Rethinking Multiple-Choice Questions for RLVR: Unlocking Potential via Distractor Design

Xu Guo[*1,2,3] **Qiming Ge**[*1,3] **Jian Tong**[3] **Kedi Chen**[2,3]
Jin Zhang[1,3] **Xiaogui Yang**[3] **Xuan Gao**[1,3] **Haijun Lv**[3]
Zhihui Lu[†1] **Yicheng Zou**[3] **Qipeng Guo**[†2,3]

[1]Fudan University [2]Shanghai Innovation Institute [3]Shanghai AI Laboratory

{geqiming, tongjian, chenkedi, zhangjin, yangxiaogui,
gaoxuan, lvhaijun, zouyicheng, guoqipeng}@pjlab.org.cn
guox24@m.fudan.edu.cn lzh@fudan.edu.cn

[*]Equal contribution [†]Corresponding authors

Abstract

Reinforcement Learning with Verifiable Rewards (RLVR) significantly enhances the reasoning capabilities of Large Language Models. When applied to RLVR, Multiple-Choice Questions (MCQs) offer a scalable source of verifiable data but risk inducing reward hacking, where models shortcut reasoning via random guessing or simple elimination. Current approaches often mitigate this by converting MCQs to open-ended formats, thereby discarding the contrastive signal provided by expert-designed distractors. In this work, we systematically investigate the impact of option design on RLVR. Our analysis highlights two primary insights: (1) Mismatches in option counts between training and testing degrade performance. (2) Strong distractors effectively mitigate random guessing, enabling effective RLVR training even with 2-way questions. Motivated by these findings, we propose **Iterative Distractor Curation** (IDC), a framework that actively constructs high-quality distractors to block elimination shortcuts and promote deep reasoning. Experiments on various benchmarks demonstrate that our method effectively enhances distractor quality and yields significant gains in RLVR training compared to the original data.

1 Introduction

Reinforcement Learning with Verifiable Rewards (RLVR) has recently improved large language models (LLM) reasoning capabilities (Lambert et al., 2025; Shao et al., 2024; Chen et al., 2025). Unlike RLHF (Ouyang et al., 2022), RLVR uses automatically verifiable rewards. The performance of RLVR depends on the accuracy of the reward signal: an LLM samples diverse outputs, receives contrastive feedback, and updates its policy accordingly. When the task outputs admit clear correctness criteria (for example, when answers can be uniquely verified or logical reasoning can be checked via formal verification), RLVR can effectively improve model performance.

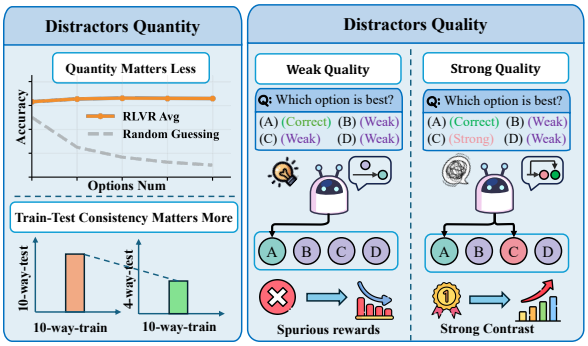


Figure 1: The impact of distractor design on RLVR.

However, not all tasks provide reliable verification. Recent work has explored Multiple-Choice Questions (MCQs) as verifiable reward signals to enable automated training across a broader range of tasks (Liu et al., 2025a; Zhang et al., 2025a; Akter et al., 2025; Guo et al., 2025). For example, AlphaMed (Liu et al., 2025a) performs RLVR with medical MCQs and obtains performance gains. Multiple-choice questions are typically considered verifiable, as their correctness can be objectively determined by comparing against the reference answer. Nevertheless, Med-RLVR (Zhang et al., 2025a) finds that models can hack rewards (Amodei et al., 2016) by shortcutting reasoning to output answers directly, highlighting the risks of using MCQs for verification. The accuracy of reward signals in MCQ-based RLVR is inherently challenged by two factors: (1) **Random guessing**: due to limited answer space, models can guess the correct answer even for unsolved ques-

Findings of the Association for Computational Linguistics: ACL 2026, pages 20092–20113

tions; and (2) **Elimination strategies**: models can exploit weak distractors (i.e., options that are obviously incorrect) to identify correct answers via elimination rather than reasoning.

To address these issues, Kimi k1.5 (Team et al., 2025a) and Seed-Thinking-v1.5 (Guo et al., 2025) either discard multiple-choice questions or convert them into open-ended formats. Similarly, Nemotron-CrossThink (Akter et al., 2025) selectively converts suitable MCQs into QA pairs, filtering out those that depend on the provided options (e.g., "Which of the following..."). These studies attribute the limitations of MCQ-based RLVR to the closed-ended format: the presence of options increases the probability of sampling correct answers by chance. To circumvent this, they transform MCQs into open-ended formats rather than training on the original questions. However, this approach discards the benefits of MCQs: beyond the correct option, MCQs contain carefully designed distractors by domain experts, providing natural contrastive signals for learning. Effective RLVR training should not only reward correct answers but also elicit and penalize incorrect outputs. Distractors can guide the model to actively distinguish among easily confusable concepts, exposing its misconceptions. To this end, we pose two questions: (1) How does the design of multiple-choice options affect RLVR training? (2) How can we fully exploit multiple-choice data for RLVR while preserving the format?

To address the first question, we empirically examine how candidate options affect RLVR training. By keeping the question stem constant and varying the quantity and quality of distractors, we systematically study the effect of option design. We evaluate this by measuring the model's downstream performance, examining its robustness across varying option counts and its accuracy on standard benchmarks. Through the analyses, we obtain two main findings: (1) The impact of the number of options is overestimated. Although increasing options can reduce the random guess rate (25% for 4-way vs. 10% for 10-way questions), the effect on RLVR training is limited. Moreover, we observe a distributional mismatch in option counts between training and testing: inconsistency in option counts between training and test sets degrades RLVR performance. This suggests that in multiple-choice RLVR, the model learns patterns tied to the distribution of option counts, leading to an option-count distribution bias. (2) Differences in distractor quality significantly influence the effectiveness of RLVR training. Even in 2-way questions (with a 50% chance of guessing correctly), high-quality distractors can make RLVR training effective.

For the second question, motivated by these findings, we propose **Iterative Distractor Curation (IDC)** to enhance distractor quality. IDC iteratively refines distractors to eliminate superficial shortcuts, compelling the model to engage in genuine reasoning rather than exploiting random guessing or trivial elimination. Remarkably, we find that models can self-generate these challenging options to enhance learning, effectively enabling autonomous self-improvement without any external supervision. We conduct experiments on four medical datasets, including MMLU-Pro Health, MedXpertQA, PubMedQA, and SuperGPQA Clinical Medicine, to validate the effectiveness of our method.

Overall, our contributions are as follows:

- We show that mismatches in option count distributions between training and testing lead to performance degradation in RLVR.
- We analyze how option design affects RLVR and demonstrate that high-quality distractors are key to effective MCQ-based RLVR.
- We propose Iterative Distractor Curation (IDC) to improve MCQ distractors for RLVR training. On Qwen2-7B and Llama-3.1-8B, IDC consistently improves average accuracy: from 39.39% to 42.62% (+3.23%) and from 48.63% to 51.90% (+3.27%), respectively.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 introduces preliminaries. Section 4 analyzes option quantity and validates count alignment. Section 5 investigates distractor quality and proposes IDC. Finally, Section 6 concludes the paper.

2 Related Work

Reinforcement Learning with Verifiable Rewards (RLVR). RLVR (Lambert et al., 2025) reduces the reliance on human preference annotation in traditional RLHF (Ouyang et al., 2022) by leveraging verifiable reward signals. DeepSeek-R1 (DeepSeek-AI et al., 2025; Shao et al., 2024) shows that large-scale RLVR can substantially improve LLM reasoning. Under this paradigm, the model samples both correct and incorrect outputs and optimizes its policy based on precise reward

feedback. Subsequent work extends RLVR along multiple directions (Zheng et al., 2025; Cui et al., 2025; Xie et al., 2025; Zeng et al., 2025; Hu et al., 2025; Luo et al., 2025c,b,a; Liu et al., 2025c), but these efforts have largely focused on mathematics and coding domains, limiting their applicability to more general question-answering tasks.

To extend RLVR, recent attempts have started to utilize multiple-choice questions (MCQs) (Liu et al., 2025a; Zhang et al., 2025a; Akter et al., 2025; Guo et al., 2025; Team et al., 2025a). The community remains divided on MCQ effectiveness: while MCQs appear to satisfy the verifiability requirement, models can guess or eliminate incorrect options to arrive at the answer, leading to unreliable reward signals. Seed-Thinking-v1.5 (Guo et al., 2025) and Kimi k1.5 (Team et al., 2025a) convert MCQs to open-ended questions or discard them, believing the MCQ format harms training. Others use MCQs directly but report limited success. While Liu et al. (2025a) directly apply RLVR on original MCQs, Nemotron-CrossThink (Akter et al., 2025) and Med-RLVR (Zhang et al., 2025a) observe modest gains, attributing it to restricted answer spaces. These approaches either abandon the verifiable MCQ format or lack systematic analysis of what affects MCQ-based RLVR effectiveness. In this paper, we retain the MCQ format and systematically investigate how distractor quantity and quality influence RLVR training.

Hard Negative Samples and Preference Learning. Hard negatives are foundational in contrastive learning (Gao et al., 2022). For LLM alignment, negative signals are crucial for preference learning methods like RLHF (Ouyang et al., 2022) and DPO (Rafailov et al., 2024). Recent advances in RLVR, such as GRPO (Shao et al., 2024) and DAPO (Yu et al., 2025), further leverage negative samples for policy optimization. Across these methods, negative samples typically come from two sources: (1) incorrect outputs naturally generated during training, or (2) artificially corrupted versions of correct outputs (Zhang et al., 2025b). Yet the role of adversarial hard negatives—specifically, MCQ distractors—remains unexplored. Unlike negatives generated during training, distractors are explicit, linguistically crafted hard negatives that guide models toward specific misconceptions or erroneous reasoning paths. We demonstrate that well-designed distractors act as effective hard negatives that can substantially enhance RLVR training.

LLM-based Data Generation. LLM-based data synthesis has become mainstream (Wang et al., 2024a, 2025; Ding et al., 2024; Luo et al., 2025d), mostly focusing on generating questions (or instructions) (Wang et al., 2023; Xu et al., 2025; Bohnet et al., 2024). In contrast, distractor generation—creating options for fixed stems—has received less attention and primarily serves evaluation benchmarks (Zhang et al., 2025c) or educational assessment (Offerijns et al., 2020). Prior works evaluate distractors via extrinsic metrics such as n-gram scores (BLEU (Papineni et al., 2002), ROUGE (Lin, 2004)) and ranking metrics (Liang et al., 2018; Chiang et al., 2022; Yu et al., 2024; Ghanem and Fyshe, 2024; Qu et al., 2024). The AI in Education community has also explored distractor generation for educational assessment, including LLM-based approaches (Feng et al., 2024; Scarlatos et al., 2024; Fernandez et al., 2024) and studies of alignment between LLM and student error patterns (Sonkar et al., 2025; Liu et al., 2025b). However, the impact of distractors on LLM *training*, especially RLVR, remains unexplored. We bridge this gap by focusing on the intrinsic training value of option design, providing both empirical evidence on how distractors affect RLVR and insights for synthesizing training-effective MCQs.

