

GRPOET-Rank: Group Relative Policy Optimization with External Text-Ranking

Stanford CS224N Default Project

Jamin Xie

Department of Electrical Engineering
Stanford University
jmx@stanford.edu

William Liu

Department of Computer Science
Stanford University
wzliu@stanford.edu

Eric Liang

Department of Computer Science
Stanford University
ehliang@stanford.edu

Abstract

Aligning large language models with human preferences often uses on Reinforcement Learning from Human Feedback (RLHF), where a learned scalar reward model guides an RL optimizer such as PPO [1] or GRPO [2], which introduces significant computational overhead and calibration challenges for subjective, creative tasks. We propose GRPOET-Rank, a novel adaptation of the Oracle-RLAIF [3] framework for pure-text generative models, to optimize policy alignment using ordinal ranking feedback instead of continuous scalar rewards. By employing an off-the-shelf “Oracle” language model to rank candidate responses generated by a GPT-2 policy, we eliminate the need for an explicit critic or reward model, greatly reducing memory requirements. We introduce a rank-aware Group Relative Policy Optimization (GRPO-Rank) objective that utilizes normalized Discounted Cumulative Gain (nDCG) to penalize policy deviations. We evaluate our methodology on stylistically constrained Shakespearean sonnet generation. Our framework presents a robust, computationally efficient alternative for aligning smaller language models in creative and highly structured domains.

1 Key Information to include

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

Team member	Late days remaining
Jamin Xie	1
William Liu	5
Eric Liang	5
Total pooled	11
Team size	3
Late days applied for the report	3

2 Introduction

The alignment of large language models (LLMs) with human intent has predominantly relied on Reinforcement Learning from Human Feedback (RLHF), typically utilizing Proximal Policy Optimization (PPO) [1]. However, the standard RLHF pipeline introduces significant computational and logistical bottlenecks. It requires training a separate reward model (RM) to predict scalar scores, effectively doubling the memory footprint during training. Furthermore, relying on scalar rewards is notoriously difficult to calibrate, particularly for inherently subjective domains such as creative text generation, where assigning an absolute numerical score to a poem is reductive and noisy.

Group Relative Policy Optimization (GRPO) [2] addresses part of this problem by eliminating the critic network and instead normalizing rewards across a group of sampled responses. Yet GRPO still depends on a scalar reward signal, which remains expensive to obtain for creative and structurally constrained tasks. More recently, the Oracle-RLAIF framework [3] demonstrated that an off-the-shelf LLM can serve as a ranker in place of a trained reward model, using ordinal feedback and nDCG-based penalties to guide policy optimization in multi-modal settings.

We propose GRPOET-Rank, a novel adaptation of the Oracle-RLAIF framework to fine-tune smaller, pure-text autoregressive models. Specifically, we apply rank-aware GRPO (GRPO-Rank) to a GPT-2 [4] policy, using Gemini 2.5 Flash [5] as the external Oracle ranker. By replacing scalar reward estimation with ordinal ranking feedback, we completely remove the need to train and maintain a separate reward model or critic network. We evaluate our approach on Shakespearean sonnet generation with rhyme-scheme adherence as the ranking criterion.

Our primary contributions are as follows:

1. We adapt the Oracle-RLAIF ranking framework from multi-modal video-language models to a pure-text generative setting.
2. We formulate a GRPO-Rank loss objective using nDCG-based penalties for structurally constrained creative text generation.
3. We compare our approach against a supervised fine-tuning (SFT) baseline.

3 Related Work

PPO and Standard RLHF. Proximal Policy Optimization [1] is the dominant algorithm for RLHF-based alignment. PPO optimizes a policy while constraining updates via a clipped surrogate objective and typically requires a continuously updated critic network to estimate value baselines. While effective at scale, the dual-model requirement (policy and critic) imposes substantial memory overhead that limits applicability on constrained hardware.

Reinforcement Learning from AI Feedback (RLAIF). RLAIF replaces costly human annotations with AI-generated feedback, using a language model as a judge [6]. This reduces the annotation bottleneck but traditionally still requires training a specialized reward model to produce calibrated scalar values from the AI feedback, preserving much of the computational cost of standard RLHF.

