

GDPO for GPT-2

Stanford CS224N Default Project

Ahmed Sherif Ahmed Elbakry Mohamed

Department of Computer Science

Stanford University

bakry@stanford.edu

Abstract

We investigate whether reinforcement learning with group-relative policy optimization can improve GPT-2 on the sonnet generation task, with a secondary focus on close-style paraphrase detection. The Sonnet Generation task requires balancing multiple objectives such as structural correctness, rhyme scheme adherence and fluency. We compare supervised fine-tuning (SFT) against Group Relative Policy Optimization (GRPO) and its group reward-decoupled normalization variant, GDPO, using reward components based on sonnet line count, rhyme scheme, and chrF similarity to reference sonnets. Across our experiments, GDPO generally matches or outperforms GRPO and achieves the best overall result, reaching a chrF score of 43.29 at group size $G=8$, compared to 42.73 for the best GRPO configuration and 42.59 for the SFT baseline. We further find that GDPO is more robust across group-size and various reward weights' settings being able to utilize these auxiliary rewards to improve on chrF. These results suggest that GDPO is a promising approach for Sonnet Generation task, where output quality depends on satisfying several objectives simultaneously.

1 Key Information to include

- Mentor: Tristan Thrush ¹
- External Collaborators (if you have any): N/A
- Sharing project: N/A

2 Introduction

Recent progress in large language models has shown that scaling pretraining compute, model size, and dataset size can produce strong emergent capabilities [1]. However, pretrained models alone often do not reliably satisfy the constraints that matter in real world use. These constraints / preferences span across several areas such as improved control / steering, improved alignment with human values and mitigating undesirable behaviors.

These constraints have motivated the use of Reinforcement Learning in LLM post-training in order to further align models with human-oriented goals. In particular, human judgments can be used to transfer knowledge of what humans consider desirable through Reinforcement Learning from Human Feedback (RLHF) [2]. Several key algorithms have been proposed for this alignment stage, including Proximal Policy Optimization (PPO) [3], which became a standard choice in early RLHF pipelines, and more recent alternatives such as Direct Preference Optimization (DPO) [4], which reformulates preference alignment as a simpler supervised objective without requiring explicit online reinforcement learning.

Another prominent technique, Group Relative Policy Optimization (GRPO) [5], has gained widespread attention because it removes the need for an explicit value model / critic by estimating the

¹tthrush@stanford.edu

advantages from a group of rollouts for the same prompt, massively reducing training computations costs as typically the value model is of the same size as the policy model, while achieving strong performance on a variety of tasks such as Coding, Maths and Tool Calling, where rewards can be accurately and efficiently computed.

One important consideration is that GRPO is well defined for a single objective setting, while naively extending it to the multi-objective setting obscures important information on each reward component and how well the model performed on that. Consequently this can negatively affect the training process.

To address this, we study Group reward-Decoupled Normalization (GDPO) [6], which extends GRPO to the multi-reward setting. Instead of relying on a single reward function, GDPO proposes to perform group-wise decoupled normalization of each reward individually prior to aggregation into the final advantage values to maintain the integrity of the signal coming from each individual reward. This makes GDPO a natural fit for constrained creative generation tasks such as sonnet writing, where quality is best expressed as a combination of formal, stylistic, and fluency-based objectives. In this project, we investigate whether such multi-reward reinforcement learning can improve GPT-2 for Shakespearean sonnet generation beyond supervised fine-tuning alone.

In this paper, we initialize from a supervised fine-tuned GPT-2 and compare standard supervised fine-tuning against reinforcement learning with GRPO and GDPO. We define a multi-reward objective combining chrF similarity with structural rewards for line-count correctness and rhyme adherence, and we evaluate whether decoupled reward normalization improves optimization behavior and final generation quality over naive reward aggregation.

Our main contributions are as follows:

- We formulate Shakespearean sonnet generation as a multi-objective reinforcement learning problem, where quality depends jointly on fluency, structural correctness and style.
- We implement and compare GRPO and GDPO for this setting on top of a GPT-2 baseline, alongside a supervised fine-tuning baseline.
- Through ablation studies over group size, and reward weighting, we show that GDPO generally matches or outperforms GRPO, with the best configuration achieving a chrF score of 43.29 compared to 42.70 for the best GRPO setting and 42.59 for supervised fine-tuning.

