 64 for SST and 8 for CFIMDB, and a learning rate of  $1 \times 10^{-4}$ , with other parameters set as default (hidden dropout probability is 0.3). It took about an hour for full model parameter fine-tuning for each dataset, and about 15 minutes for the last linear layer.

| Task       | Dataset | Model                                | Accuracy/chrF |
|------------|---------|--------------------------------------|---------------|
| Sentiment  | SST     | Last-Linear (Dev)                    | 43.69%        |
| Sentiment  | SST     | Last-Linear (Test)                   | 44.71%        |
| Sentiment  | SST     | Full Model (Dev)                     | 50.50%        |
| Sentiment  | SST     | Full Model (Test)                    | 54.48%        |
| Sentiment  | CFIMDB  | Last-Linear (Dev)                    | 86.94%        |
| Sentiment  | CFIMDB  | Full Model (Dev)                     | 97.96%        |
| Paraphrase | Quora   | GPT-2 with simple linear layer (Dev) | 89.6%         |
| Sonnet     | Sonnet  | GPT-2 with simple linear layer (Dev) | 41.962        |

Table 1: Baseline model accuracies.

### 5.4 Paraphrase Detection

The baseline accuracy of our GPT-2 model with a simple linear layer achieved dev accuracy of 0.896 as shown in Table 1. We uses 10 training epochs, a batch size of 32, a learning rate of  $1 \times 10^{-5}$  with a standard gpt2 model. The experiment takes around 5 hours on a GPU.

#### 5.4.1 LoRA

We tested LoRA with GPT-2 Large at  $lr = 1e - 5, r = 8, \alpha = 16$ , achieving dev accuracy of 90.0% after epoch 8, compared to a baseline of 89.6%. LoRA is effective for paraphrase detection, slightly outperforming the full fine-tuning baseline while using significantly fewer trainable parameters. This

is consistent with the problem – the paraphrase detection requires the model to learn a relatively simple decision boundary, which is probably sufficient with low-rank updates to the model. LoRA prevents the model from overfitting to the training data and encourages the model to align better with the pretrained model.

## 5.5 Sonnet Generation

The baseline accuracy of paraphrase detection achieved corpus level chrF of 41.962 as shown in Table 1. We use 10 training epochs, a batch size of 8, a learning rate of  $1 \times 10^{-5}$  with a standard GPT-2 model. The experiment takes less than 3 minutes on GPU.

### 5.5.1 Flash Attention

The training time for 10 epochs with GPT2 model of different sizes is reported below in Table 2. We used a sequence length of 128. For GPT-2 and GPT-2 Medium, Flash Attention reduces training time by  $1.18\times$  and  $1.26\times$  respectively, suggesting that medium-sized models benefit the most from the memory-efficient attention computation. However, for GPT-2 Large, the speedup is negligible ( $1.01\times$ ), suggesting that attention computation is no longer the main runtime bottleneck at larger model sizes. Other components such as the feed-forward layers and memory bandwidth probably dominate the runtime.

| Model        | Without Flash Attn (s) | With Flash Attn (s) | Speedup       |
|--------------|------------------------|---------------------|---------------|
| GPT-2        | 164                    | 139                 | $1.18 \times$ |
| GPT-2 Medium | 217                    | 172                 | $1.26 \times$ |
| GPT-2 Large  | 388                    | 384                 | $1.01 \times$ |

Table 2: Runtime comparison with and without Flash Attention over 10 epochs, with a learning rate of  $1e-5$ . Speedup is computed as  $t_{\text{without}}/t_{\text{with}}$ .

### 5.5.2 Hyperparameter Tuning

We tuned the learning rate, model size, and number of training epochs for GPT-2. As shown in figure 1, while the training loss decreases consistently, dev chrF peaks early in training (around epoch 5 for  $lr = 3 \times 10^{-5}$ ), indicating overfitting to the small sonnet dataset.

We also observe that larger models require smaller learning rates for stable training. In particular, GPT-2 Medium performs best around  $1 \times 10^{-5}$ , while smaller GPT-2 models tolerate higher learning rates such as  $7 \times 10^{-5}$  (figure 2). This likely reflects the greater sensitivity of larger models to optimization step sizes.