Group Relative Policy Optimization (GRPO). Introduced alongside DeepSeekMath [2], GRPO eliminates the critic model entirely. For each prompt, it samples a group of K candidate responses, obtains scalar rewards for each, and normalizes these rewards within the group to compute advantages. This group-relative normalization serves as the baseline in place of a learned value function, significantly reducing memory requirements.

Direct Preference Optimization (DPO). DPO [7] bypasses the reward model entirely by reparameterizing the RLHF objective as a classification loss over preference pairs. It optimizes the policy directly from offline preference data in the form of (x, y_w, y_l) triples, where y_w is the preferred response and y_l is the dispreferred one. DPO is computationally efficient but requires access to high-quality preference data, which may not be naturally available for creative generation tasks.

Oracle-RLAIF. The Oracle-RLAIF framework [3] was proposed for multi-modal video-language models and replaces trained scalar reward models with an Oracle LLM ranker. It introduces $\text{GRPO}_{\text{rank}}$, which substitutes scalar reward deviations with rank-aware nDCG penalties. Our work adapts this framework to pure-text autoregressive generation, demonstrating its viability for structurally constrained creative tasks.

4 Approach

4.1 Base Architecture and Generation

We utilize a decoder-only GPT-2 [4] architecture (117M parameters) equipped with masked multi-head self-attention mechanisms. During the training loop, rather than executing standard supervised fine-tuning (SFT) or estimating expected returns via a value function, our policy π_θ autoregressively generates a group of K candidate responses G_i for a given prompt.

4.2 Oracle Ranking and nDCG Penalties

An external Oracle (Gemini 2.5 Flash [5]) evaluates the K candidates against task-specific criteria and ranks them from best to worst. For sonnet generation, the Oracle ranks based on adherence to the ABAB CDCD EFEF GG rhyme scheme.

To shape the gradient updates during backpropagation, we calculate a rank-aware advantage function $\hat{A}_{rank}$ that utilizes normalized Discounted Cumulative Gain (nDCG). The objective penalizes incorrect rankings near the top exponentially more than discrepancies at the bottom. The penalty term δ_j for a candidate j measures the deviation of the model’s internal log-probability ranking from the Oracle’s ground-truth ranking:

$$\delta_j = 1 - nDCG_j = 1 - \frac{DCG(rank_j)}{DCG(rank^*)} \quad (1)$$

where the Discounted Cumulative Gain is defined as:

$$DCG(rank) = \frac{\frac{1}{1+rank}}{\log_2(2 + rank)} \quad (2)$$

4.3 Rank-Aware Advantage Function

By framing the feedback distribution as ordinal rather than continuous, we bypass the high variance associated with scalar reward estimation. The final advantage $\hat{A}_{rank}$ assigned to candidate i is the residual difference between its specific penalty and the expected group penalty:

$$\hat{A}_{rank,i} = \mathbb{E}_{j \in G_i}[\delta_j] - \delta_i \quad (3)$$

where the expected group penalty operates as a baseline:

$$\mathbb{E}_{j \in G_i}[\delta_j] = \frac{1}{K} \sum_{j \in G_i} \delta_j \quad (4)$$

This advantage is then fed into a standard clipped policy gradient objective. By eliminating the parameter-heavy critic network, this formulation drastically reduces memory overhead, making RL-style post-training highly viable for smaller compute budgets.