3 Preliminaries

We begin by formalizing the RLVR framework for multiple-choice questions. Consider a multiple-choice question with stem q and n options $\mathcal{O} = \{o_1, \ldots, o_n\}$, where o^* denotes the correct option. The model π_θ generates a complete output $y = (c, a)$, where c is the reasoning chain and $a \in \mathcal{O}$ is the selected answer. The reward is defined as $r = \mathbb{I}[a = o^*]$. The goal of RLVR is to optimize the policy π_θ to maximize the expected reward over the data distribution $\mathcal{D}$:

$$\mathcal{J}(\theta) = \mathbb{E}_{(q,\mathcal{O})\sim\mathcal{D}, y\sim\pi_\theta(\cdot|q,\mathcal{O})}\left[r(y)\right].$$

For brevity, we omit the KL divergence term. However, optimizing this objective is non-trivial. The effectiveness of RLVR critically depends on the accuracy of the reward signal (Lambert et al., 2025; Huang et al., 2025), and prior work has shown that noisy rewards can severely degrade RL performance (Shao et al., 2025; Wu et al., 2025).

To analyze the reward mechanism in multiple-choice settings, we decompose the probability of

receiving a reward based on reasoning validity. We partition the space of reasoning chains into correct ($c \in \mathcal{C}_+$) and incorrect ($c \in \mathcal{C}_-$) sets. The probabilities of receiving a reward ($r = 1$) or no reward ($r = 0$) can be expressed as:

$$\begin{aligned}\Pr[r=1] &= \Pr[a = o^* \mid c \in \mathcal{C}_+] \Pr[c \in \mathcal{C}_+] \\ &\quad + \Pr[a = o^* \mid c \in \mathcal{C}_-] \Pr[c \in \mathcal{C}_-], \\ \Pr[r=0] &= \Pr[a \neq o^* \mid c \in \mathcal{C}_+] \Pr[c \in \mathcal{C}_+] \\ &\quad + \Pr[a \neq o^* \mid c \in \mathcal{C}_-] \Pr[c \in \mathcal{C}_-].\end{aligned}$$

The term $\Pr[a \neq o^* \mid c \in \mathcal{C}_+]$ represents the probability that valid reasoning leads to an incorrect selection, typically due to instruction-following failures or formatting errors; assigning zero reward in these instances is consistent with our optimization goals. However, the critical challenge arises from **spurious rewards**. This occurs when the model selects the correct answer despite flawed reasoning:

$$\Pr_{\text{spurious}} = \Pr[a = o^* \mid c \in \mathcal{C}_-] \Pr[c \in \mathcal{C}_-].$$

Such rewards provide misleading positive reinforcement, encouraging the model to maintain incorrect reasoning paths.

Unlike open-ended tasks where the output space is effectively unbounded, multiple-choice questions suffer from a finite output space where incorrect reasoning is more likely to accidentally yield the correct answer. Although converting MCQs to open-ended formats mitigates spurious rewards, it discards distractors that embody human priors and provide natural hard negative signals. Additionally, rewriting risks semantic alteration or hallucination. Thus, we retain the original MCQ structure by fixing the stem q and the correct answer o^*, while focusing on the distractor set $\mathcal{O}_-$ as the key control variable. We model the probability of selecting the correct answer under incorrect reasoning as a mixture of systematic preference and random guessing:

$$\Pr[a = o^* \mid c \in \mathcal{C}_-] = (1 - \lambda) s(\mathcal{O}_-) + \frac{\lambda}{n}.$$

where n is the total number of options, $\lambda \in [0, 1]$ represents the weight of random guessing, and $s(\mathcal{O}_-) \in [0, 1]$ captures the systematic preference of the model for o^* over distractors despite flawed reasoning. This formulation reveals two key factors that control spurious rewards:

1. **Number of Distractors.** The number of options determines the baseline probability of random guessing. Increasing the distractor count reduces the likelihood of accidental correctness under flawed reasoning (the λ-weighted term). This effectively mitigates spurious rewards arising from stochasticity.

2. **Distractor Strength.** We define strength as the semantic competitiveness of distractors. **Weak distractors** are easily eliminated, allowing the model to assign high probability to o^* even with flawed reasoning (increasing $s(\mathcal{O}_-)$). These samples provide minimal negative feedback, and the RL algorithm fails to obtain informative penalty signals from trajectories with flawed reasoning but correct outcomes. Conversely, **strong distractors** are deceptive and align with potential misconceptions. They compete for probability mass, reducing $s(\mathcal{O}_-)$ and serving as sinks for flawed reasoning. This ensures that incorrect logic leads to incorrect predictions ($r = 0$).

4 Impact of the Number of Distractors

4.1 Analysis Study

Experiment Setting. We use MMLU-Pro (Wang et al., 2024b) as the base dataset. To construct variants with different option counts, we keep the correct answer and randomly sample distractors, yielding five variants: 2-choice (`mcq2`), 4-choice (`mcq4`), 6-choice (`mcq6`), 8-choice (`mcq8`), and 10-choice (`mcq10`). We train Llama-3.1-8B-Instruct and Qwen2-7B-Instruct using GRPO (Shao et al., 2024) (see Appendix A for hyperparameters and Appendix B for data preprocessing).[1]

Metrics. For the complete results of our cross-evaluation (5 training $\times$ 5 testing settings), please refer to Table 7 in Appendix C. We define the **option-count gap** Δ as:

$$\Delta = m - n, \tag{1}$$

where m and n denote the number of options in training and testing sets, respectively. To account for the varying baseline difficulties of test sets with different option counts, we standardize performance to enable fair comparison across settings. Specifically, we compute the normalized score:

$$z_{m,n} = \frac{A_{m,n} - \mu_n}{\sigma_n}, \tag{2}$$

[1]We use Qwen2 instead of Qwen2.5 due to concerns about data contamination (Wu et al., 2025).

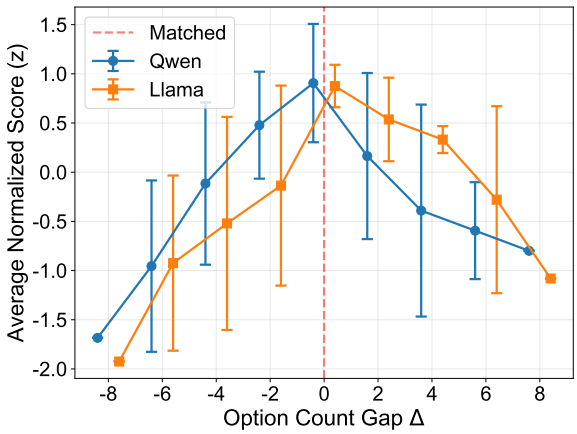


Figure 2: Relationship between average normalized z score and option-count gap Δ. Note that data points are slightly offset horizontally for clarity.

where $A_{m,n}$ denotes the accuracy of a model trained with m options and evaluated on n options. Here, μ_n and σ_n are the mean and standard deviation of accuracy over all training option counts $m \in \{2, 4, 6, 8, 10\}$ for the fixed n-way test set. Figure 2 illustrates the relationship between average normalized score z and option-count gap Δ.

Finding: Option-count mismatch affects RLVR performance. As shown in the full results (Table 7 in Appendix C), simply increasing training option count does not guarantee better performance. Instead, we find that the alignment between training and test option counts is critical. As shown in Figure 2, three observations emerge: (1) performance peaks at $\Delta = 0$, confirming the benefit of option-count alignment; (2) performance declines monotonically as $|\Delta|$ increases, showing that larger mismatches cause greater degradation; (3) The curve exhibits asymmetry where the region with fewer training options declines more steeply than that with more options. Consequently, when $|\Delta|$ is the same, having more training options ($\Delta > 0$) is generally preferable to having fewer ($\Delta < 0$), though the advantage is model-dependent. These findings indicate that matching training and test option counts is crucial. Larger mismatches lead to more severe performance degradation. A likely explanation is that the policy becomes overfitted to the specific number of choices during RLVR training. Consequently, when the option count changes at test time, the policy fails to generalize, leading to suboptimal decisions. This phenomenon persists in larger models (Appendix P), confirming it stems from a fundamental property of LLMs rather than insufficient model capacity.

Training Configuration	Llama-3.1-8B	Qwen2-7B
Train on 2-way	52.44	46.66
2→10 (Self)	56.12	50.71
2→10 (Qwen2.5-32B)	55.91	**51.01**
2→10 (GPT-OSS-120B)	56.26	50.88
Train on 4-way	56.19	48.57
4→10 (Self)	<u>56.72</u>	50.60
4→10 (Qwen2.5-32B)	56.63	50.59
4→10 (GPT-OSS-120B)	56.44	<u>51.00</u>
Train on 10-way	**58.61**	50.07

Table 1: Effect of option expansion on RLVR performance. All results are evaluated on the `mcq10` test set and reported as accuracy (%). **Bold** and <u>Underlined</u> indicate the best and second-best results.

4.2 Aligning Train-Test Formats Benefits RLVR Training

We propose that aligning the number of options between training and testing is critical for effective RLVR adaptation. We address potential mismatches as follows: if the training data has more options than the target test set, we randomly subsample the options to match the target count; if it has fewer, we augment the set by generating synthetic distractors. Specifically, the model is prompted to produce $N_{\text{test}} - N_{\text{train}}$ additional options given the question stem and existing choices. This strategy mitigates the inductive bias from mismatched formats and facilitates policy transfer.

We validate this approach on MMLU-Pro using Qwen2-7B and Llama-3.1-8B. We simulate mismatches by down-sampling original 10-choice questions to 2 or 4 choices, and then reconstructing them to 10 choices via option expansion (Table 1, detailed experimental setup and prompt are provided in Appendix D). We observe that reducing the option count gap consistently improves performance. Notably, self-generated distractors yield gains comparable to those from stronger models (e.g., GPT-OSS-120B with high reasoning effort). This result suggests that the improvement stems primarily from format alignment rather than knowledge distillation from stronger models. However, the effectiveness varies by model. While Qwen2-7B recovers its original performance, Llama-3.1-8B lags behind the human-written baseline. This indicates that adding more options is beneficial but insufficient, suggesting that distractor quality plays a pivotal role. In the next section, we investigate the impact of distractor strength.