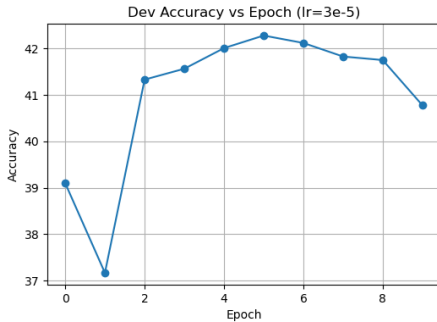


Figure 1: Dev accuracy trend with epoch for  $lr = 3e-5$  on the sonnet generation dataset, with standard gpt-2 model.

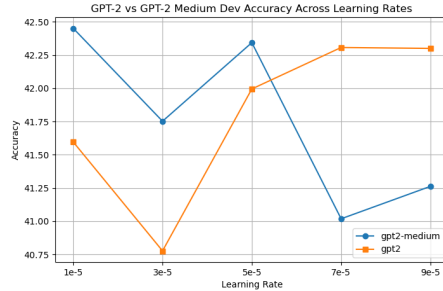


Figure 2: Dev accuracy trend with learning rate on the sonnet generation dataset, GPT-2 vs GPT-2 Medium.

### 5.5.3 LoRA

We tested LoRA with GPT-2 Large at  $lr = 1e-5$ ,  $r = 8$ ,  $\alpha = 16$ , achieving a corpus-level chrF score of 38.57, which is notably lower than the non-LoRA baseline of 41.96. Furthermore, DPO fine-tuning with LoRA further degraded the score to 36.53, while DPO without LoRA improved the baseline

to 42.31. This suggests that LoRA is detrimental to the sonnet generation task. By constraining the weight updates to a low-rank space, LoRA may be too restrictive for the sonnet generation task, which requires the model to learn fine-grained stylistic and structural patterns compared to ordinary texts. While LoRA is generally parameter-efficient and effective for many downstream tasks, it may not be suitable for creative generation tasks that require higher rank in fine-tuning.

#### 5.5.4 DPO

As the baseline of DPO, we experimented with model-generated data while sweeping the DPO-specific learning rate. This DPO fine-tuning is based on the supervised-fine-tuned gpt2 model with  $lr = 1e - 5$ , which has corpus-level chrF score of 41.596. The trend of dev accuracy with learning rate is shown in Figure 3. As compared to Figure 2, with the same gpt2 model, DPO achieve its peak accuracy with a much smaller learning rate. The best learning rate is  $3e - 6$ .

This is because DPO optimizes relative preferences, which tends to generate larger gradients. In addition, since DPO starts with an already fine-tuned model by supervised learning, only small adjustments are needed to align with preferences. Lastly, because our positive-negative pairs for DPO is automatically generated, the dataset tends to have more variance than the data used in supervised fine tuning.

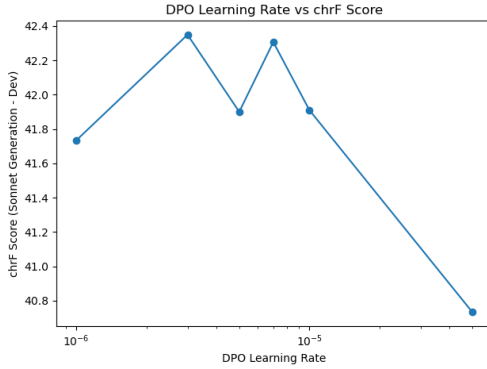


Figure 3: Dev accuracy trend with learning rate on the sonnet generation dataset, with standard gpt-2 model. Peak accuracy on dev dataset is reached at  $lr = 3e - 6$ .

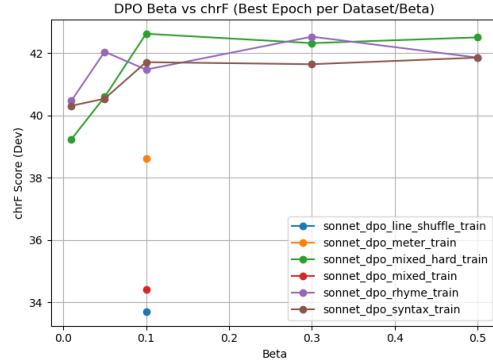


Figure 4: Dev accuracy trend with dpo  $\beta$ , with various dataset construction methods, taking the model with highest dev accuracy over 10 epochs. The experiments for meter, mixed, and line shuffle corruption are run only with the best  $\beta = 0.1$ .

#### 5.5.5 Pairwise Preference Data Construction for DPO

We experimented with the various rule-based datasets generated by our synthetic preference data construction, as described in Section 4.2.

