RLPrompt with GRPO and Loglikelihood Rewards

Joshua Bello[1]
[1]MIT,
{jobello}@mit.edu

Abstract

We extend the RLPrompt framework by comparing Soft Q-Learning and Group Relative Policy Optimization (GRPO) for discrete prompt optimization using gap-based and log-likelihood rewards. Using DistilGPT-2 on BoolQ and GSM8K, we find that both methods yield modest improvements over the baseline on BoolQ, while GRPO exhibits substantially higher-variance training dynamics than Soft Q-Learning. On GSM8K, neither method improves performance and GRPO further degrades accuracy. These results suggest that RL-based prompt optimization is effective for shallow classification with small models, but is fundamentally limited by model capacity on multi-step reasoning tasks.

1 Introduction

Prompts are sequences of text prepended to an input to steer a language model toward more accurate or task-specific outputs. Although large language models (LLMs) can perform many tasks, they are pretrained only for next-token prediction, and improving performance on a new task traditionally requires fine-tuning—an expensive process that modifies model parameters. However, Language Models are Few-Shot Learners demonstrated that LLMs can achieve strong performance using only well-crafted prompts and a few in-context examples. This finding elevated the importance of prompt design as a lightweight alternative to fine-tuning and motivated two major families of systematic prompt optimization methods: discrete prompt optimization and soft prompt optimization.

1.1 Soft Prompt Optimization

In soft prompt optimization, learnable continuous embedding vectors are prepended to the model's input. Because these vectors lie in the same embedding space as token embeddings, they can be directly optimized using gradient descent and backpropagation. However, soft prompts have two notable limitations. First, they are not interpretable—the learned embeddings do not correspond to human-readable text, making their behavior opaque. Second, they are not transferable across language models.

1.2 Discrete Prompt Optimization

The second class of systematic prompt optimization methods, discrete prompt optimization, addresses the interpretability limitations of soft prompts by learning actual text tokens. Because tokens are non-differentiable, they cannot be optimized with backpropagation, and early approaches relied on heuristic search methods such as enumeration and selection. These methods are interpretable but computationally inefficient.

RLPrompt (Deng et al., 2022) introduced a reinforcement-learning framework for discrete prompt optimization, achieving strong gains on few-shot classification and style transfer tasks compared to fine-tuning and manual prompt design.

In this paper, I extend the RLPrompt framework by comparing Soft Q-Learning (Guo et al., 2022) with Group Relative Policy Optimization, introducing alternative reward functions, and evaluating on datasets such as BoolQ (Clark et al., 2019) and GSM8K (Cobbe et al., 2021) rather than the binary classification tasks used in the original work.

2 Related Works

In this section, I summarize the key components of RLPrompt's classification framework that are relevant to my experiments. RLPrompt frames prompt generation as a reinforcement learning (RL) problem, where

- **State:** The partially generated prompt $z_{<t}$ at time step t.
- **Action:** Selection of the next prompt token z_t.

- **Policy:** A trainable MLP that maps the frozen LLM's hidden states to token logits. In RLPrompt, the policy network is a two-layer MLP with a 2048-dimensional hidden layer.
- **Ensvironment:** A frozen Language Model (DistilGPT-2 HuggingFace (2019) in the original paper), which produces the hidden states used by the policy to choose the next token.
- **Reward:** Computed by a separate frozen classifier model (DistilRoBERTa-base Liu et al. (2019) in RLPrompt), which evaluates the completed prompt on the downstream classification task. RLPrompt uses the *gap reward*, discussed in detail later, to quantify the performance of the generated prompt.

In the paper, the authors formally describe the optimization problem as one in which the agent generates a prompt $z = [z_1, \ldots, z_T]$ one token at a time to maximize a downstream reward. At step t, the agent observes the partial prompt $z_{<t}$ and samples the next token z_t from the policy $\pi(z_t \mid z_{<t})$. After producing a complete prompt $\hat{z}$, it receives the task reward $R(y_{\text{LM}}(\hat{z}, x))$. Parameterizing the policy by θ, the optimization objective is:

$$\max_{\theta} \mathbb{E}_{\hat{z} \sim \prod_{t=1}^{T} \pi_\theta(z_t \mid z_{<t})} \left[R\big(y_{\text{LM}}(\hat{z}, x)\big) \right].$$

The RL algorithm used in RLPrompt is a simplified variant of Soft Q-Learning, which I briefly describe below.