Llama-3.1-8B Configuration	Acc	Qwen2-7B Configuration	Acc
mcq2-random	<u>52.44</u>	mcq2-random	46.66
w/o Llama-3.1-8B	49.36	w/o Qwen2-7B	39.87
w/ Llama-3.1-8B	**53.22**	w/ Qwen2-7B	**46.98**
w/ Qwen2-7B	52.25	w/ Llama-3.1-8B	<u>46.94</u>
w/ DS-V3.1	51.13	w/ DS-V3.1	46.41

Table 2: Impact of distractor strength on RLVR performance (%). We compare random distractors (mcq2-random), selecting the weakest distractor (**w/o**, lowest $\hat{s}_j$), or selecting the strongest distractor (**w/**, highest $\hat{s}_j$) as estimated by a reference model. DS-V3.1 denotes DeepSeek-V3.1. **Bold** and <u>Underlined</u> indicate the best and second-best results.

5 The Effect of Distractor Strength

5.1 Analysis Study

Based on the definition in Section 3, we hypothesize that *distractor strength* determines the quality of the reward signal. Strong distractors effectively compete with the ground truth o^*, providing high-contrast feedback, whereas weak distractors are trivial to reject even with flawed reasoning. To analyze the impact of distractor strength, we conduct ablation studies by reducing MMLU-Pro questions to a 2-choice format. For a given question, we select the single distractor based on its empirical strength $\hat{s}_j$, estimated by sampling trajectories from a reference model. Specifically, $\hat{s}_j$ is defined as the conditional frequency with which a distractor is chosen among all incorrect responses:

$$\hat{s}_j = \begin{cases} \frac{\sum_k \mathbb{I}[a_k = o_j]}{\sum_k \mathbb{I}[a_k \neq o^*]}, & \text{if } \sum_k \mathbb{I}[a_k \neq o^*] > 0 \\ 0, & \text{otherwise} \end{cases}$$

where $\{a_k\}_{k=1}^K$ are answers sampled from a reference policy. This metric quantifies distractor attractiveness, where a higher $\hat{s}_j$ indicates that o_j acts as a stronger trap for the model.

We analyze distractor strength by comparing three settings: (i) *random*, selecting a distractor uniformly at random; (ii) *strong* (**w/**), selecting the distractor with the highest $\hat{s}_j$; and (iii) *weak* (**w/o**), selecting the distractor with the lowest $\hat{s}_j$ (detailed setup in Appendix F). We experiment with Llama-3.1-8B and Qwen2-7B as policy models, using each model itself or external models (e.g., DeepSeek-V3.1) as the reference for estimating $\hat{s}_j$. Table 2 shows the experimental results.

Finding 1: Distractor strength is critical for RLVR training. We observe substantial performance differences across distractor configurations. For both Llama-3.1-8B and Qwen2-7B, selecting the weakest distractor leads to performance that is clearly inferior to the random baseline. Conversely, reintroducing model-based strong distractors restores or improves accuracy. These results demonstrate that sufficiently *strong* distractors are essential for learning. We attribute this to the informative preference signals they provide.

To further analyze this phenomenon, we visualize training dynamics in Figure 3. We observe that for 'mcq2 w/o' (where the weakest distractor is selected), the solve-all ratio (the proportion of questions where the model answers correctly in all sampled attempts) increases rapidly, indicating the model quickly learns to answer many questions correctly with high probability. In contrast, both 'mcq2 w/' and 'mcq10' remain challenging, maintaining a consistently lower solve-all ratio. Additionally, 'mcq2 w/o' results in shorter outputs, suggesting the model tends to skip detailed reasoning and provide answers more quickly, whereas the other two settings encourage longer responses. These patterns hold for both Qwen and Llama.

Finding 2: Self-targeted distractors are most effective. We find that distractors selected based on the target model itself yield the best results. For both Llama and Qwen, using self-estimated distractors consistently outperforms those derived from other models (e.g., DeepSeek-V3.1) or the random baseline. This means any model can serve as its own reference for distractor curation without relying on external models. Self-targeted distractors better align with the model's intrinsic error patterns and decision boundaries, providing more precise discriminative signals for optimization.

5.2 Iterative Distractor Curation

Building on the empirical findings, we propose the **Iterative Distractor Curation** framework to systematically enhance distractor strength for RLVR. We use the empirical strength $\hat{s}_j$ as the core metric to optimize the distractor set $\mathcal{O}_-(q)$ while preserving the stem and correct answer. The process iterates through two phases: (1) **Filling Phase**: When effective distractors (with $\hat{s}_j > 0$) are below the target count, we generate new candidates and add them if they demonstrate non-zero empirical strength. (2) **Replacement Phase**: Once the pool reaches capacity, we identify the weakest distractor and replace it with a newly generated candidate if and only if the new candidate exhibits higher

Dataset	Qwen2-7B						Llama-3.1-8B					
	Base	Orig.	Direct	Filter	Rewrite	IDC	Base	Orig.	Direct	Filter	Rewrite	IDC
MedQA	47.21	49.25	48.70	48.00	48.39	**49.80**	59.07	62.14	65.20	65.04	64.41	**66.14**
SuperGPQA (Clinical)	25.62	25.53	26.11	26.60	26.27	**27.91**	33.25	33.58	35.39	35.39	35.39	**35.71**
MMLU-Pro (Health)	44.38	46.21	47.07	46.09	45.72	**49.51**	58.56	57.46	57.95	59.17	58.19	**60.76**
MedXpertQA	10.24	10.98	10.29	10.86	10.24	**11.88**	14.29	16.29	15.47	16.45	15.67	**19.31**
PubMedQA	69.50	72.50	72.00	72.90	71.20	**74.00**	78.00	77.80	78.30	78.50	**78.80**	77.60
Average	39.39	40.89	40.83	40.89	40.36	**42.62**	48.63	49.45	50.46	50.91	50.49	**51.90**

Table 3: Downstream performance on medical datasets. **Base**: Base instruct model. **Orig.**: Standard RLVR on original MCQs. **Direct**: Direct conversion to short-answer. **Filter**: Filtering non-convertible questions. **Rewrite**: Model-based conversion. **IDC**: Our Iterative Distractor Curation method. All results reported as accuracy (%).

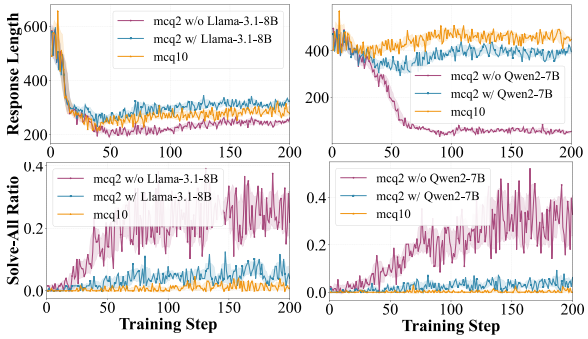


Figure 3: Training dynamics analysis: *response length* (top row) and *solve-all ratio* (bottom row) for **Llama** (left column) and **Qwen** (right column) across distractor settings. Selecting the weakest distractor (w/o) leads to rapid saturation and shorter responses.

empirical strength. After T iterations, we select the option set that maximizes the number of effective distractors while minimizing the model's pass rate. This approach functions as *rejection sampling in the option space*, progressively replacing weak distractors with adversarial ones to provide clearer high-contrast reward signals. The process is summarized in Algorithm 1 (see Appendix Q), while detailed prompts for distractor generation, evaluation, and semantic equivalence checks (labeled "Conflict" in Figure 4) are provided in Appendix H.

5.3 Experimental Setup

In this section, we apply the proposed framework to different models and evaluate its impact on downstream tasks. We use the following experimental settings (see Appendix A for details).

Models. Following the analytical experiments, we use Qwen2-7B-Instruct and Llama-3.1-8B-Instruct for both RLVR training and distractor strength estimation. For distractor generation, we employ Qwen2.5-32B-Instruct as the default generator due to its strong instruction-following ability and moderate cost. We further investigate the impact of generator models and explore the potential for **self-improvement** (using the target model itself as the generator) in the ablation studies.

Datasets. We focus on the medical domain. We use the MedQA (Jin et al., 2020) training split as our training data, and evaluate in-domain performance on the MedQA test set. For out-of-distribution (OOD) evaluation, we use the clinical subset of SuperGPQA (Team et al., 2025b) and the health subset of MMLU-Pro (Wang et al., 2024b). We also evaluate on MedXpertQA (Zuo et al., 2025), a challenging benchmark designed for evaluating expert-level medical knowledge and reasoning capabilities. In addition, we evaluate the trained models on PubMedQA (Jin et al., 2019) to measure transfer to the short-answer (QA) format.

Baselines. We compare our method against the following baselines (see Appendix E for details):

- *Direct conversion:* strip options and convert all questions to a short-answer format.
- *Filtering non-convertible questions (Akter et al., 2025):* first filter out multiple-choice questions that cannot be directly converted, then convert remaining questions to short-answer questions by stripping the options.
- *Model-based conversion:* use a language model to rewrite each question into a fill-in-the-blank or short-answer format.

5.4 Results

Table 3 presents the results on downstream medical tasks. IDC improves performance in the medical domain on both in-domain and OOD evaluations, consistently outperforming baselines based on direct conversion or model-based rewriting. The only exception is PubMedQA with Llama-3.1-8B, which we attribute to format alignment bias: PubMedQA is a short-answer QA task (not

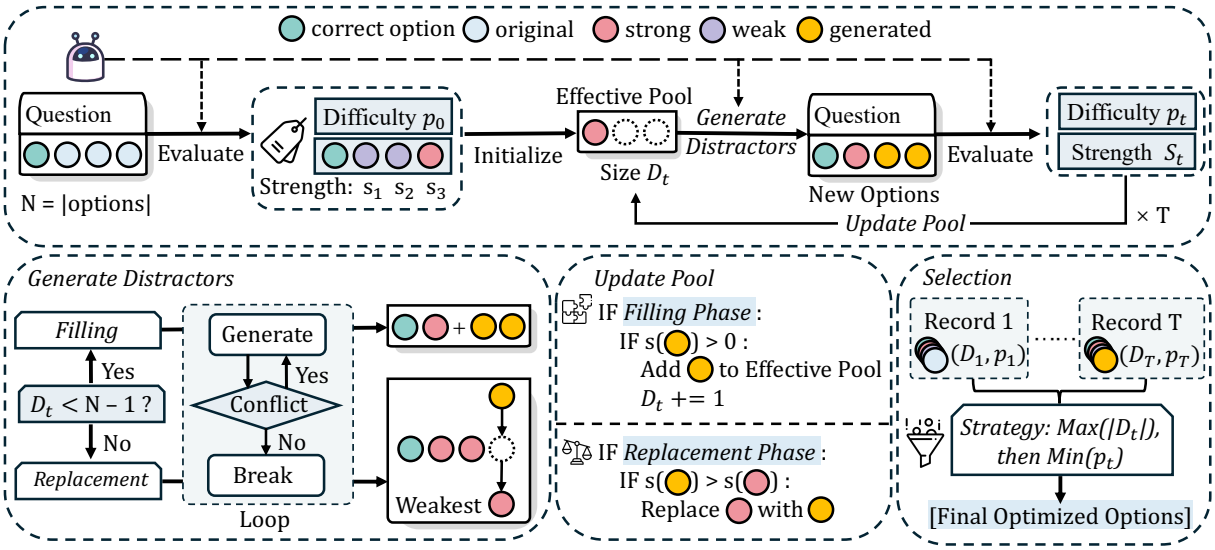


Figure 4: The framework of Iterative Distractor Curation (IDC).