In this project, we focus mainly on the Sonnet Generation task, and report the final metrics obtained from SFT on the Close-style paraphrasing task.

3 Related Work

Recent progress in LLM alignment has been driven by post-training methods that optimize models toward human-preferred behavior beyond pretraining alone. Reinforcement Learning from Human Feedback (RLHF) became a standard paradigm for this purpose, with InstructGPT [2] demonstrating that preference-informed post-training via Proximal Policy Optimization (PPO) can substantially improve instruction following and perceived helpfulness. Subsequent work proposed simpler alternatives such as Direct Preference Optimization (DPO) [4], which replaces explicit online reinforcement learning with a direct preference-learning objective.

Group Relative Policy Optimization (GRPO) [5] has emerged as an attractive alternative to PPO-style RLHF because it removes the need for an explicit value model and instead computes advantages from groups of sampled responses for the same prompt. This makes GRPO especially appealing in settings where rewards can be functionally verified over full completions. However, GRPO is well defined for mainly single objective tasks. Dr.GRPO [7] and DeepSeek-v3.2 [8] experimented with a variant of multi-objective GRPO that removes the standard deviation term in the advantage computation, while this preserves more signal regarding different rewards, it still does not address the main issue behind naively summing the rewards. Recent work on GDPO argues that naively applying GRPO in multi-reward settings can collapse distinct reward combinations into similar advantage values, weakening the optimization signal. GDPO addresses this by normalizing each reward dimension separately before aggregation, thereby preserving more information about how a model performs on each objective. Our work is a direct empirical study of this idea in a constrained generation setting.

4 Approach

GPT-2, a decoder-only auto-regressive transformer, was one of the early influential LLMs, it demonstrated that scaling model size and dataset can give strong zero-shot performance on various downstream tasks [9]. In this paper, we use GPT-2 as the foundation for our experiments. We establish zero-shot baselines using GPT-2, then explore SFT and RL on top of the pretrained model.

Reward Construction Sonnet Generation lends itself naturally to multiple objectives beyond regular pretraining and supervised finetuning. Particularly a Shakespearean Sonnet consists of 3 quatrains followed by a final couplet, with a total of 14 lines. It must be composed in iambic pentameter, that is it should follow a rhyme scheme of ABAB CDCD EFEF GG, where each character denotes the rhyme scheme for the word ending that line. For example ABAB denotes that the last word of line 1 should rhyme with the last word of line 3, and the last word of line 2 should rhyme with the last word of line 4. To this end, a well trained model should be able to adhere to the following constraints in addition to closely using Shakespearean style, beyond next word prediction task.

We formulate 3 main rewards for the multi-objective reward:

- **Length reward:** encourages exactly 14 generated lines. The reward is computed by examining the model’s generation and calculating the number of lines. Reward is 1 if and only if the number of lines is equal to 14, otherwise 0.
- **Rhyme reward:** encourages the Shakespearean rhyme scheme ABAB CDCD EFEF GG, computed from the final word of each line using a pronunciation lexicon. We use CMUDict² for our rhyme based reward.
- **chrF based reward:** optimizes the model for chrF, encouraging higher similarity with the reference Shakespearean text.

These rewards are going to be used in the Reinforcement Learning stage to encourage the model to generate higher quality sonnets.

GDPO Recall that for a given prompt q drawn from a dataset D , GRPO samples a set of G responses $\{o_j\}_{j=1}^G$ from the old policy $\pi_{\theta_{\text{old}}}(\cdot | q)$, assigns each response a reward based on the sum of multiple rewards $r_{\text{sum}}^{(q,j)} = r_1^{(q,j)} + r_2^{(q,j)} + \dots + r_N^{(q,j)}$ for N rewards, then computes a group-relative advantage by standardizing rewards within the group:

$$A^{(q,j)} = \frac{r_{\text{sum}}^{(q,j)} - \text{mean}\left(\left\{r_{\text{sum}}^{(q,1)}, \dots, r_{\text{sum}}^{(q,G)}\right\}\right)}{\text{std}\left(\left\{r_{\text{sum}}^{(q,1)}, \dots, r_{\text{sum}}^{(q,G)}\right\}\right)} \quad (1)$$

Then the multi-reward GRPO objective is defined as

$$\mathcal{J}_{\text{GRPO}}(\theta) = \mathbb{E} \left[\frac{1}{G} \sum_{j=1}^G \frac{1}{|o_j|} \sum_{t=1}^{|o_j|} \min\left(s_{j,t}(\theta) A^{(q,j)}, \text{clip}(s_{j,t}(\theta), 1 - \epsilon, 1 + \epsilon) A^{(q,j)}\right) \right],$$

where $q \sim D$, $\{o_j\}_{j=1}^G \sim \pi_{\theta_{\text{old}}}(\cdot | q)$, $s_{j,t}(\theta) = \frac{\pi_{\theta}(o_j^t | q, o_j^{<t})}{\pi_{\theta_{\text{old}}}(o_j^t | q, o_j^{<t})}$, and ϵ is the clipping threshold.