4.4 GRPO-Rank objective

The RL objective to be maximized, based on the GRPO framework and a rank-aware advantage function, is formulated as:

$$\begin{aligned}
\mathcal{J}_{\text{GRPO-Rank}}(\theta) = & \frac{1}{G} \sum_{i=1}^G \frac{1}{|o_i|} \sum_{t=1}^{|o_i|} \left[\min \left(r_{i,t}(\theta) \hat{A}_{\text{rank},i}, \text{clip}(r_{i,t}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_{\text{rank},i} \right) \right. \\
& - \beta \cdot D_{\text{KL}}(\pi_{\theta}(\cdot | q, o_{i,<t}) \parallel \pi_{\theta_{\text{ref}}}(\cdot | q, o_{i,<t})) \\
& \left. + c_{\text{entropy}} \cdot \mathcal{H}(\pi_{\theta}(\cdot | q, o_{i,<t})) \right] \tag{5}
\end{aligned}$$

where the policy importance ratio $r_{i,t}(\theta)$ is explicitly defined as:

$$r_{i,t}(\theta) = \frac{\pi_{\theta}(o_{i,t} | q, o_{i,<t})}{\pi_{\theta_{\text{old}}}(o_{i,t} | q, o_{i,<t})} \tag{6}$$

It is important to distinguish between the two frozen policies used in this formulation: $\pi_{\theta_{\text{old}}}$ is the policy from the start of the current update step, used to compute the importance ratio for the clipped PPO surrogate, whereas $\pi_{\theta_{\text{ref}}}$ is the static SFT checkpoint used to anchor the KL divergence penalty.

Unlike standard implementations that approximate the KL divergence using only the sampled token, we calculate the exact analytical KL divergence and Shannon entropy across the full vocabulary $\mathcal{V}$ at each timestep t :

$$D_{\text{KL}}(\pi_{\theta} \parallel \pi_{\theta_{\text{ref}}}) = \sum_{v \in \mathcal{V}} \pi_{\theta}(v | q, o_{i,<t}) \log \frac{\pi_{\theta}(v | q, o_{i,<t})}{\pi_{\theta_{\text{ref}}}(v | q, o_{i,<t})} \tag{7}$$

$$\mathcal{H}(\pi_{\theta}) = - \sum_{v \in \mathcal{V}} \pi_{\theta}(v | q, o_{i,<t}) \log \pi_{\theta}(v | q, o_{i,<t}) \tag{8}$$

We choose to use the exact KL because our base GPT-2 model is small; therefore, we can afford to iterate across the full probability distribution and entire vocabulary for more stabilized training.

The first term in our objective represents the standard clipped PPO surrogate, replacing the typical reward-based advantage with our rank-aware advantage $\hat{A}_{\text{rank},i}$. The second term applies a KL-divergence penalty, anchoring the policy to the initial SFT checkpoint ($\pi_{\theta_{\text{ref}}}$) to prevent catastrophic forgetting of the sonnet structure. Finally, while we include an exact entropy bonus to encourage exploration, we set $c_{\text{entropy}} = 0$ for our primary experiments, as promoting excessive token diversity tends to break the rigid meter and rhyme constraints of the sonnet format. Exploring non-zero entropy coefficients remains an area for future work.

Because optimization frameworks are designed to minimize a loss function, the actual loss $\mathcal{L}(\theta)$ computed and minimized during training (which is also the metric reported in our learning curves) is simply the negative of the expected objective:

$$\mathcal{L}_{\text{GRPO-Rank}}(\theta) = -\mathcal{J}_{\text{GRPO-Rank}}(\theta) \tag{9}$$

4.5 Supervised Fine-Tuning Baseline

Prior to applying GRPO-Rank, we first fine-tune the pretrained GPT-2 model via standard supervised fine-tuning (SFT) on each task’s training data. This SFT checkpoint serves as the reference policy π_{ref} for the subsequent RL stage, providing a strong initialization that ensures the model can already produce coherent task-relevant outputs before ranking-based optimization begins. For parameter-efficient SFT, we freeze all parameters from the base GPT-2 model besides the top two transformer blocks and the final layer normalization layer. Validation and early stopping is based on the held-out dev corpus chrF. Inference decoding uses temperature scaling, nucleus sampling (top-p), and a sliding-window repetition penalty.