Dataset	Qwen2-7B						Llama-3.1-8B					
	Base	Self	Qwen-32B	OSS	DS-V3.1	Llama-70B	Base	Self	Qwen-32B	OSS	DS-V3.1	Llama-70B
MedQA	47.21	47.29	49.80	48.70	49.41	**49.88**	59.07	66.61	66.14	65.75	**67.87**	67.09
SuperGPQA	25.62	27.75	27.91	**28.49**	27.75	27.34	33.25	36.45	35.71	**38.51**	34.81	36.45
MMLU-Pro	44.38	**49.88**	49.51	46.21	47.07	47.31	58.56	**61.00**	60.76	59.17	60.88	**61.00**
MedXpertQA	10.24	12.45	11.88	**13.14**	11.80	12.49	14.29	**20.29**	19.31	19.27	17.96	18.04
PubMedQA	69.50	**74.60**	74.00	71.60	71.50	72.50	78.00	**78.10**	77.60	76.80	77.90	77.00
Average	39.39	42.39	**42.62**	41.63	41.51	41.90	48.63	**52.49**	51.90	51.90	51.88	51.92

Table 4: Ablation on distractor generators. We compare models trained with distractors generated by the model itself ("Self") versus stronger external models. "Base" denotes the original instruct model without RLVR. Qwen-32B: Qwen2.5-32B-Instruct; OSS: GPT-OSS-120B; DS-V3.1: DeepSeek-V3.1; Llama-70B: Llama-3.1-70B-Instruct.

MCQ), so conversion baselines enjoy a direct format-alignment advantage on this benchmark. This effect is amplified by Llama's stronger sensitivity to format mismatch—as Table 7 shows, Llama exhibits strict diagonal dominance in cross-option generalization. On Qwen2-7B, which is less format-sensitive, IDC outperforms all baselines on PubMedQA, confirming that IDC's gains do transfer beyond format boundaries. Multi-seed experiments (Appendix M) confirm that these gains are stable across independent runs, and an additional law domain experiment (Appendix O) validates that IDC generalizes beyond the medical domain.

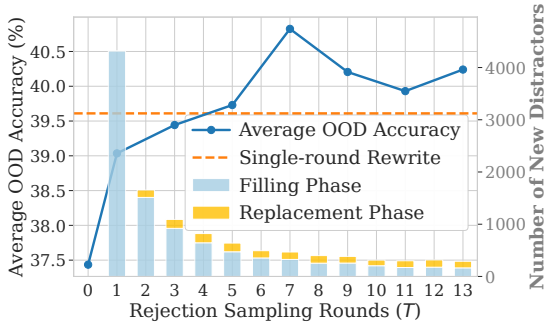


Figure 5: Performance gains and distractor dynamics across rejection sampling rounds.

5.5 Ablation Studies

Impact of rejection sampling rounds. We study the influence of rejection sampling iterations T. Figure 5 compares our iterative approach with a single-round baseline that rewrites all distractors at once without strength estimation. Note that the training data (MedQA) uses 4-choice questions (1 correct + 3 distractors), so the single-round baseline replaces all 3 distractors simultaneously, while IDC replaces one at a time. The single-round strategy (dashed line) is outperformed by our iterative method, demonstrating the necessity of iterative refinement. We observe that newly effective distractors decrease rapidly after the initial rounds. Based on this saturation, we fix $T = 7$ for all experiments to balance performance and efficiency. Experimental details are in Appendix G.

Computational cost analysis. Table 5 summarizes

Stage	Wall-Clock Time
Distractor Generation (per round)	∼15–20 min
Strength Evaluation (per round)	<5 min
Full IDC Pipeline (7 rounds)	∼1.5 h
RLVR Training	18–20 h
IDC Overhead / Total	**<10%**

Table 5: Computational cost breakdown on 8×H200 GPUs (MedQA, ∼10k samples).

Factor	Setting	n	Spurious Rate	
			Before	After
Quantity	`mcq2` → `mcq10`	400	0.241	**0.095**
Quality	w/o → w/ IDC	200	0.260	**0.110**

Table 6: Spurious reward rates before and after varying option design on MMLU-Pro (row 1) and MedQA (row 2). n: annotated response pairs.

the wall-clock time for each stage. The overhead of IDC is modest: strength evaluation only requires the model to output a single answer label without chain-of-thought, and the target model itself serves as both generator and evaluator (Table 4), eliminating the need for any external model. The full IDC pipeline accounts for less than 10% of the total pipeline time.

Spurious reward rate analysis. To empirically validate our theoretical framework (Section 3), we measure how distractor design affects spurious reward rates (Table 6). For each sampled response, DeepSeek-V3.1 provides structured judgments on answer correctness, reasoning correctness, and whether a spurious reward occurred (correct answer despite flawed reasoning), followed by human verification. Results are pooled over Qwen2-7B and Llama-3.1-8B. Both increasing option count and applying IDC substantially reduce spurious rewards, confirming that option design directly impacts reward signal accuracy.

Ablation Study on Distractor Generators We investigate whether a stronger generator necessarily leads to better performance. Table 4 compares using the target model itself against stronger external models as generators. Results show that stronger generators do not guarantee better training outcomes. Self-generated distractors yield competitive or superior results on MMLU-Pro and PubMedQA. This suggests that distractor effectiveness depends on the current policy: self-generated options likely lie closer to the model's decision boundary, providing more pertinent supervision than "harder" but distributionally distinct options from stronger models. Additionally, the experiment in Appendix J demonstrates the scalability of our approach to larger model architectures. We also verify in Appendix K that self-verification (estimating strength with the model itself) is sufficient.

Semantic collapse analysis. A potential failure mode of self-generated distractors is semantic collapse, where a generated distractor becomes semantically equivalent to the correct answer. We analyze this on 5,000 MedQA training questions (35,000 IDC generation rounds). Among all candidates, 652 were identified as equivalent to the gold answer (648 semantically, 4 exact matches), spanning 222 out of 5,000 questions (4.44%). Representative cases include medical aliases: *Vitamin B1* vs. *Thiamine*, *Giant cell arteritis* vs. *Temporal arteritis*, and *Anti-Ro antibodies* vs. *Anti-SSA antibodies*. Such cases are caught by the semantic equivalence check (Section 5.2) and replaced with new candidates, ensuring reward signal integrity.

6 Conclusions

In this work, we systematically study factors influencing the suitability of multiple-choice questions for RLVR. Through controlled experiments on MMLU-Pro, we find that train-test consistency in option count and distractor strength are key factors. Based on these insights, we propose model-based option expansion and strength-guided distractor synthesis. This approach addresses the identified issues and enhances utilization of existing multiple-choice data. Crucially, models can self-generate these challenging distractors, enabling self-improvement without external supervision.

Limitations

Our method effectively mitigates challenges in applying RLVR to multiple-choice questions. However, predicting a model's capability to synthesize effective distractors remains difficult *a priori*. The relationship between general model capability and synthetic distractor quality is not yet clear. We believe that training specialized distractor generators aligned with RLVR objectives is an important direction for future work.

Use of AI Assistants

We primarily use AI assistants to improve and enrich our writing, especially by leveraging LLMs to help us write taxonomy in LaTeX.

Acknowledgments

The authors wish to thank the AC and anonymous reviewers for their constructive comments. This work was supported by Shanghai AI Laboratory and the National Natural Science Foundation of China (72595845).

References

Syeda Nahida Akter, Shrimai Prabhumoye, Matvei Novikov, Seungju Han, Ying Lin, Evelina Bakhturina, Eric Nyberg, Yejin Choi, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2025. Nemotron-crossthink: Scaling self-learning beyond math reasoning.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in ai safety.

Bernd Bohnet, Kevin Swersky, Rosanne Liu, Pranjal Awasthi, Azade Nova, Javier Snaider, Hanie Sedghi, Aaron T Parisi, Michael Collins, Angeliki Lazaridou, Orhan Firat, and Noah Fiedel. 2024. Long-span question-answering: Automatic question generation and qa-system ranking via side-by-side evaluation.

Kedi Chen, Dezhao Ruan, Yuhao Dan, Yaoting Wang, Siyu Yan, Xuecheng Wu, Yinqi Zhang, Qin Chen, Jie Zhou, Liang He, Biqing Qi, Linyang Li, Qipeng Guo, Xiaoming Shi, and Wei Zhang. 2025. A survey of inductive reasoning for large language models.

Shang-Hsuan Chiang, Ssu-Cheng Wang, and Yao-Chung Fan. 2022. CDGP: Automatic cloze distractor generation based on pre-trained language model. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 5835–5840, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, and 6 others. 2025. Process reinforcement through implicit rewards.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, and 181 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. 2024. Data augmentation using LLMs: Data perspectives, learning paradigms and challenges. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 1679–1705, Bangkok, Thailand. Association for Computational Linguistics.

Wanyong Feng, Jaewook Lee, Hunter McNichols, Alexander Scarlatos, Digory Smith, Simon Woodhead, Nancy Otero Ornelas, and Andrew Lan. 2024. Exploring automated distractor generation for math multiple-choice questions via large language models.

Nigel Fernandez, Alexander Scarlatos, Wanyong Feng, Simon Woodhead, and Andrew Lan. 2024. Divert: Distractor generation with variational errors represented as text for math multiple-choice questions.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2022. Simcse: Simple contrastive learning of sentence embeddings.

Bilal Ghanem and Alona Fyshe. 2024. Disto: Textual distractors for multiple choice reading comprehension questions using negative sampling. In *Proceedings of the 17th International Conference on Educational Data Mining*, pages 6–17, Atlanta, Georgia, USA. International Educational Data Mining Society.

Dong Guo, Faming Wu, Feida Zhu, Fuxing Leng, Guang Shi, Haobin Chen, and 1 others. 2025. Seed1.5-vl technical report.

Jingcheng Hu, Yinmin Zhang, Qi Han, Daxin Jiang, Xiangyu Zhang, and Heung-Yeung Shum. 2025. Open-reasoner-zero: An open source approach to scaling up reinforcement learning on the base model.