(2)

The KL divergence term is omitted from (2) for clarity. The main issue with the above approach is that group-wise normalization is applied directly to r_{sum} which can collapse different combinations of rewards into the same advantage, as a result we get a less meaningful training signal, unequal learning across the multiple objectives and in some cases early training failure.

To overcome the limitation in the multi-objective GRPO, Group reward-Decoupled normalization Policy Optimization (GDPO) proposes a modification, instead of performing group-wise normaliza-

²<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>

tion to the aggregated r_{sum} , GDPO performs group-wise normalization to each reward separately before aggregation. Each advantage for a given prompt q and rollout j is computed as

$$A_k^{(q,j)} = \frac{r_k^{(q,j)} - \text{mean}\left(\left\{r_k^{(q,1)}, \dots, r_k^{(q,G)}\right\}\right)}{\text{std}\left(\left\{r_k^{(q,1)}, \dots, r_k^{(q,G)}\right\}\right)}, \quad k = 1, \dots, n. \quad (3)$$

then the overall advantage is obtained by summing the normalized advantages across all objectives:

$$A_{\text{sum}}^{(q,j)} = \sum_{k=1}^n A_k^{(q,j)}, \quad \hat{A}_{\text{sum}}^{(q,j)} = \frac{A_{\text{sum}}^{(q,j)} - \text{mean}\left(\left\{A_{\text{sum}}^{(q',j')} \mid q' \in D_{\text{Batch}}, j' = 1, \dots, G\right\}\right)}{\text{std}\left(\left\{A_{\text{sum}}^{(q',j')} \mid q' \in D_{\text{Batch}}, j' = 1, \dots, G\right\}\right)}. \quad (4)$$

The reformulated advantage computation is used instead of (1) in (2).

5 Experiments

5.1 Data

The Sonnet Dataset consists of 154 sonnets written by Shakespeare. The dataset has been split into train, dev & test. During training we only use the dev portion to tune our model, then once we have the best model we submit that for evaluation on the held-out test set.

Task / Dataset	Train	Dev	Test
Sonnet Generation (Shakespeare Sonnets)	131	12	12
Cloze-Style Paraphrase Detection (Quora QQP subset)	141,506	20,215	40,431

Table 1: Dataset split statistics for the default project tasks.

For dev and test, the model is given only the first 3 lines of a Sonnet and tasked with generating the rest. We also include the dataset statistics of the close-style paraphrasing task.

5.2 Evaluation method

For model evaluation, the main metric we use to track the model’s performance is chrF [10], a character level n-gram score between the predicted model outputs and the ground truth sonnets. chrF effectively balances precision and recall for the character n-grams, providing a holistic overview of the model’s performance.

Additionally, for GRPO and GDPO where we have verifiable rewards, we keep track of the rewards values and use it to measure the training progress.

5.3 Experimental details

For our main experiments we use the pretrained GPT-2 small model, which is a 12 layer, 768 hidden dimension and 12 attention heads with a total of 117M parameters. The model is implemented from scratch based on the Default Project Handout, and all missing portions have been implemented and verified to be functioning correctly.

Since the dataset for the Sonnet Generation task is small, the model becomes more prone to overfitting. As a result, we use the AdamW [11] optimizer with weight decay set to 0.1 for all of our experiments in an attempt to increase regularization and improve generalization. For the initial SFT experiments we kept the learning rate set to $1e - 4$, and used a batch size of 16, training for a total of 20 epochs.

Decoding was kept constant across all experiments with a top_p sampling of 0.9 and a temperature of 1.2. This seemed to work well with our initial SFT experiments as well as in our RL experiments, where these generation parameters control the quality of the rollouts and rewards calculated.