5 Experiments

5.1 Data

We evaluate the GRPO-Rank algorithm on the Shakespearean sonnet generation task. We use a corpus of 155 Shakespearean sonnets split into 131 training sonnets (sonnets 1–131, full text), 12 development sonnets (sonnets 132–143), and 12 test sonnets (sonnets 145–156). At inference time, the model conditions on the first three lines of a held-out sonnet and autoregressively generates the remaining lines. Generated completions are evaluated against the ground-truth full sonnets. The Oracle provides ranking feedback based on adherence to the ABAB CDCD EFEF GG rhyme scheme.

5.2 Training and evaluation methods

For sonnet generation, we evaluate using the chrF metric [8], a character-level n-gram comparison that captures stylistic proximity to the ground-truth Shakespeare without requiring exact lexical matches. We use the complete sonnets for chrF calculation rather than excluding the first three lines. Additionally, we perform qualitative analysis of generated sonnets to assess coherence, meter adherence, and rhyme-scheme correctness beyond what chrF captures.

Training runs for up to $S = 30$ steps, saving checkpoints and evaluating on the dev set every 10 steps, where in each step a batch of $B = 4$ prompts is selected via random sampling with replacement, meaning each training sonnet is used approximately 1 time on average. For every prompt in a batch, the policy generates a group of G independent candidate completions (rollouts) using top-p sampling, resulting in a total of $S \cdot B \cdot G$ total completions across each training run. Each group of candidates is evaluated by a rank Oracle (Gemini-2.5-Flash), and the Oracle’s ranking is compared against the model’s internal log-probabilities using nDCG metric. Finally, the model optimizes the GRPO-Rank objective to update the policy.

During GRPO-Rank training, we periodically evaluate the average chrF of sonnets generated by the policy from the first three lines of the dev set against the full sonnets in the dev set. The corpus chrF on the dev set is used as a metric to evaluate the policy’s performance. Note that the average sentence chrF is noisier on small sets like the development sets, so we prefer corpus chrF. During inference, we generate full sonnets using the held-out first three lines of the test set.

5.3 Experimental details

We constrain our hardware and hyperparameter configurations to be strictly feasible for limited-compute academic setups. Operating without a parallel value model, as enabled by our GRPO-Rank formulation, is critical to fitting the training state into memory while maintaining a sufficiently large batch size.

- **Architecture:** Decoder-only GPT-2 base model (117M parameters)
- **Hardware:** $1 \times$ NVIDIA GeForce RTX 2080 (11 GB VRAM)
- **Oracle model:** Gemini 2.5 Flash
- **Prompt batch size (B):** 4

5.4 Results

We evaluated GRPO-Rank on group sizes 3, 4, 5, 8, 16, and 32, running each up to 30 iterations based on computational budget. Results for training dynamics and dev set performance are shown in the below figures.

Dev chrF comparison. Figure 1 reports the dev corpus chrF for the best checkpoint of each configuration, compared against the TA baseline and our SFT baseline. Our SFT model (chrF = 41.50) slightly outperforms the TA baseline (41.40). Among the GRPO-Rank variants, $G=8$ achieves the highest chrF of **42.19**, surpassing both baselines and demonstrating that rank-aware RL fine-tuning can improve over SFT for sonnet generation. Mid-range group sizes $G=5$ (40.75) and $G=4$ (36.21) show moderate performance, while the extremes: $G=3$ (33.98), $G=16$ (33.28), and $G=32$ (32.80) degrade substantially below the SFT baseline.