Yuzhen Huang, Weihao Zeng, Xingshan Zeng, Qi Zhu, and Junxian He. 2025. From accuracy to robustness: A study of rule- and model-based verifiers in mathematical reasoning.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2020. What disease does this patient have? a large-scale open domain question answering dataset from medical exams.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577, Hong Kong, China. Association for Computational Linguistics.

Wouter Kool, Herke van Hoof, and Max Welling. 2019. Buy 4 reinforce samples, get a baseline for free! In *DeepRLStructPred@ICLR*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2025. Tulu 3: Pushing frontiers in open language model post-training.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Che Liu, Haozhe Wang, Jiazhen Pan, Zhongwei Wan, Yong Dai, Fangzhen Lin, Wenjia Bai, Daniel Rueckert, and Rossella Arcucci. 2025a. Beyond distillation: Pushing the limits of medical llm reasoning with minimalist rule-based rl.

Naiming Liu, Shashank Sonkar, and Richard G. Baraniuk. 2025b. Do llms make mistakes like students? exploring natural alignment between language models and human error patterns.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025c. Understanding r1-zero-like training: A critical perspective.

Michael Luo, Naman Jain, Jaskirat Singh, Sijun Tan, Ameen Patel, Qingyang Wu, Alpay Ariyak, Colin Cai, Tarun Venkat, Shang Zhu, Ben Athiwaratkun, Manan Roongta, Ce Zhang, Li Erran Li, Raluca Ada Popa, Koushik Sen, and Ion Stoica. 2025a. Deepswe: Training a state-of-the-art coding agent from scratch by scaling rl. https://pretty-radio-b75.notion.site/DeepSWE-Training-a-Fully-Open-sourced-State-of-the-Art-Coding-Agent-by-Scaling-RL-22281902c1468193aabbe9a8c59bbe33. Notion Blog.

Michael Luo, Sijun Tan, Roy Huang, Ameen Patel, Alpay Ariyak, Qingyang Wu, Xiaoxiang Shi, Rachel Xin, Colin Cai, Maurice Weber, Ce Zhang, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025b. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51. Notion Blog.

Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. 2025c. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2. Notion Blog.

Ziyang Luo, Kaixin Li, Hongzhan Lin, Yuchen Tian, Mohan Kankanhalli, and Jing Ma. 2025d. Tree-of-evolution: Tree-structured instruction evolution for code generation in large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 297–316, Vienna, Austria. Association for Computational Linguistics.

Jeroen Offerijns, Suzan Verberne, and Tessa Verhoef. 2020. Better distractions: Transformer-based distractor generation and multiple choice question filtering.

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. 2022. Training language models to follow instructions with human feedback.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Fanyi Qu, Hao Sun, and Yunfang Wu. 2024. Unsupervised distractor generation via large language model distilling and counterfactual contrastive decoding. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 827–838, Bangkok, Thailand. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model.

Alexander Scarlatos, Wanyong Feng, Digory Smith, Simon Woodhead, and Andrew Lan. 2024. Improving automated distractor generation for math multiple-choice questions with overgenerate-and-rank.

Rulin Shao, Shuyue Stella Li, Rui Xin, Scott Geng, Yiping Wang, Sewoong Oh, Simon Shaolei Du, Nathan Lambert, Sewon Min, Ranjay Krishna, Yulia Tsvetkov, Hannaneh Hajishirzi, Pang Wei Koh, and Luke Zettlemoyer. 2025. Spurious rewards: Rethinking training signals in rlvr.

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.

Shashank Sonkar, Naiming Liu, Xinghe Chen, and Richard G. Baraniuk. 2025. The imitation game for educational ai.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, and 1 others. 2025a. Kimi k1.5: Scaling reinforcement learning with llms.

P Team, Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shawn Gavin, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, and 78 others. 2025b. Supergpqa: Scaling llm evaluation across 285 graduate disciplines.

Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, and Yunhong Wang. 2024a. A survey on data synthesis and augmentation for large language models.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. 2024b. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark.

Zaitian Wang, Jinghan Zhang, Xinhao Zhang, Kunpeng Liu, Pengfei Wang, and Yuanchun Zhou. 2025. Diversity-oriented data augmentation with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22265–22283, Vienna, Austria. Association for Computational Linguistics.

Mingqi Wu, Zhihao Zhang, Qiaole Dong, Zhiheng Xi, Jun Zhao, Senjie Jin, Xiaoran Fan, Yuhao Zhou, Huijie Lv, Ming Zhang, Yanwei Fu, Qin Liu, Songyang Zhang, and Qi Zhang. 2025. Reasoning or memorization? unreliable results of reinforcement learning due to data contamination.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. 2025. Logic-rl: Unleashing llm reasoning with rule-based reinforcement learning.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2025. Wizardlm: Empowering large pre-trained language models to follow complex instructions.

Han Cheng Yu, Yu An Shih, Kin Man Law, KaiYu Hsieh, Yu Chen Cheng, Hsin Chih Ho, Zih An Lin, Wen-Chuan Hsu, and Yao-Chung Fan. 2024. Enhancing distractor generation for multiple-choice questions with retrieval augmented pretraining and knowledge graph integration. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 11019–11029, Bangkok, Thailand. Association for Computational Linguistics.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, and 16 others. 2025. Dapo: An open-source llm reinforcement learning system at scale.

Weihao Zeng, Yuzhen Huang, Qian Liu, Wei Liu, Keqing He, Zejun Ma, and Junxian He. 2025. Simplerl-zoo: Investigating and taming zero reinforcement learning for open base models in the wild.

Sheng Zhang, Qianchu Liu, Guanghui Qin, Tristan Naumann, and Hoifung Poon. 2025a. Med-rlvr: Emerging medical reasoning from a 3b base model via reinforcement learning.

Xinghua Zhang, Haiyang Yu, Cheng Fu, Fei Huang, and Yongbin Li. 2025b. Iopo: Empowering llms with complex instruction following via input-output preference optimization.

Yuhui Zhang, Yuchang Su, Yiming Liu, Xiaohan Wang, James Burgess, Elaine Sui, Chenyu Wang, Josiah Aklilu, Alejandro Lozano, Anjiang Wei, Ludwig Schmidt, and Serena Yeung-Levy. 2025c. Automated generation of challenging multiple-choice questions for vision language model evaluation.

Chujie Zheng, Shixuan Liu, Mingze Li, Xiong-Hui Chen, Bowen Yu, Chang Gao, Kai Dang, Yuqiong Liu, Rui Men, An Yang, Jingren Zhou, and Junyang Lin. 2025. Group sequence policy optimization.

Yuxin Zuo, Shang Qu, Yifei Li, Zhangren Chen, Xuekai Zhu, Ermo Hua, Kaiyan Zhang, Ning Ding, and Bowen Zhou. 2025. Medxpertqa: Benchmarking expert-level medical reasoning and understanding.

A Experimental Hyperparameter Settings

All experiments use the same hyperparameter configuration to ensure comparability. The base models are Llama-3.1-8B-Instruct and Qwen2-7B-Instruct, trained using the GRPO algorithm, with RLOO (Kool et al., 2019) employed as the advantage estimator. All experiments are conducted on a single node equipped with 8 H200 GPUs.

Batch Size and Sampling. The training batch size is 128, and the mini-batch size is also 128 (i.e., fully on-policy). We sample 8 responses per question for group-wise advantage estimation.

Optimizer Configuration. The learning rate is set to 1e-6 using a cosine warmup strategy. The warmup duration is 10 steps, and the minimum learning rate is 0.1 times the initial learning rate. The weight decay coefficient is 0.1, and the gradient clipping threshold is 1.0.

Reinforcement Learning Settings. We use KL divergence loss to constrain policy updates, with a coefficient of 0.001. The entropy regularization coefficient is 0.001.

Sequence Length. Both the maximum prompt length and maximum response length are set to 2048 tokens.

Generation Configuration. For training, sampling uses temperature=1.0, top-p=1.0, and top-k=-1; for testing, we use greedy decoding.

B MMLU-Pro Data Preprocessing Details

This section details the preprocessing procedure for the MMLU-Pro dataset used in Section 4.

The original MMLU-Pro (Wang et al., 2024b) test set comprises 12,032 questions, approximately 83% of which are 10-choice questions, with the remainder being 3-9 choice questions. We retain only the 10-choice questions to ensure consistency in subsequent processing. We remove a small number of duplicate samples (based on exact matches of question stems and options), resulting in 9,417 10-choice samples for experiments. These are randomly split into a training set (8,004 samples) and a test set (1,413 samples), maintaining an approximate ratio of 85:15.

C Detailed Cross-Option Generalization Results

This section presents the full experimental results of cross-option generalization, corresponding to the analysis in Section 4.

Table 7 reveals a clear diagonal pattern: the highest accuracies and darkest shading concentrate along the diagonal where training and test option counts match. For Llama-3.1, the best performance consistently appears on the diagonal. Qwen2 shows minor deviations (e.g., 4-train peaks on 6-test, 10-train on 8-test), but generally confirms that proximity to the test option-count matters more than the absolute count. This suggests that the number of training options has a limited impact on performance compared to the alignment between training and test option counts.

D Experimental Setup for Option Expansion

D.1 Detailed Experimental Configuration

This section provides the detailed experimental configuration for the option expansion experiments reported in Table 1.

Models. We use three types of option generator models for distractor synthesis, all with temperature set to 0.7:

- **GPT-OSS-120B**: The GPT-OSS-120B model with high reasoning effort enabled.
- **Qwen2.5-32B-Instruct**: The Qwen2.5-32B-Instruct model.
- **Self**: The target model itself (Qwen2-7B or Llama-3.1-8B) is used to generate distractors.

Output Postprocessing. Model outputs are post-processed using a regex-based cleaning script to remove common output prefixes. Specifically, we strip patterns such as `"Option A: xxxxx"`, `"I. xxxxx"`, or similar formatting, retaining only the core option content itself. This ensures that the generated distractors are clean and consistent with the original option format.

D.2 Option Expansion Prompt

This section provides the full prompt used for option expansion experiments in Section 4.