5.4 Results

5.4.1 Baselines:

Close-Style Paraphrasing:

Paraphrase results. For paraphrase detection, we first evaluated a zero-shot baseline using the pretrained model without supervised fine-tuning. We then ran supervised fine-tuning (SFT) with a linear layer for binary classification.

Method	Dev Acc.	Dev F1
Zero-shot baseline	0.372	0.287
SFT + linear-layer	0.889	0.882

Table 2: Paraphrase detection results on the dev set. The zero-shot baseline performs substantially worse than SFT with a linear layer.

Sonnet Generation:

Zero-Shot: We establish zero-shot prompt based baseline by constructing 3 prompt versions Prompt A (§B.1) no prompt and provides the first 3 lines directly to the model. Prompt B (§B.2) explicitly specifies the 14-line Shakespearean rhyme scheme. Prompt C (§B.3) provides a minimal Shakespeare-style instruction. Using these prompts we evaluate the pretrained GPT-2 small on the dev set and report the results in Table 3. Prompt C performed best and scored 41.1 chrF on the dev set.

Supervised Fine-Tuning. To obtain a stronger task-specific baseline and initialize reinforcement-learning-based post-training, we next perform supervised fine-tuning (SFT) on pretrained GPT-2 small using the sonnet generation training data. This stage serves two purposes. First, it measures the gains obtainable from SFT alone. Second, it provides the starting checkpoint for both GRPO and GDPO, ensuring that reinforcement learning begins from a model that has already learned the basic structure and style of the task. As shown in Table 3, supervised fine-tuning improves substantially over all zero-shot baselines, reaching a chrF score of 42.59.

5.4.2 Reinforcement Learning Results

Starting from the supervised fine-tuned checkpoint, we compare GRPO and GDPO on the sonnet generation dev set, where we used a more conservative learning rate of $5e-6$ to avoid overfitting and complete mode collapse. Table 3 summarizes the best chrF obtained for each method under our selected configurations, together with the zero-shot and SFT baselines. Overall, both RL methods are competitive with or better than supervised fine-tuning, but GDPO achieves the strongest result. In particular, the best GRPO configuration reaches a chrF of 42.73, while the best GDPO configuration reaches 43.29, exceeding the SFT baseline of 42.59 by 0.70 chrF and the best GRPO result by 0.56 chrF.

We also compare GRPO and GDPO under a shared baseline RL setting with group size $G=10$, KL coefficient $\beta=0.2$, and equal reward weights. Under this matched setup, GRPO reaches a best chrF of 42.70, while GDPO reaches 43.06. GDPO also produces stronger auxiliary reward values, improving exact 14-line generation from 0.083 to 0.417 and rhyme reward from 0.0139 to 0.0417. This indicates that GDPO not only improves the primary evaluation metric, but also better satisfies the structural constraints of the task.

Training Stability. One notable challenge during training was the instability of the reward signal. As shown in Figure 1, the combined reward curves for both GRPO and GDPO remain noisy and exhibit oscillation rather than smooth consistent improvement. This behavior could be justified in both methods as they optimize sequence-level rewards estimated from sampled rollouts, so the update signal is inherently high-variance, and this variance is further amplified by the multi-objective nature of the task. In particular, improvements in one reward component, such as line count or rhyme, do not always translate to immediate gains in the combined reward. We also observed that increasing the learning rate and relaxing the KL penalty allowed the model to optimize the line and rhyme

Method	Configuration	dev chrF	test chrF
Zero-shot	Prompt A	40.73	-
	Prompt B	40.65	-
	Prompt C	41.10	-
Supervised Fine-Tuning	GPT-2 small + SFT	42.59	-
Reinforcement Learning	GRPO ($G=10$, $\beta=0.1$)	42.73	42.291
	GDPO ($G=8$, $\beta=0.2$)	43.29	41.86

Table 3: Dev-set chrF results across zero-shot prompting, supervised fine-tuning, and reinforcement-learning-based post-training. Supervised fine-tuning improves substantially over zero-shot prompting, and GDPO achieves the strongest overall performance among all evaluated methods.

Method	chrF	Line Reward	Rhyme Reward
GRPO	42.70	0.083	0.0139
GDPO	43.06	0.417	0.0417

Table 4: GRPO vs. GDPO under the shared baseline RL setting ($G=10$, $\beta=0.2$, equal reward weights). GDPO improves both chrF and structural reward satisfaction.

rewards better, but at the same time this harmed the model severely on the main chrF metric. Overall, these results suggest that while RL-based post-training can improve sonnet quality, careful tuning of optimization hyperparameters remains important for balancing exploration, policy improvement, and stability.