Figure 4 depicts the effect of  $\beta$  on the corpus-level chrF scores. We report the chrF scores by taking the model that generated the highest chrF score over 10 epochs of DPO training, on a learning rate of  $1e - 5$ . As can be observed from the chart, the most effective data construction schemes are syntax and rhyme corruption, while a dataset with mixed corruptions leads to a slightly better result. The peak performance over this experiment is with the rule-based constructed dataset with 3+ mixed corruptions.

#### 5.5.6 Task-aware Decoding Techniques

**Beam Search.** We experimented with beam widths of 1, 3, 5, and 10, where a breadth of 1 degenerates to a greedy search scheme. The generated sonnets were repetitive and semantically weak, as beam search greedily favors high-probability tokens at the expense of diversity. An example is quoted below:

*That I may be comforted, and comforted, and comforted, and comforted,*

| Training Objective | Negative Construction                  | Model Size | # Epochs | LR   | $\beta$ | Held-out ChrF |
|--------------------|----------------------------------------|------------|----------|------|---------|---------------|
| Cross-entropy      | –                                      | GPT2       | 10       | 1e−5 | –       | 41.962        |
| Cross-entropy      | –                                      | GPT2-large | 10       | 1e−5 | –       | 42.111        |
| DPO                | Rule-based rhyme violation             | GPT2       | 10       | 1e−5 | 0.1     | 42.157        |
| DPO                | Rule-based meter corruption            | GPT2       | 10       | 1e−5 | 0.1     | 39.632        |
| DPO                | Rule-based line permutation            | GPT2       | 10       | 1e−5 | 0.1     | 33.686        |
| DPO                | Rule-based syntax corruption           | GPT2       | 10       | 1e−5 | 0.1     | 42.150        |
| DPO                | Rule-based mixed (2 corruptions)       | GPT2       | 10       | 1e−5 | 0.1     | 38.215        |
| DPO                | Rule-based mixed-hard (3+ corruptions) | GPT2       | 10       | 1e−5 | 0.1     | <b>42.644</b> |
| DPO                | Model-generated                        | GPT2       | 10       | 3e−6 | 0.1     | <b>42.349</b> |

Table 3: Held-out ChrF for cross-entropy vs. DPO models trained on different negative-construction strategies.

This is consistent with prior observations that beam search maximizes likelihood rather than generation quality, which makes it poorly suited for open-ended creative tasks such as poetry.

**Random Generation.** Following our observation from section 4.6, we generated multiple samples per prompt and selected the candidate with the highest model probability  $p(y|x)$ . However, this approach reduced chrF scores ( $\approx 38$  vs.  $\approx 42$  for standard top-p sampling). Selecting the most probable sample effectively behaves like beam search and tends to produce generic outputs with limited diversity.

**Structure-enforcing Generation.** We implemented with structure-enforcing generation, and experimented with both the baseline cross-entropy and DPO objective. The chrF score went up slightly after enforcing the constraints on baseline, but degrades after an additional DPO fine-tuning. We will discuss this more detail in Section 6.

## 6 Analysis

To better understand model behavior beyond the aggregate ChrF metric, we performed qualitative error analysis on generated sonnets using the held-out development set. Our analysis examines structural properties of the generated poems, where generation diverges from the reference, and representative failure examples.

**Model learns global lexical similarity but misses sonnet structural constraints.** The model achieves a full-sonnet ChrF score of 42.35, but a much lower line-level ChrF of 14.81. This discrepancy suggests that while the model captures some global lexical similarity with Shakespearean language, it struggles to reproduce correct line-by-line structure. Two structural issues dominate the errors:

- **Incorrect line counts:** 11 of 12 generated sonnets do not contain exactly 14 lines.
- **Early divergence from the reference text:** the first mismatch typically occurs immediately after the prompt boundary (around lines 3–4).

This behavior suggests that the training objective prioritizes local token prediction over global structural constraints such as rhyme schemes, meter, and fixed line counts.

**Decoding constraints partially address structural failures.** Based on the observed failure modes, we introduced structural decoding constraints (Section 4.6) to enforce exactly 14 lines, require a newline after the prompt, and impose a minimum length for the final line. As shown in Table 4, applying these constraints slightly improves performance from 42.111 to 42.471. This suggests that some generation errors stem from decoding behavior rather than the model’s internal representations. However, applying DPO fine-tuning on top of the structurally constrained baseline reduces the ChrF score to 40.718. One possible explanation is that preference optimization shifts the model toward stylistic or lexical features captured in the preference data while interfering with structural patterns already encouraged by constrained decoding, distorting the learned representations.