Prompt: Option Expansion

```
I have a multiple-choice question with
{current_num} options, one of which is
correct, and I need to expand it to a
{target_num}-option multiple-choice question.
Please generate {new_options_num} additional
plausible but incorrect options to accompany
the original options.
```

Train	Test					Avg
	2	4	6	8	10	
2	**80.27**	67.82	60.50	54.28	52.44	63.06
4	79.95	**69.96**	63.79	57.40	56.19	65.46
6	79.57	69.58	**64.07**	59.21	57.87	**66.06**
8	76.76	69.65	63.93	**60.92**	58.17	65.89
10	76.98	69.79	63.32	60.19	**58.61**	65.78
Avg	78.71	69.36	63.12	58.40	56.66	65.25

(a) Llama-3.1-8B

Train	Test					Avg
	2	4	6	8	10	
2	**77.61**	64.19	56.34	49.50	46.66	58.86
4	72.15	**65.15**	**58.22**	51.92	48.57	**59.20**
6	71.17	64.36	57.49	52.71	49.68	59.08
8	67.68	61.59	56.34	52.13	48.37	57.22
10	68.68	63.67	57.82	**53.50**	**50.07**	58.75
Avg	71.46	63.79	57.24	51.95	48.67	58.62

(b) Qwen2-7B

Table 7: Cross-option generalization results (Accuracy %). **Bold** and Underlined denote the best and second-best results in each column (excluding the mix strategy). Background color intensity represents normalized score z, with darker colors indicating better performance.

```
Requirements:
1. The new options should be plausible and
   similar in style to the existing options
2. The new options should be clearly incorrect
   (not ambiguous)
3. The new options should test different
   aspects of the question topic
4. Please think step by step about how to
   create good distractors

IMPORTANT - Output Format:
You MUST output ONLY a valid JSON object
wrapped in ```json and ``` markers.
Do NOT include any other text before or after
the JSON code block.
The JSON must have exactly these two fields:
- "thinking": a string containing your
  reasoning process
- "new_options": an array of exactly
  {new_options_num} strings

Example format:
```json
{
  "thinking": "I will create options that are
     similar to the existing ones but test
     different concepts...",
  "new_options": ["Option 1", "Option 2",
     "Option 3"]
}
```

Here is the original question:
<original question>
{raw_question}
</original question>

Now generate exactly {new_options_num} new
incorrect options in the JSON format specified.
```

E Baseline Prompts

This section provides the detailed prompts used in the baseline methods described in Section 5.3.

E.1 Filtering Prompt for Evaluating Question Convertibility

The following prompt is used to evaluate whether a multiple-choice question can be transformed into a standalone QA pair:

Prompt: Question Convertibility Filter

```
You are an expert in educational assessment.
Evaluate whether the following Multiple-Choice
Question can be transformed into a standalone
QA pair.

Original Question: {question}

Correct Answer: {correct_answer}

Task:
Determine if the stem is understandable
and answerable without seeing the options.

Criteria for "NOT_CONVERTIBLE":
- Contains phrases like "Which of the
  following", "EXCEPT", "NOT true",
  "All of the above".
- Requires comparing options to find the "best"
  answer (e.g., "best describes", "best
  explains").
- The answer depends entirely on the specific
  choices given.
- Refers to figures, graphs, or tables that are
  part of the options.

Criteria for "CONVERTIBLE":
- The question has a clear, specific answer
  that doesn't depend on seeing options.
- The question is complete and self-contained.

CRITICAL: You MUST end your response with one
of these EXACT lines:
FINAL_LABEL: CONVERTIBLE
OR
FINAL_LABEL: NOT_CONVERTIBLE

Example Output 1:
Analysis: The question asks "Which of the
following is a fruit?" which implies a
selection from a specific list. Without
```

```
options, the answer is too open-ended.
FINAL_LABEL: NOT_CONVERTIBLE

Example Output 2:
Analysis: The question asks "What is the most
likely diagnosis?" which is clear and specific.
It can be answered without seeing the options.
FINAL_LABEL: CONVERTIBLE

Your Output:
```

E.2 Model-Based Conversion Prompt

The following prompt is used to convert multiple-choice questions to fill-in-the-blank or short-answer format:

Prompt: Model-Based Conversion

```
I will give you a multiple-choice question with
answer options and its correct answer. Please
convert it into a fill-in-the-blank or
short-answer question.

IMPORTANT INSTRUCTIONS:
1. PRESERVE ALL CONTEXT: Keep the entire
   clinical scenario, patient information,
   symptoms, lab values, physical exam
   findings, and background information
   EXACTLY as given. Do NOT summarize,
   shorten, or remove any details.

2. ONLY modify the question format: Remove the
  answer options (A, B, C, D) and convert the
  final question into a fill-in-the-blank or
   short-answer format.

3. The modified question should ask for the
  same information, but without providing the
   multiple-choice options.

4. Provide the correct answer as a short,
   direct response (no explanation needed).

Original question with options:
{question}

Correct answer:
{answer}

Return the output using this format:
<Question>{Modified Question - with ALL
original context preserved}</Question>
<Answer>{Short Answer}</Answer>

Example:
If given: "A 50-year-old man presents with
chest pain... BP 120/80... What is the
diagnosis? A. MI B. Angina C. PE D. GERD"

You should return: "A 50-year-old man presents
with chest pain... BP 120/80... What is the
diagnosis?" (keeping all context, just
removing A/B/C/D options)
```

F Experimental Setup for Distractor Strength Analysis

This section provides the detailed experimental configuration for the distractor strength analysis experiments reported in Section 5.1 and Table 2. To estimate the empirical strength $\hat{s}_j$ of each distractor o_j, we adopt the following procedure:

Sampling Configuration. For each question, we sample $K = 8$ trajectories from the evaluation model with temperature set to 0.7. The model generates complete outputs $\{y_k = (c_k, a_k)\}_{k=1}^{K}$, where c_k is the reasoning chain and a_k is the selected option.

Distractor Selection. After computing empirical strengths for all distractors in each question, we rank them by $\hat{s}_j$ in descending order. For the experiments in Table 2:

- **mcq2-random**: Randomly select one distractor from the original 9 distractors.
- **w/o [Model]**: Retain the distractor with the lowest empirical strength estimated by the specified model.
- **w/ [Model]**: Retain the distractor with the highest empirical strength estimated by the specified model.

G Experimental Setup for Rejection Sampling Analysis

This section details the experimental setup for the rejection sampling analysis presented in Figure 5. The y-axis in Figure 5 represents the mean accuracy of Qwen2-7B-Instruct across four out-of-distribution (OOD) datasets: SuperGPQA (Clinical), MMLU-Pro (Health), MedXpertQA, and PubMedQA. For the single-round baseline in Figure 5, we use Qwen2.5-32B-Instruct as the generator model and do not use a strength estimator; the model directly rewrites all distractors into a new option set, as shown in the prompt below.

Prompt: Direct Distractor Rewriting

```
You are a professional expert in generating
distractors for multiple-choice questions.
Your task is to evaluate the existing
distractors and improve them if necessary.

Question Information:
Question Text: {question_text}
Correct Option: {correct_option}
Correct Answer: {correct_answer}
Existing Options (with all options
```

```
including correct answer): {all_options}

Your Task:
Review the existing distractors (incorrect
options). You have TWO choices:
1. If you think the existing distractors are
   already reasonable and effective, you
   can KEEP them as-is
2. If you think some or all distractors need
   improvement, REPLACE only those that need
   improvement

Requirements for generating/keeping
distractors:
1. Distractors must be plausible and able to
   attract students who are not careful enough
   or have insufficient knowledge
2. Distractors should be consistent with the
   correct answer in form and style
3. **CRITICAL**: Distractors MUST NOT be the
   same as or similar to the correct answer
   "{correct_answer}"
4. **CRITICAL**: Do NOT use the correct option
   identifier "{correct_option}" as a key in
   your output
5. **CRITICAL**: ONLY use the SAME option
   identifiers as the existing distractors. For
   example, if existing distractors use keys A,
   C, D, your output must ONLY use keys A, C, D
6. Distractors should have discrimination,
   avoiding duplication or being too similar to
   each other
7. If you decide to keep existing distractors,
   output them exactly as they are

Decision:
First, decide whether you want to KEEP all
existing distractors or IMPROVE some/all of
them.

Output Format:
Please return the result in JSON format with
the following fields:
- decision: string type, either "KEEP_ALL" or
  "IMPROVE"
- distractors: dictionary type, with keys as
  option identifiers (MUST use the same keys as
  existing distractors)
  - If decision is "KEEP_ALL", output the
    original distractors unchanged
  - If decision is "IMPROVE", output the
    improved distractors (you can keep some
    and improve others)
- reasoning: string type, explaining your
  decision and rationale

Example 1 (Keep all):
If existing options are: {"A": "distractor 1",
"B": "correct answer", "C": "distractor 2",
"D": "distractor 3"}
And correct option is B, you might output:
{"decision": "KEEP_ALL", "distractors":
{"A": "distractor 1", "C": "distractor 2",
"D": "distractor 3"}, "reasoning": "The
existing distractors are already well-designed
and plausible."}

Example 2 (Improve some):
If existing options are: {"A": "obviously
wrong", "B": "correct answer", "C":
"reasonable distractor", "D": "too similar
to B"}
And correct option is B, you might output:
{"decision": "IMPROVE", "distractors":
{"A": "improved distractor 1", "C":
"reasonable distractor", "D": "improved
distractor 2"},
"reasoning": "Options A and D need
improvement, but C is already good."}
```

H Prompts for Iterative Distractor Curation

This section provides the detailed prompts used in the Iterative Distractor Curation framework described in Section 5.2. The framework employs three key prompts to ensure the quality and effectiveness of generated distractors: (1) a prompt for generating new distractor candidates, (2) a prompt for evaluating model performance on the multiple-choice question, and (3) a prompt for semantic equivalence checking to ensure generated distractors do not become alternative correct answers.

H.1 Distractor Generation Prompt

The following prompt is used to generate new distractor candidates during both the filling and replacement phases:

Prompt: Distractor Generation

```
You are a professional expert in generating
distractors for multiple-choice questions.
Your task is to generate high-quality
REPLACEMENT distractors for the given
question.

Question Information:
Question Text: {question_text}
Correct Option: {correct_option}
Correct Answer: {correct_answer}
Number of Distractors to Generate:
{num_distractors}
Existing Distractors (to be replaced):
{existing_distractors}

Your Task:
You need to REPLACE the existing distractors
with better, more effective ones. The new
distractors should be more challenging and
attractive to students who lack careful
thinking or sufficient knowledge.

Requirements:
1. The generated distractors must be plausible
   and able to attract students who are not
   careful enough or have insufficient
   knowledge
2. Distractors should be consistent with the
   correct answer in form and style
```

```
3. **CRITICAL**: Distractors MUST NOT be the
   same as or similar to the correct answer
   "{correct_answer}". They should be different
   but plausible alternatives.
4. **CRITICAL**: Do NOT use the correct option
   identifier "{correct_option}" as a key
   in your output.
5. **CRITICAL**: ONLY use the SAME option
   identifiers as the existing distractors. For
   example, if existing distractors use keys
   A, C, D, your output must ONLY use keys A,
   C, D. Do NOT create new keys like E, F, G, etc.
6. Distractors should have discrimination,
   avoiding duplication or being too similar
   to each other
7. The new distractors should be more effective
   than the existing ones - they should be
   harder to distinguish from the correct
   answer or more tempting to select

{failed_feedback}

Output Format:
Please return the result in JSON format with
the following fields:
- distractors: dictionary type, with keys as
  option identifiers (MUST use the same keys
  as existing distractors, e.g., if existing
  has A, C, D, output must have A, C, D)
- reasoning: string type, explaining your
  rationale for generating these distractors

Example:
If existing distractors are: {"A": "old
distractor 1", "C": "old distractor 2",
"D": "old distractor 3"}
Your output should be: {"A": "new
distractor 1", "C": "new distractor 2",
"D": "new distractor 3"}
Do NOT output: {"E": "...", "F": "...",
"G": "..."}
```