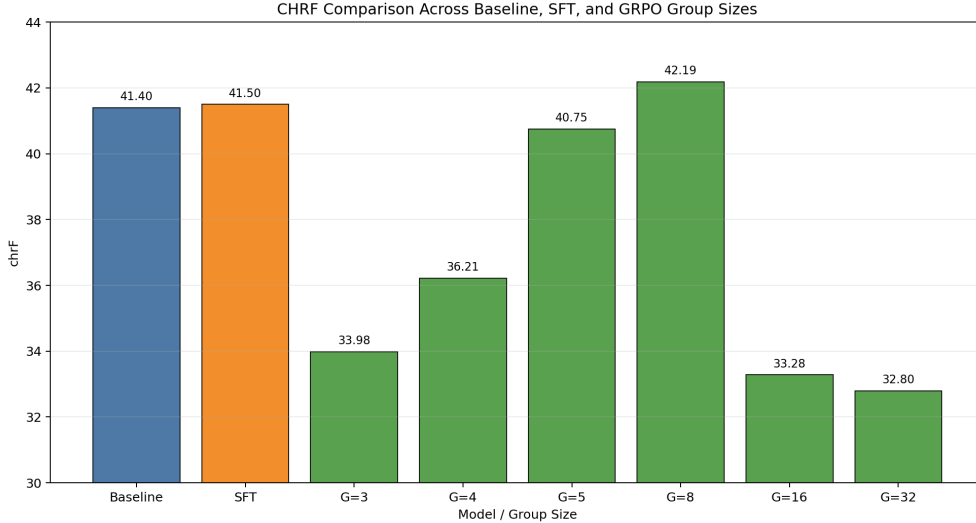


Figure 1: Dev corpus chrF across baselines and GRPO-Rank group sizes. $G=8$ achieves the best performance at 42.19, exceeding both the TA baseline (41.40) and our SFT checkpoint (41.50). Extreme group sizes ($G=3$, $G=16$, $G=32$) degrade below SFT.

Training dynamics. Figure 2 shows key training metrics over the first 25 steps for $G \in \{3, 4, 5\}$. The GRPO-Rank loss exhibits high variance but trends downward for all group sizes, with $G=3$ achieving the lowest per-step loss due to more stable Oracle rankings over fewer candidates. The average exact KL divergence from the SFT reference decreases from approximately 0.41 to 0.32 across training, indicating that the KL penalty successfully prevents the policy from diverging catastrophically from the pretrained checkpoint. Meanwhile, average entropy increases slightly (from ~ 4.4 to ~ 4.8), suggesting the policy explores a wider token distribution as training progresses, despite the entropy coefficient being set to zero.

The average clip fraction remains high throughout training (~ 0.66 – 0.75), indicating that a large proportion of token-level importance ratios fall outside the clipping window at each step. This is expected given the relatively aggressive policy updates characteristic of rank-based advantages and the small number of training steps.

6 Analysis

6.1 Group Size Ablation

The chrF results in Figure 1 reveal a non-monotonic relationship between group size and downstream performance. We attribute this to two competing forces that must be balanced when selecting G .

Ranking stability vs. signal strength. With small groups ($G=3$), the Oracle’s ranking task is easier (comparing three candidates yields more consistent orderings) but the nDCG-based advantage computed from only three items provides a weak gradient signal. Conversely, large groups ($G=16$, $G=32$) produce richer ranking information in principle, but the Oracle must rank many similar candidates, introducing noise that propagates into the advantage estimates.

Advantage variance analysis. The standard deviation of the advantage function provides direct evidence for this trade-off. Across our experiments, the mean standard deviation of the rank-aware advantage increases sharply with group size: $\sigma(\hat{A}_{\text{rank}}) = 1.55$ for $G=3$, 2.13 for $G=4$, and 7.97 for $G=16$. This approximately five-fold increase from $G=3$ to $G=16$ reflects the wider spread of nDCG penalties δ_j within larger groups. The advantage $\hat{A}_{\text{rank},i} = \mathbb{E}[\delta_j] - \delta_i$ naturally exhibits higher variance when the group contains candidates spanning a broader range of Oracle-assessed quality. At moderate group sizes ($G=5$ – 8), this variance is large enough to produce meaningful gradient updates but small enough to keep training stable, explaining why $G=8$ achieves the peak chrF of 42.19.