Additionally, the loss kept increasing for GDPO, primarily driven by the increase in the KL loss, while GRPO did not exhibit the same behaviour. This could suggest that GDPO drifted more from the reference model as training progressed.

Baseline Training Reward Curves: GRPO vs GDPO ($G=10$, $\beta=0.2$)

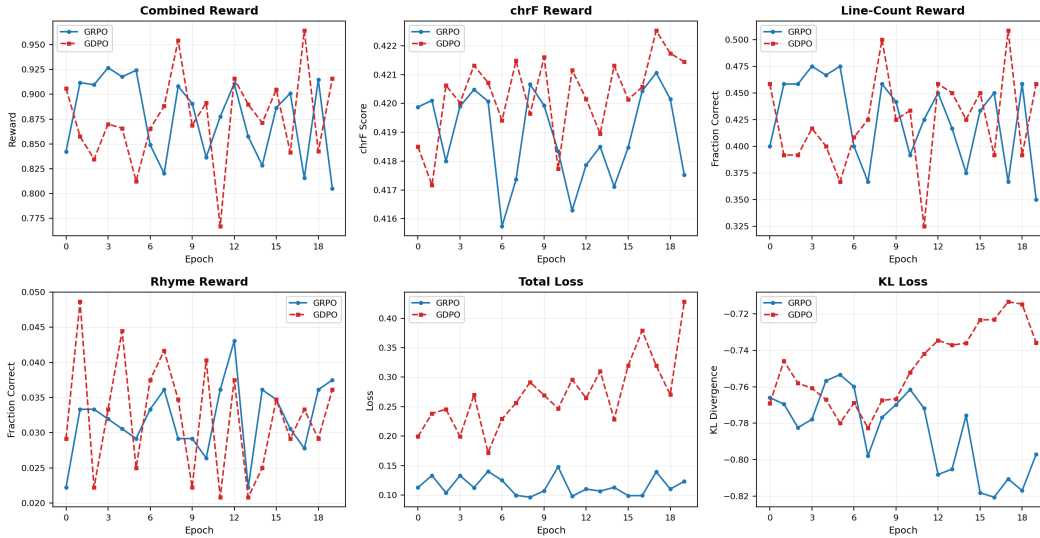


Figure 1: Training Metrics for GRPO and GDPO in the Baseline RL experiments.

Summary of results. These results show that reinforcement-learning-based post-training improves over supervised fine-tuning alone, and that GDPO is the strongest overall method in our experiments. Its gains are reflected both in chrF and in better adherence to structural constraints such as correct sonnet length and rhyme.

6 Analysis

Effect of group size. We vary the rollout group size $G \in \{4, 8, 10, 16\}$ to study how the quality of group-relative advantages affects training. Table 5 shows that GRPO is relatively insensitive to this choice and peaks at $G=10$ with chrF 42.70. In contrast, GDPO benefits more from a moderate group size and achieves its best overall result at $G=8$, reaching 43.29 chrF. Increasing the group size further to $G=16$ does not improve either method. A likely explanation is that very large groups dilute the relative-advantage signal, while very small groups do not provide enough variation for reliable advantage computation.

Method	$G=4$	$G=8$	$G=10$	$G=16$
GRPO	42.52	42.43	42.70	42.56
GDPO	42.63	43.29	43.06	42.85

Table 5: Effect of rollout group size on dev-set chrF. GDPO performs best at a moderate group size of $G=8$.

Effect of reward weighting. To better understand how each algorithm handles competing objectives, we vary the relative weights assigned to chrF, line count, and rhyme. Table 6 shows that upweighting chrF by $3\times$ gives the strongest GRPO result, reaching 42.97 chrF. This is expected, since chrF is also the main evaluation metric. Contrast to GRPO, GDPO achieves its best results of 43.06 with an equal weight setting. This could imply that GDPO benefited from line & rhyme rewards more than GRPO.