H.2 Model Evaluation Prompt

The following prompt is used to evaluate whether the base model can correctly answer the multiple-choice question with the generated distractors:

Prompt: Model Evaluation

```
You are a student taking an exam. Please read
the following question carefully and choose
the answer you believe is correct.

Question:
{question_text}

Options:
{options}
```

H.3 Semantic Equivalence Check Prompt

The following prompt is used to ensure that generated distractors are not semantically equivalent to the correct answer, preventing the creation of questions with multiple valid answers:

Prompt: Semantic Equivalence Check

```
You are an expert in evaluating whether two
answer options in a multiple-choice question
are semantically equivalent. Your task is to
determine if Text 2 conveys EXACTLY the same
meaning as Text 1 (the correct answer), such
that both would be equally correct.

Question Context:
{question_text}

Text 1 (Correct Answer): {text1}
Text 2 (Generated Distractor - to be checked):
{text2}

CRITICAL: Are these two texts IDENTICAL in
meaning? Would both be equally correct as
the answer to the question above?

Evaluation Guidelines:
1. **Be VERY STRICT**: Only mark as EQUIVALENT
   if the texts are COMPLETELY identical in
   meaning
2. **Context matters**: Evaluate the texts
   specifically in the context of the question
   above

Please respond with ONLY ONE of the following:
- "EQUIVALENT" - if and only if the texts are
  completely identical in meaning and would
  both be equally correct
- "NOT_EQUIVALENT" - if there is ANY difference
  in meaning, specificity, certainty, or scope

Your response:
```

I Case Study: Iterative Distractor Evolution

This section provides a detailed case study illustrating how the Iterative Distractor Curation framework progressively strengthens distractors through the filling and replacement phases. We trace the evolution of a medical question from MedQA that initially had a passrate of 1.0 (the model was not distracted by any incorrect options).

I.1 Original Question

The original question contains weak distractors that fail to challenge the model:

Initial State: Passrate = 1.0

```
Question: A 58-year-old man comes to the
emergency department for complaints of
crushing chest pain for 4 hours. He was
shoveling snow outside when the pain started.
It is rated 7/10 and radiates to his left arm.
An electrocardiogram (ECG) demonstrates
ST-segment elevation in leads V2-4. He
subsequently undergoes percutaneous coronary
intervention and is discharged with aspirin,
clopidogrel, carvedilol, atorvastatin, and
```

```
lisinopril. Five days later, the patient is
brought to the emergency department by his wife
with complaints of dizziness. He reports
lightheadedness and palpitations for the past
2 hours but otherwise feels fine. His
temperature is 99.7°F (37.6°C), blood pressure
is 95/55 mmHg, pulse is 105/min, and
respirations are 17/min. A pulmonary artery
catheter is performed and demonstrates an
increase in oxygen concentration at the
pulmonary artery. What finding would you
expect in this patient?

A. Midsystolic click followed by a late
   systolic murmur at the apex
B. Low-pitched, rumbling diastolic murmur at
   the mid-left sternal border
C. Soft, blowing systolic murmur heard best at
   the apex
D. Harsh, loud, holosystolic murmur at the
   lower left sternal border [CORRECT]
```

I.2 Filling Phase

The algorithm begins by generating new distractors to replace the ineffective ones.

First Successful Round: The generator produces two effective distractors.

Filling Round 1

```
Generated Distractors:
- Soft, blowing diastolic murmur at the apex
- High-pitched, crescendo-decrescendo sys-
  tolic murmur at the right upper sternal border
- Continuous machinery-like murmur at
  the left upper sternal border

Reasoning: The distractors have been chosen to
be plausible findings that could be observed in
a patient with similar symptoms, but each
represents a different type of murmur or heart
sound. The distractors are consistent in form
and style with the correct answer, presenting
descriptions of heart murmurs with their
location and characteristics. Each distractor
is chosen to represent a different cardiac
condition, such as aortic stenosis, aortic
regurgitation, or patent ductus arteriosus
, making them plausible but distinct
alternatives to the correct answer.

Validation Result: The first two distractors
successfully confuse the model.
Status: 2 out of 3 distractor slots filled.
```

Second Successful Round: The generator produces one more distractor to complete the filling phase.

Filling Round 2

```
Generated Distractor:
- Low-pitched, rumbling diastolic murmur at
   the mid-left sternal border

Reasoning: The generated distractor 'Low-
pitched, rumbling diastolic murmur at the
mid-left sternal border' is plausible and
consistent with the form and style of the
correct answer. It offers a different type
of murmur (diastolic vs. systolic) and
origin, which could distract students who
may not fully understand the clinical context.
This distractor is designed to challenge
students' knowledge of cardiac murmurs and
their ability to correlate symptoms with
specific cardiac conditions, making it more
effective than the existing distractors.

Validation Result: The new distractor
successfully confuses the model.
Status: Filling phase complete
(3/3 slots filled).
```

I.3 Replacement Phase

After the filling phase, the algorithm enters the replacement phase, where it attempts to replace weaker distractors with stronger ones.

First Successful Round: The generator attempts to replace the weakest distractor.

Replacement Round 1

```
Generated Distractor:
B. Midsystolic click followed by a late
   systolic murmur at the apex

Reasoning: The generated distractor,
'Midsystolic click followed by a late
systolic murmur at the apex,' is plausible
because it represents a common finding in
patients with mitral valve prolapse, which
is a condition that could present with
symptoms of dizziness and palpitations.
However, it is distinctly different from
the correct answer, a harsh, loud, holosys-
tolic murmur at the lower left sternal border,
which is indicative of acute mitral regurgi-
tation. This distractor maintains a consistent
style with the correct answer and existing
distractors, making it plausible yet incorrect,
thus serving as an effective distractor.

Replacement Details:
- Replaced: "Soft, blowing diastolic murmur at
  the apex"
- New: "Midsystolic click followed by a late
  systolic murmur at the apex"
```

Second Successful Round: The generator attempts to replace another weak distractor.

Replacement Round 2

```
Generated Distractor:
- Soft, blowing systolic murmur heard best at
   the apex
```

Dataset	Base	w/ Origin	Self	DS-V3.1	OSS	Qwen2.5-32B	Llama-70B
MedQA	69.76	72.74	73.92	74.71	**74.86**	73.13	74.23
SuperGPQA	37.19	38.83	40.07	39.16	39.66	38.92	**40.48**
MMLU-Pro	63.94	64.79	65.53	65.16	**65.77**	64.55	64.30
MedXpertQA	14.41	16.41	18.20	18.04	**18.69**	18.65	17.96
PubMedQA	77.90	76.80	78.00	78.20	77.30	78.50	**78.60**
Average	52.64	53.91	55.14	55.05	**55.26**	54.75	55.11

Table 8: Ablation study on distractor generators for Qwen2.5-14B-Instruct. The **Base** column shows the performance of the original instruct model without RLVR training. The **w/ Origin** column shows training with original distractors from the dataset. OSS: GPT-OSS-120B; DS-V3.1: DeepSeek-V3.1.

```
Reasoning: The generated distractor 'Soft,
blowing systolic murmur heard best at the apex'
is plausible and consistent with the correct
answer in form and style. It describes a
different type of murmur with a different
location, which makes it a plausible but
incorrect option for students who might not
fully understand the specific characteristics
of the murmur associated with the patient's
condition. This distractor avoids duplicating
the style or specifics of the correct answer,
such as the harshness, loudness, and
holosystolic nature of the murmur at the lower
left sternal border. The distractor is
designed to be sufficiently different from
the correct answer and other existing
distractors to challenge the students'
understanding without being too obvious.

Replacement Details:
- Replaced: "High-pitched, crescendo-
  decrescendo systolic murmur at the right
  upper sternal border"
- New: "Soft, blowing systolic murmur heard
  best at the apex"
```

I.4 Final Question State

After completing the iterative curation process, the question now contains significantly stronger distractors:

Final State: Enhanced Distractors

```
Final Options:
A. Midsystolic click followed by a late
   systolic murmur at the apex [NEW - Effective]
B. Low-pitched, rumbling diastolic murmur at
   the mid-left sternal border [NEW - Effective]
C. Soft, blowing systolic murmur heard best at
   the apex [NEW - Effective]
D. Harsh, loud, holosystolic murmur at the
   lower left sternal border [CORRECT]

Summary:
- Original passrate: 1.0 (model not challenged)
- Final state: 3 effective distractors added
- The model now faces significantly higher
  difficulty, providing clearer reward signals
  for RLVR training
```

This case study demonstrates how the iterative curation framework systematically strengthens distractors through targeted generation and replacement, transforming a trivial question into a challenging learning signal for RLVR.

J Generalizability to Larger Models

To further validate the generalizability of our findings across different model scales, we conducted additional experiments using Qwen2.5-14B-Instruct as the base model. Table 8 presents the results of comparing different distractor generators for this larger model. The findings are consistent with the observations from Qwen2-7B and Llama-3.1-8B: self-generated distractors remain highly competitive. This confirms that the effectiveness of self-generated distractors is not limited to smaller models but extends to intermediate-scale models.

K Impact of Estimator Selection

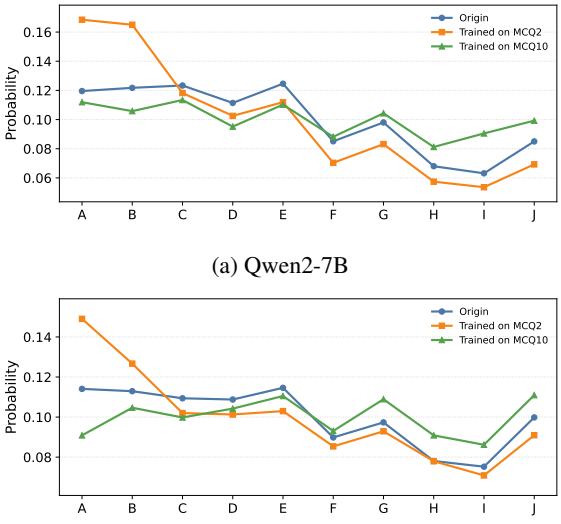
In our main method, we use the base model itself to estimate the empirical strength of distractors (Self-Estimation). To validate this design choice, we conduct an ablation study where we fix the generator model (Qwen2.5-32B-Instruct) but vary the model used for strength estimation (the "Estimator"). We compare two settings: (1) **Self-Estimation**: The model being trained is also used to estimate distractor strength (our default setting). (2) **Cross-Estimation**: A different model is used to estimate distractor strength (e.g., using Llama-3.1-8B to filter distractors for Qwen2-7B).