**Negative example quality is critical for DPO learning.** The DPO negative-example ablation study shows that harder and more diverse corruptions produce stronger preference signals. As shown in Table 3, both rhyme violation and syntax corruption achieve performance close to the baseline (42.157 and 42.150 respectively). These corruption types modify local linguistic properties while

| Model                                   | Description                                                                                                                                                                           | Held-out ChrF |
|-----------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| Baseline                                | Cross-entropy training objective.                                                                                                                                                     | 42.111        |
| Baseline + Structural Constraints       | Structural constraints enforced during generation: exactly 14 lines, forced newline after the prompt, and the final line must contain at least 5 words.                               | 42.471        |
| Baseline + Structural Constraints + DPO | Initialized from the structurally constrained baseline and further optimized with DPO using mixed hard-negative data, where each negative sonnet contains at least three corruptions. | 40.718        |

Table 4: Effects of model variants on sonnet generation. All hyperparameters are held constant: GPT2-large, 10 epochs, learning rate  $1 \times 10^{-5}$ , and  $\beta = 0.1$  for DPO.

preserving overall sentence structure, creating moderately difficult negatives that help the model learn stylistic constraints. By contrast, line permutation dramatically disrupts the global structure of the poem and results in a substantial drop in ChrF score to 33.686. This likely occurs because the resulting negatives are trivially distinguishable from valid sonnets, reducing the informativeness of the preference signal. The mixed corruption strategy, which combines multiple corruption types, performs better than most individual strategies and achieves the highest ChrF score of 42.644. This suggests that exposing the model to diverse structural violations helps it learn richer preferences over valid sonnet structure. While rule-based corruptions directly encode domain knowledge about sonnet form to target specific structural properties, model-generated negative examples may better resemble real model errors.

**DPO optimization is only loosely correlated with ChrF.** Across all experiments, dev ChrF remains relatively low ( $< 43$ ). One plausible explanation is that the DPO objective optimizes a *relative preference likelihood* between pairs of outputs rather than directly optimizing similarity to a reference text. In contrast, ChrF measures character n-gram overlap with a single ground-truth continuation. Because creative generation tasks such as sonnet writing admit many valid continuations for the same prompt, improving preference alignment does not necessarily increase overlap with the specific reference used for evaluation. In other words, DPO learns to rank outputs that look more like valid sonnets compared to corrupted alternatives, whereas ChrF rewards outputs that are lexically similar to the reference continuation. These objectives are therefore only weakly aligned. This highlights a broader challenge in creative text generation: automatic metrics based on reference overlap may underestimate improvements in stylistic quality or structural correctness learned through preference optimization.

## 7 Conclusion

Overall, our results highlight that the effectiveness of parameter-efficient and alignment techniques is highly task-dependent, and that careful consideration of task characteristics is essential when selecting and tuning these methods. Flash Attention provides a consistent speedup of up to  $1.26\times$  for GPT-2 Medium, but does not provide much speedup for larger models. LoRA demonstrates task-dependent effectiveness, where the low-rank constraint cannot capture the task-dependent stylistic and structural patterns required for sonnet generation. DPO fine-tuning with carefully constructed preference data, both rule-based and model-generated, modestly improves sonnet generation quality. We compared and analyzed the importance of the quality of constructed data. For generation, we found that enforcing sonnet structures further improves the coherence and readability of the generated sonnets.

## Team Contributions

- Jessie Ou: Paraphrase DPO, rule-based corruptions for sonnet generation,  $\beta$  tuning, structure-enforcing generation.
- Weixin Yu: Sonnet DPO, randomized negative sample generation, FlashAttention integration, LoRA implementation, learning rate tuning, beam search.

## References

- [1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [2] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.
- [3] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.
- [4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.
- [5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [6] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.
- [7] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- [8] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6894–6910, 2021.
- [9] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.
- [10] Stanford CS224N Course Staff. Cs224n default final project instructions. [https://web.stanford.edu/class/cs224n/project/DFP\\_Instructions.pdf](https://web.stanford.edu/class/cs224n/project/DFP_Instructions.pdf), 2026. Stanford University.