2.1 Gap Reward

To avoid reward hacking, the authors did not use the raw probability of the correct class. Instead, RLPrompt adopts the gap reward. For a prompt z and example (x, c), the reward is defined as the difference between the probability of the correct label and the highest probability among incorrect labels. Using $P_z(c) := P_{\text{LM}}(c \mid z, x)$, the gap is $\text{Gap}_z(c) = P_z(c) - \max_{c' \neq c} P_z(c')$. This value is positive when the model predicts the correct label. Let $\text{Correct} := \mathbf{1}\left[\text{Gap}_z(c) > 0\right]$. RLPrompt scales positive gaps to emphasize correct predictions, yielding the reward

$$R(x, c) = \lambda_1^{1-\text{Correct}} \lambda_2^{\text{Correct}} \text{Gap}_z(c).$$

2.2 Soft Q Learning

Soft Q-Learning provides an RL formulation for language models by exploiting the one-to-one correspondence between LM logits $f_\theta(a \mid s)$ and soft Q-values $Q(s, a)$. This interpretation allows the algorithm to treat the entire vocabulary as the action space and update all Q-values in parallel via the softmax, rather than only the sampled action as in standard Q-learning. The resulting optimal policy is the Boltzmann (maximum-entropy) policy,

$$\pi(a \mid s) = \frac{\exp(Q(s, a))}{\sum_{a'} \exp(Q(s, a'))}.$$

RLPrompt uses only the on-policy component of Soft Q-Learning, treating prompt generation as a single-step MDP and reducing the learning objective to:

$$L_{\text{soft-Q}} = \left(Q_\theta(z) - \left(R + \gamma \alpha \log \sum_{a'} \exp\left(\tfrac{Q_\theta(s', a')}{\alpha} \right) \right) \right)^2.$$

where $Q_\theta(z)$ is the predicted soft Q-value of the prompt, α is the temperature parameter, γ is the discount factor, and R is the reward.

3 Methods

This section outlines the experimental framework used to extend RLPrompt, including the model architecture, datasets, reward formulations, andreinforcement learning objectives.

3.1 Prompt Policy Architecture

Following RLPrompt, the prompt policy consists of a frozen language model and a trainable two-layer MLP. The LLM provides last-layer hidden states, which the MLP transforms into modified embeddings that are passed through the LM head to produce token logits. Thus, the policy maps hidden states h_t to action distributions, with only the MLP parameters updated during training.

3.2 Datasets

We evaluate our method on two datasets beyond those used in the original RLPrompt work:

- **BoolQ**: A true/false question-answering dataset.
- **GSM8K**: A grade-school math reasoning dataset in which each example contains a multi-step natural-language solution ending with a numerical answer.

We sample 16 examples from each class for each dataset, 32 samples for BoolQ and 16 samples for GSM8K.

3.3 Reward Functions

We consider two reward formulations:

- **Gap Reward**: Following RLPrompt, the gap reward measures the difference between the probability of the correct label and the highest probability among incorrect labels. For GSM8K, there is no natural notion of an "incorrect class" because answers are numerical. To approximate the gap reward, we generate 4 plausible but incorrect numeric answers and compute the maximum probability among them. The probability is only over the answer, not the reasoning before the answer.

- **Log-Likelihood (LL)**: We also evaluate a reward based on the log-likelihood of the correct output. For BoolQ, this corresponds to the log-likelihood of the model generating the tokens "`yes`" or "`no`". For GSM8K, the log-likelihood is computed over the entire sequence of tokens in the chain-of-thought answer.

Instead of using a separate frozen classifier (e.g., DistilRoBERTa-base as in RLPrompt), we compute both rewards using the same frozen LLM and its LM head.

3.4 Reinforcement Learning Algorithms

We compare two RL objectives for prompt optimization:

- **Soft Q-Learning.**: We adopt the simplified Soft Q-Learning objective used in RLPrompt, in which LM logits are interpreted as soft Q-values and prompt generation is treated as a single-step MDP.

- **Group Relative Policy Optimization (GRPO) Shao et al. (2024)**: As an alternative, we evaluate GRPO, which stabilizes training by normalizing rewards within each batch. Given a batch of rewards R_i, the advantage is computed as

$$A_i = R_i - \frac{1}{B}\sum_{j=1}^{B} R_j.$$

  The policy is then updated with a KL-regularized objective similar to PPO, but applied at the prompt level.