Table 9 presents the results. For Qwen2-7B, Self-Estimation achieves slightly better performance across most datasets. For Llama-3.1-8B, the results are mixed, with Cross-Estimation showing marginal gains on some tasks while Self-Estimation works better on others. Given that self-estimation eliminates the dependency on external models and simplifies the pipeline, we conclude that it serves as a sufficient and effective strategy

for filtering distractors.

Dataset	Train Qwen2-7B		Train Llama-3.1-8B	
	Est: Self	Est: Llama	Est: Self	Est: Qwen
MedQA	**49.80**	49.41	66.14	**67.71**
SuperGPQA	**27.91**	27.42	**35.71**	35.63
MMLU-Pro	**49.51**	48.04	**60.76**	60.39
MedXpertQA	**11.88**	11.59	**19.31**	18.45
PubMedQA	**74.00**	73.30	77.60	**78.50**
Average	**42.62**	41.95	51.90	**52.14**

Table 9: Ablation on the strength estimator. We fix the generator and compare different models for estimating distractor strength. "Self" indicates that the estimator is the same model as the one being trained (e.g., using Qwen2-7B to estimate for Qwen2-7B), while other names indicate cross-model estimation.



(a) Qwen2-7B

(b) Llama-3.1-8B

Figure 6: Option-label distributions under origin vs. `mcq2` vs. `mcq10` training. Both models show A/B bias after `mcq2` training, a near-uniform distribution after `mcq10`, and origin in between.

L Option-Label Distribution Analysis

Experiment Setting. We quantify option-label bias under controlled conditions. Starting from the 10-choice test split of MMLU-Pro, we permute the correct label so that it is uniformly distributed over A–J. For each item, we query the model with temperature 0.7 and draw 8 responses. We aggregate the selected labels across the entire test split to obtain the empirical distribution. Figure 6 overlays these distributions for three training strategies: the original model (origin), training on `mcq2`, and training on `mcq10`.

Findings. The pattern is consistent across Qwen and Llama. After `mcq2` training, both models assign disproportionate mass to early labels A/B, producing a front-loaded curve. After `mcq10` training, the curve flattens and approaches the uniform baseline. The origin models lie in between. This indicates a distributional mismatch: models adapt their output-label distribution to the option space seen during training, and neither fewer nor more options is inherently superior.

M Multi-Seed Results

To verify the statistical significance of IDC's gains, we conducted additional experiments with 3 independent random seeds for both the standard RLVR baseline (Orig.) and our IDC method. Table 10 reports mean ± std.

The results confirm that IDC provides consistent gains over the baseline across multiple runs, with non-overlapping intervals on most benchmarks.

Dataset	Qwen2-7B		Llama-3.1-8B	
	Orig.	IDC	Orig.	IDC
MedQA	47.16±0.29	**49.72±0.44**	59.01±0.38	**66.02±0.47**
SuperGPQA	25.53±0.40	**27.85±0.38**	33.24±0.36	**35.65±0.43**
MMLU-Pro	44.35±0.32	**49.43±0.49**	58.53±0.39	**60.70±0.48**
MedXpertQA	10.20±0.33	**11.82±0.37**	14.25±0.43	**19.22±0.43**
PubMedQA	69.39±0.40	**73.91±0.42**	77.92±0.44	77.58±0.48
Average	39.33±0.23	**42.55±0.24**	48.59±0.25	**51.83±0.24**

Table 10: Multi-seed results (mean ± std over 3 independent runs) for the standard RLVR baseline (Orig.) and IDC. IDC provides stable and consistent gains across runs.

N Mixed-Option Training

In Section 4, we showed that option-count alignment is important. A natural question is: what happens when training on a mixture of option counts? We construct a mixed training set from the original 10-choice MMLU-Pro questions, where each question is randomly assigned a variant with an equal proportion (20% each) of 2-, 4-, 6-, 8-, and 10-choice questions.

On both models, the mix strategy outperforms the single-strategy average. On Llama, mix is essentially on par with the best single strategy (mcq6). The mix strategy provides robust generalization without requiring prior knowledge of the test distribution, making it a practical default when the test format is unknown or heterogeneous.

Model	Train	Test-2	Test-4	Test-6	Test-8	Test-10	Avg
Llama-3.1-8B	mix	80.01	70.21	62.86	59.46	57.56	66.02
	Best single (mcq6)	79.57	69.58	**64.07**	59.21	57.87	66.06
	Single-strategy Avg	78.71	69.36	63.12	58.40	56.66	65.25
Qwen2-7B	mix	71.29	64.57	56.99	51.94	49.01	58.76
	Best single (mcq4)	**72.15**	**65.15**	**58.22**	51.92	48.57	**59.20**
	Single-strategy Avg	71.46	63.79	57.24	51.95	48.67	58.62

Table 11: Mixed-option training results on MMLU-Pro. "Single-strategy Avg" denotes the average over five single-option-count training runs (mcq2, mcq4, mcq6, mcq8, mcq10). The mix strategy outperforms the single-strategy average and provides robust generalization without prior knowledge of the test distribution.

Setting	Accuracy (mean $\pm$ std)
Base (Qwen2.5-14B-Instruct)	$61.72 \pm 1.51\%$
Orig. (standard RLVR)	$68.32 \pm 0.57\%$
IDC	$\mathbf{71.05 \pm 1.14\%}$

Table 12: IDC results on the law subset of MMLU-Pro using Qwen2.5-14B-Instruct (3-seed average).

Model	mcq2-test	mcq10-test
Qwen2.5-14B (base)	81.88	62.63
+ mcq2-train	**86.13**	62.92
+ mcq10-train	83.79	**67.94**

Table 13: Option-count mismatch persists in Qwen2.5-14B-Instruct. The mcq2-trained model excels on mcq2-test but shows minimal improvement on mcq10-test, and vice versa.

O Generalization to the Law Domain

To validate that IDC generalizes beyond the medical domain, we conducted an experiment on the law subset of MMLU-Pro ($\sim$1k samples, 85:5:10 train/dev/test split) using Qwen2.5-14B-Instruct. Due to the relatively small test set, we report results averaged over 3 independent runs (Table 12).

IDC yields a clear improvement over standard RLVR in the law domain (+2.73%), confirming that its benefits are not restricted to the medical domain.

P Option-Count Mismatch on Larger Models

To address whether option-count mismatch persists for larger models, we conducted additional experiments with Qwen2.5-14B-Instruct on MMLU-Pro (Table 13).

The mismatch pattern is consistent with our findings on smaller models: the mcq2-trained model excels on mcq2-test but barely improves on mcq10-test, while the mcq10-trained model shows the reverse pattern. This confirms that option-count mismatch is not mitigated by scaling up model size, as it stems from a fundamental property of LLMs rather than insufficient model capacity.

Q Algorithm Details

We provide the detailed algorithm for Iterative Distractor Curation in Algorithm 1.

Algorithm 1 Iterative Distractor Curation (IDC)

Input: Question stem q; Correct answer o^*; Original distractors $\mathcal{D}_{orig}$;
Generator model $\mathcal{M}_{gen}$; Evaluator model $\mathcal{M}_{eval}$;
Target option count $N = |\mathcal{D}_{orig}| + 1$; Max iterations T;

Output: Optimized distractor set $\mathcal{D}_{final}$.

1. **Initialization:**
   (a) Construct initial option set $\mathcal{C}_0 = \{o^*\} \cup \mathcal{D}_{orig}$.
   (b) Evaluate $\mathcal{C}_0$ using $\mathcal{M}_{eval}$ to obtain empirical strengths $\hat{s}(\cdot)$. Specifically, for each distractor $d \in \mathcal{D}_{orig}$, sample K trajectories and compute:

$$N_d = \sum_{k=1}^{K} \mathbb{I}[a_k = d,\, a_k \neq o^*], \quad N_{\text{err}} = \sum_{k=1}^{K} \mathbb{I}[a_k \neq o^*],$$

   then define $\hat{s}(d) = N_d / N_{\text{err}}$ if $N_{\text{err}} > 0$, otherwise $\hat{s}(d) = 0$.
   (c) Initialize effective pool $\mathcal{D} \leftarrow \{d \in \mathcal{D}_{orig} \mid \hat{s}(d) > 0\}$.
   (d) Initialize history $\mathcal{H} \leftarrow \{(\mathcal{D}, p_0)\}$.
2. **Iterative Optimization:** For $t = 1$ to T:
   (a) Let target distractor count $K_d = N - 1$.
   (b) **Mode 1: Filling** (If $|\mathcal{D}| < K_d$):
      i. Calculate needed count $k = K_d - |\mathcal{D}|$.
      ii. Generate k candidates $\mathcal{S}_{new}$ using $\mathcal{M}_{gen}(q, o^*, \text{exclude} = \mathcal{D})$.
      iii. Construct test set $\mathcal{C}_t = \{o^*\} \cup \mathcal{D} \cup \mathcal{S}_{new}$.
      iv. Evaluate $\mathcal{C}_t$ with $\mathcal{M}_{eval}$ to obtain strengths $\hat{s}_t(\cdot)$.
      v. Identify new effective items: $\mathcal{D} \leftarrow \mathcal{D} \cup \{d \in \mathcal{S}_{new} \mid \hat{s}_t(d) > 0\}$.
   (c) **Mode 2: Replacement** (If $|\mathcal{D}| \geq K_d$):
      i. Identify weakest link $d_{weak} = \arg\min_{d \in \mathcal{D}} \hat{s}(d)$.
      ii. Generate 1 candidate d_{new} using $\mathcal{M}_{gen}$.
      iii. Construct test set $\mathcal{C}_t = \{o^*\} \cup (\mathcal{D} \setminus \{d_{weak}\}) \cup \{d_{new}\}$.
      iv. Evaluate $\mathcal{C}_t$ with $\mathcal{M}_{eval}$ to obtain strengths $\hat{s}_t(\cdot)$.
      v. If $\hat{s}_t(d_{new}) > \hat{s}(d_{weak})$, then update pool:
      $\mathcal{D} \leftarrow (\mathcal{D} \setminus \{d_{weak}\}) \cup \{d_{new}\}$.
   (d) Record current state to history: $\mathcal{H} \leftarrow \mathcal{H} \cup \{(\mathcal{D}, p_t)\}$.
3. **Selection:**
   (a) Find maximum pool size $S_{max} = \max_{(\mathcal{D}, \cdot) \in \mathcal{H}} |\mathcal{D}|$.
   (b) Select version $(\mathcal{D}^*, p^*) \in \mathcal{H}$ where $|\mathcal{D}^*| = S_{max}$ and p^* is minimized.
   (c) Set $\mathcal{D}_{final} \leftarrow \mathcal{D}^*$.
   (d) While $|\mathcal{D}_{final}| < K_d$, fill with unreplaced options from $\mathcal{D}_{orig}$.
4. Return $\mathcal{D}_{final}$.