Weights	Method	chrF	Line Reward	Rhyme Reward
Equal	GRPO	42.70	0.083	0.0139
	GDPO	43.06	0.417	0.0417
$\text{chrF} \times 3$	GRPO	42.97	0.250	0.0417
	GDPO	42.64	0.417	0.0556
$\text{Lines} \times 3$	GRPO	42.60	0.583	0.0417
	GDPO	42.58	0.500	0.0000
$\text{Rhyme} \times 3$	GRPO	42.64	0.500	0.0000
	GDPO	42.77	0.417	0.0556

Table 6: Reward-weight ablation on the dev set.

Reward Choices Reward values reveal several things. First is that the exact 14-line generation is learnable: both GRPO and GDPO improve line-count adherence relative to the SFT baseline, and line reward can become substantially higher when explicitly emphasized. In contrast, rhyme remains difficult across all settings, with rhyme reward staying below 0.06 and in some cases it is 0.

To better understand the performance of our rhyme checker we plot the reward scores computed on the actual training data in Figure 2. The rhyme based checker has a mean score of 0.833, indicating that there are cases in the ground truth where they are marked as not rhyming despite being correct.

In Table 7, we present several failed cases that were marked as not rhyming. The main issue behind this is that these words can have different pronunciation, and since the rhyme checker relies on the phones in CMUDict these were marked as non rhyming.

Why GDPO helps. The main advantage of GDPO appears in settings where we have multiple objectives. In our task, chrF rewards stylistic similarity to the reference, while line-count and rhyme rewards enforce explicit structural constraints that should indirectly contribute to a better chrF. If these rewards are summed before normalization, different reward combinations can collapse into similar advantage values, weakening the learning signal. GDPO avoids this by normalizing each reward separately before aggregation, which likely explains its stronger structural performance and best overall chrF. Despite the training instability, GDPO consistently performed better than GRPO under different settings.

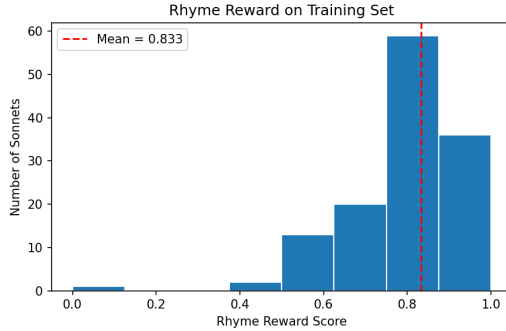


Figure 2: Rhyme rewards on the Ground Truth of the training set

Sonnet ID	Lines	Word 1	Word 2
0	[1, 3]	die	memory
0	[8, 10]	ornament	content
1	[1, 3]	field	held
3	[8, 10]	alone	gone

Table 7: Rhyme-checker errors

7 Conclusion

In this work, we explored reinforcement-learning-based post-training for Shakespearean sonnet generation, a constrained creative-generation task that naturally requires balancing multiple objectives. Starting from a supervised fine-tuned GPT-2 baseline, we compared GRPO and GDPO using rewards based on chrF, exact 14-line structure, and Shakespearean rhyme adherence. Our results GDPO and GRPO can improve upon the SFT baseline, and that GDPO is the strongest overall method in our setting. In particular, GDPO achieved the best dev-set performance of 43.29 chrF, outperforming both the SFT baseline and the best GRPO configuration, while also producing stronger structural reward values under matched settings.

Our experiments also highlight the challenges of RL-based post-training on a small dataset, and investigate potential issues with our rhyme based checker due to different pronunciations.

Overall, these findings indicate that GDPO is a promising approach for structured text generation tasks in which quality cannot be captured by a single reward alone.

Future work may include a revision of the rhyme based checker to improve its coverage across the Shakespeare training dataset as well as the exploration of additional rewards signals that can help the model adhere to the Shakespearean sonnets.

References

- [1] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020.
- [2] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
- [3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