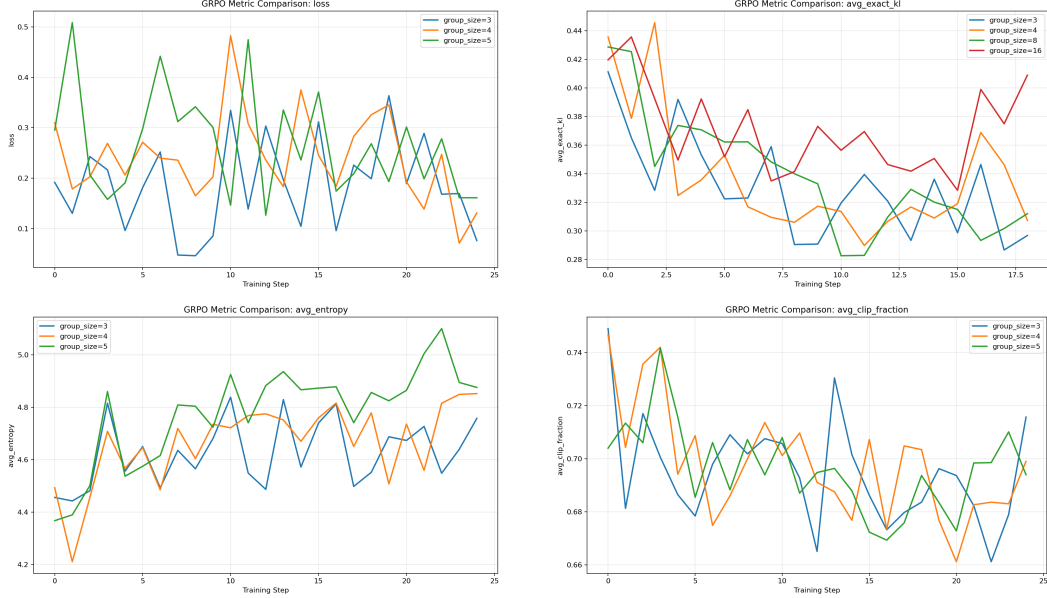


Figure 2: Training metrics over 25 steps for group sizes $G \in \{3, 4, 5\}$. Top-left: GRPO-Rank loss. Top-right: average exact KL divergence from the SFT reference. Bottom-left: average token entropy. Bottom-right: average clip fraction.

6.2 Wall-Time Cost

Figure 3 shows the per-step wall time for $G \in \{3, 4, 5\}$. Increasing group size raises the wall time roughly proportionally, since each additional candidate requires a forward pass for generation and an Oracle API call for ranking. Group size 3 averages approximately 85 seconds per step, group size 4 approximately 105 seconds, and group size 5 approximately 170 seconds. With improved batching over candidates and parallel Oracle API calls, much of this overhead could be amortized.

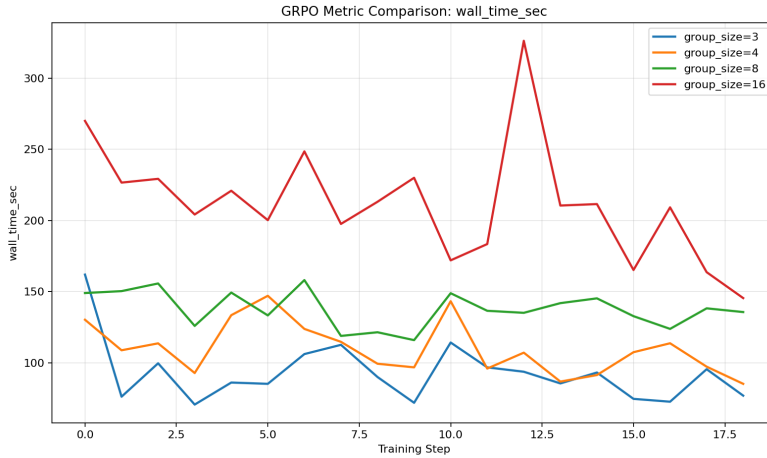


Figure 3: Per-step wall time (seconds) for $G \in \{3, 4, 5\}$. Larger groups incur proportionally higher cost due to additional rollouts and Oracle API calls.

6.3 KL Divergence and Policy Drift

The downward trend in KL divergence (Figure 2, top-right) warrants careful interpretation. Rather than indicating the policy is converging back toward the SFT reference, the decreasing KL likely

reflects the policy settling into a narrower region of output space: one that happens to overlap more with the reference distribution on average, even as the policy’s behavior on specific prompts may differ substantially. The KL penalty plays a critical role: without it, we observed in preliminary experiments that the policy quickly degenerated into repetitive or incoherent outputs, losing the sonnet structure learned during SFT.