3.5 Training Procedure

For each dataset, we repeatedly sample 16 prompt candidates, compute rewards, update the policy network using the chosen RL objective, and evaluate on a held-out validation set every 10 steps as shown in Figure 2 in the Appendix. We train for a total of 1,000 steps. The final prompt used for testing is constructed sequentially, greedily selecting each token.

4 Results

4.1 Training

During training, Figure 1 in the Appendix shows that GRPO updates the policy in a significantly noisier manner than Soft Q-learning. In GRPO, the policy gradient is scaled by a signed advantage that depends on whether a sampled reward is above or below the batch mean. As a result, gradient updates can be either positive or negative, leading to higher-variance and less stable parameter updates. In contrast, Soft Q-learning applies uniformly positive weighting through the exponentiated Q-values, yielding smoother and more consistent optimization behavior.

An additional source of instability is that GRPO relies on Monte Carlo estimates, which are inherently higher variance than temporal-difference (TD) methods, the category to which Soft Q-learning belongs.

4.2 Prompts

Table 1 shows the greedy-decoded prompts extracted after training for each dataset and learning configuration. Although the learned tokens are individually interpretable, the resulting prompt strings don't form meaningful English phrases, consistent with the findings of RLPrompt that prompt optimization need not follow human language patterns. (Deng et al., 2022)

4.3 BoolQ and GSM8K

Tables 2 and 3 summarize the evaluation results of RL-based prompt optimization on BoolQ and GSM8K. On BoolQ, all RL variants yield modest but consistent gains over the baseline: GRPO with log-likelihood reward reaches 0.6222 accuracy (vs. 0.5723 for the baseline), with the other GRPO and SoftQ configurations clustered in the 0.608–0.619 range. The gap between SoftQ and GRPO, and between gap-based and log-likelihood rewards, is relatively small, suggesting that, for this

Table 1: Learned prompts across datasets using DistilGPT-2.

Dataset	Algo	Reward	Prompt Text
BoolQ	SoftQ	Gap	ridorridorridorridorridor
BoolQ	GRPO	Gap	Roberts Klu sloganssupRoberts
BoolQ	SoftQ	LL	disgusted disgusted weary disgusted disappointed
BoolQ	GRPO	LL	alikelease alikeGMTuve
GSM8K	SoftQ	Gap	ConsideringConsideringConsideringConsideringConsidering
GSM8K	GRPO	Gap	080iversessesountoids
GSM8K	SoftQ	LL	Particularly Especially Especially Especially Especially
GSM8K	GRPO	LL	illanceillanceillanceillanceillance

binary classification task, differences between RL algorithms and reward types are minor relative to the overall benefit of prompt optimization.

Table 2: BoolQ Accuracy Results

Algorithm	Reward	Prompt Len	Steps	Accuracy
Baseline	Baseline	0	0	0.5723
GRPO	LL	5	1000	0.6222
GRPO	GAP	5	1000	0.6192
SoftQ	LL	5	1000	0.6124
SoftQ	GAP	5	1000	0.6081

In contrast, on GSM8K all methods remain close to 0% accuracy. The baseline reaches 0.0030, SoftQ essentially matches this value, and GRPO reduces performance by roughly an order of magnitude. This pattern is consistent with the intuition that small models such as DistilGPT-2 (under 100M parameters) have enough capacity for prompt-based improvements on shallow classification tasks like BoolQ, where better prompts can activate knowledge the model already possesses. Multi-step mathematical reasoning, however, is fundamentally capacity-limited: GSM8K requires structured, multi-hop arithmetic and logical inference that these small models cannot reliably internalize through short-horizon prompt RL.

Table 3: GSM8K Accuracy Results

Algorithm	Reward	Prompt Len	Steps	Accuracy
Baseline	Baseline	0	0	0.003033
SoftQ	LL	5	1000	0.003033
SoftQ	GAP	5	1000	0.002274
GRPO	GAP	5	1000	0.000758
GRPO	LL	5	1000	0.000758

The degradation observed with GRPO likely arises from its use of relative advantages. When the model performs uniformly poorly early in training, GRPO still assigns positive advantages to some generations merely because they are less bad than the others. This can cause GRPO to reinforce systematically flawed reasoning trajectories, amplifying undesirable patterns rather than correcting them. In such settings, the optimization dynamics may become unstable, and additional RL updates may push the model further away from meaningful reasoning behavior.