- [4] 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, 2024.
- [5] 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, 2024.
- [6] Shih-Yang Liu, Xin Dong, Ximing Lu, Shizhe Diao, Peter Belcak, Mingjie Liu, Min-Hung Chen, Hongxu Yin, Yu-Chiang Frank Wang, Kwang-Ting Cheng, Yejin Choi, Jan Kautz, and Pavlo Molchanov. Gdpo: Group reward-decoupled normalization policy optimization for multi-reward rl optimization, 2026.
- [7] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025.
- [8] DeepSeek-AI, Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenhao Xu, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Erhang Li, Fangqi Zhou, Fangyun Lin, Fucong Dai, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Hao Li, Haofen Liang, Haoran Wei, Haowei Zhang, Haowen Luo, Haozhe Ji, Honghui Ding, Hongxuan Tang, Huanqi Cao, Huazuo Gao, Hui Qu, Hui Zeng, Jialiang Huang, Jiashi Li, Jiaxin Xu, Jiewen Hu, Jingchang Chen, Jingting Xiang, Jingyang Yuan, Jingyuan Cheng, Jinhua Zhu, Jun Ran, Jinguang Jiang, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Kexin Huang, Kexing Zhou, Kezhao Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Wang, Liang Zhao, Liangsheng Yin, Lihua Guo, Lingxiao Luo, Linwang Ma, Litong Wang, Liyue Zhang, M. S. Di, M. Y. Xu, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Panpan Huang, Peixin Cong, Peiyi Wang, Qiancheng Wang, Qihao Zhu, Qingyang Li, Qinyu Chen, Qiushi Du, Ruiling Xu, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runqiu Yin, Runxin Xu, Ruomeng Shen, Ruoyu Zhang, S. H. Liu, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaofei Cai, Shaoyuan Chen, Shengding Hu, Shengyu Liu, Shiqiang Hu, Shirong Ma, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, Songyang Zhou, Tao Ni, Tao Yun, Tian Pei, Tian Ye, Tianyuan Yue, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjie Pang, Wenjing Luo, Wenjun Gao, Wentao Zhang, Xi Gao, Xiangwen Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaokang Zhang, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xingyou Li, Xinyu Yang, Xinyuan Li, Xu Chen, Xuecheng Su, Xuehai Pan, Xuheng Lin, Xuwei Fu, Y. Q. Wang, Yang Zhang, Yanhong Xu, Yanru Ma, Yao Li, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Qian, Yi Yu, Yichao Zhang, Yifan Ding, Yifan Shi, Yiliang Xiong, Ying He, Ying Zhou, Yinmin Zhong, Yishi Piao, Yisong Wang, Yixiao Chen, Yixuan Tan, Yixuan Wei, Yiyang Ma, Yiyuan Liu, Yonglun Yang, Yongqiang Guo, Yongtong Wu, Yu Wu, Yuan Cheng, Yuan Ou, Yuanfan Xu, Yuduan Wang, Yue Gong, Yuhan Wu, Yuheng Zou, Yukun Li, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehua Zhao, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhixian Huang, Zhiyu Wu, Zhuoshu Li, Zhuping Zhang, Zian Xu, Zihao Wang, Zihui Gu, Zijia Zhu, Zilin Li, Zipeng Zhang, Ziwei Xie, Ziyi Gao, Zizheng Pan, Zongqing Yao, Bei Feng, Hui Li, J. L. Cai, Jiaqi Ni, Lei Xu, Meng Li, Ning Tian, R. J. Chen, R. L. Jin, S. S. Li, Shuang Zhou, Tianyu Sun, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xinnan Song, Xinyi Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, Dongjie Ji, Jian Liang, Jianzhong Guo, Jin Chen, Leyi Xia, Miaojun Wang, Mingming Li, Peng Zhang, Ruyi Chen, Shangmian Sun, Shaoqing Wu, Shengfeng Ye, T. Wang, W. L. Xiao, Wei An, Xianzu Wang, Xiaowen Sun, Xiaoxiang Wang, Ying Tang, Yukun Zha, Zekai Zhang, Zhe Ju, Zhen Zhang, and Zihua Qu. Deepseek-v3.2: Pushing the frontier of open large language models, 2025.
- [9] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [10] Maja Popović. chrF: character n-gram F-score for automatic MT evaluation. In Ondřej Bojar, Rajan Chatterjee, Christian Federmann, Barry Haddow, Chris Hokamp, Matthias Huck, Varvara Logacheva, and Pavel Pecina, editors, *Proceedings of the Tenth Workshop on Statistical Machine*

Translation, pages 392–395, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

A Appendix

B Baseline Sonnet Prompts

B.1 Prompt A

Sonnet Generation – Prompt A

{first_3_lines}

B.2 Prompt B

Sonnet Generation – Prompt B

Complete this Shakespearean sonnet with 14 lines total, with a rhyme scheme of ABAB CDCD EFEF GG:
{first_3_lines}

B.3 Prompt C

Sonnet Generation – Prompt C

You are Shakespeare. Complete this sonnet:
{first_3_lines}