6.4 Memory Efficiency

A key practical advantage of GRPO-Rank is its memory efficiency. Standard PPO requires maintaining both a policy network and a critic network of comparable size in GPU memory, effectively doubling the parameter footprint. GRPO-Rank eliminates the critic entirely by using group-relative normalization, and further removes the reward model by outsourcing evaluation to the Oracle API. On our single RTX 2080 (11 GB), this made the difference between feasibility and infeasibility: a PPO setup with a GPT-2 policy and a GPT-2-sized critic would leave insufficient memory for the batch of rollouts needed at each step.

7 Conclusion

We presented GRPOET-Rank, an adaptation of the Oracle-RLAIF framework to pure-text autoregressive generation, using rank-aware Group Relative Policy Optimization to align a GPT-2 model for Shakespearean sonnet generation. By replacing scalar reward models with ordinal ranking feedback from an external Oracle (Gemini 2.5 Flash), we eliminated both the critic network and the reward model, making RL-based post-training feasible on a single consumer GPU.

Our group size ablation revealed that GRPO-Rank’s effectiveness is highly sensitive to the number of candidates per prompt. A group size of $G=8$ achieved the best dev chrF of 42.19, surpassing both the TA baseline (41.40) and our SFT checkpoint (41.50), while extreme group sizes degraded performance substantially. We traced this to the standard deviation of the rank-aware advantage, which grows roughly five-fold from $G=3$ to $G=16$, creating a tension between gradient signal strength and update stability.

Several limitations remain. Our evaluation is restricted to dev set chrF without held-out test evaluation, and chrF captures only surface-level character overlap rather than semantic coherence, meter, or true rhyme-scheme adherence. The Oracle API introduces latency and cost that scale linearly with group size, and Oracle ranking consistency may degrade for large groups of similar-quality candidates. Finally, our experiments are limited to a small corpus of 131 training sonnets.

Future directions include further hyperparameter tuning for GRPO-Rank, such as group size, prompt batch size, and number of iterations. In particular, scaling up the base model size and exploring stronger oracle models are promising, as well as exploring the effects of KL divergence and entropy and the effects of different decoding strategies for sonnet generation. As an alternate baseline, DPO with an AI judge could be explored. For implementation, our implementation could be better parallelized over the batch size and group size to amortize the cost of Oracle API calls.

Team contributions (Required for multi-person team)

Eric implemented the optimizer, transformer matrix multiplication, the GRPO-Rank algorithm, and ran the experiments. Eric also wrote the abstract, introduction, related works, methods, and future work sections of the milestone/report. Also compiled the numbers and created the figures used in the final report, and provided discussion paragraphs on the chrF, σ_{adv} and wall clock efficiency.

William did the primary paper analysis in the proposal and implemented the transformer layer, embeddings, and sonnet forward pass. William also implemented the classifier and paraphrase detection forward pass. Later, William performed experiments for group size on the sonnet generation. William also performed writing and editing for the final report and poster.

Jamin implemented, improved, and debugged the paraphrase detection, sonnet generation SFT and GRPO-Rank, data processing, and experimental framework. Additionally, he ran experiments on sentiment classification as well as SFT and GRPO-Rank for sonnet generation. Jamin also worked on the experiments and analysis sections of the milestone and report.

References

- [1] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.
- [2] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.
- [3] Derek Shi, Ruben Glatt, Christine Klymko, Shubham Mohole, Hongjun Choi, Shashank Kushwaha, Sam Sakla, and Felipe Leno da Silva. Oracle-rlaif: An improved fine-tuning framework for multi-modal video models through reinforcement learning from ranking feedback. *arXiv preprint arXiv:2510.02561*, 2025.
- [4] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [5] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.
- [6] Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*, 2023.
- [7] 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. *Advances in Neural Information Processing Systems*, 36, 2023.
- [8] Maja Popović. chrF: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395, 2015.