5 Conclusion

Our results show that Soft Q-learning provides smoother and more stable optimization than GRPO. On BoolQ, all RL-based prompt methods yield modest but consistent improvements over the baseline, with only minor differences across algorithms and reward types. In contrast, on GSM8K none of the methods achieve meaningful gains, and GRPO in particular degrades performance by reinforcing relatively better—but still incorrect—reasoning trajectories.

5.1 Future Work

A key limitation of this study is that all models were trained for only 1,000 steps, whereas the RLPrompt paper trains for over 10,000 steps. Future work could investigate whether longer training horizons lead to improved performance on harder reasoning tasks such as GSM8K. In addition, extending these methods to larger-capacity models, such as Llama 3 variants, which already exhibit baseline performance on GSM8K, may make prompt-based RL more effective and easier to optimize.

References

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman.

2021. Training verifiers to solve math word problems.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P. Xing, and Zhiting Hu. 2022. Rlprompt: Optimizing discrete text prompts with reinforcement learning.

Han Guo, Bowen Tan, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. 2022. Efficient (soft) q-learning for text generation with limited good data.

HuggingFace. 2019. Distilgpt-2. https://huggingface.co/distilgpt2. Accessed: 2025-02-10.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

A Appendix

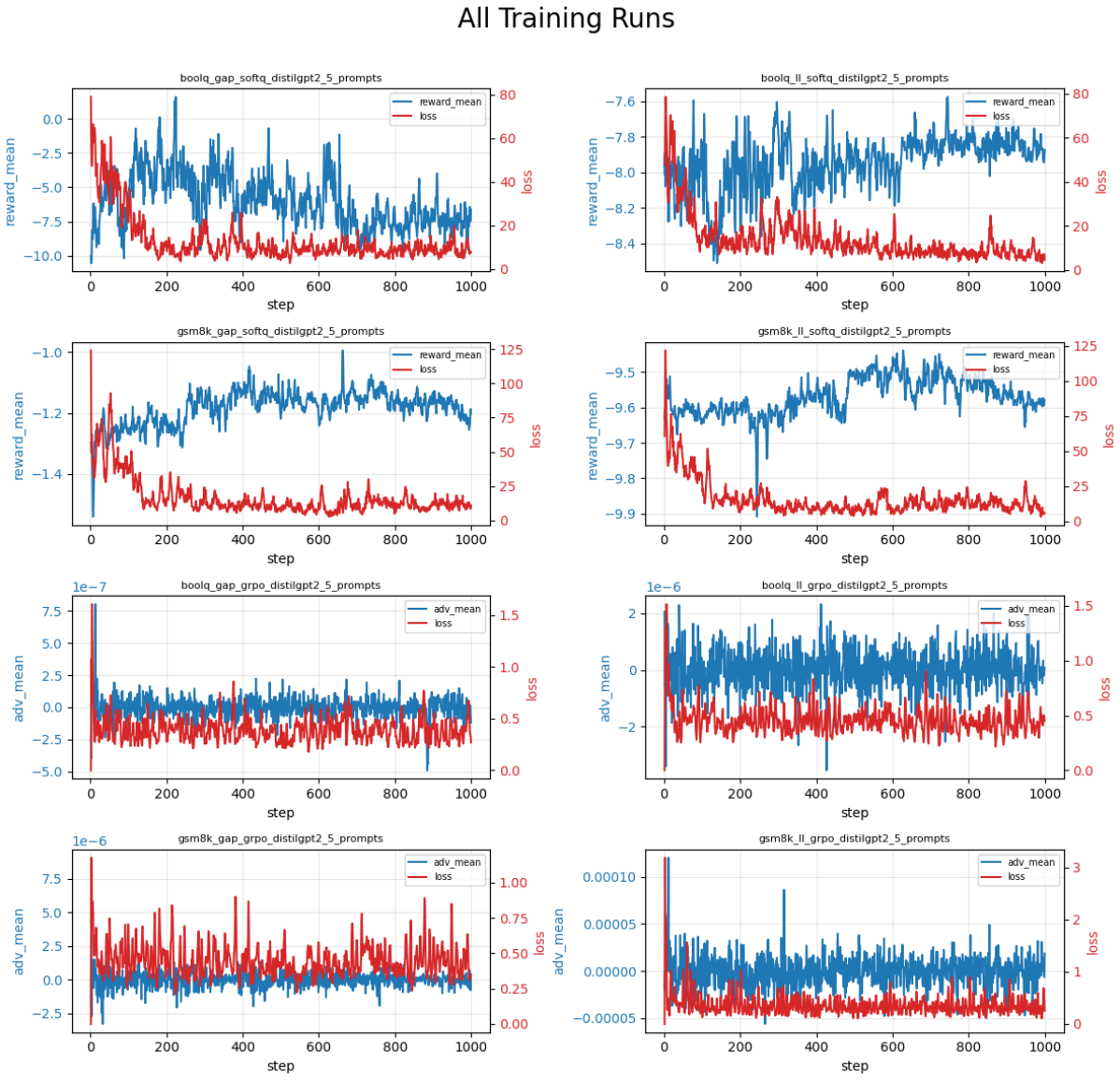

Figure 1: Plot of loss for all training runs over 1000 steps, along with either the mean reward or mean advantage

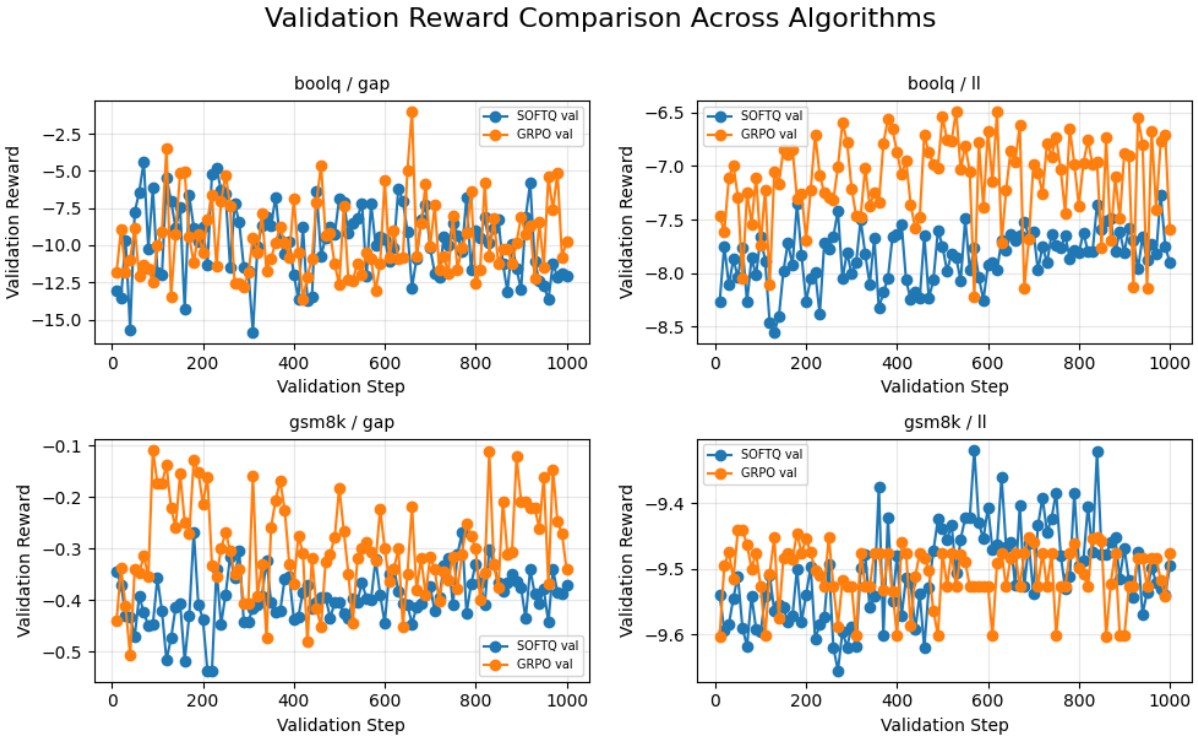

Figure 2: Plots comparing validation reward between GRPO and Soft Q learning across all reward and learning policy combinations