
RECURSIVE INTROSPECTION: Teaching Foundation Model Agents *How to Self-Improve*

Yuxiao Qu¹ Tianjun Zhang² Naman Garg³ Aviral Kumar¹

Abstract

A central piece in enabling intelligent agentic behavior in foundation models is to make them capable of introspecting upon their behavior, to reason and correct their mistakes. However, powerful proprietary large language models (LLMs) lack the ability to sequentially improve their responses, even when explicitly informed about their mistakes. In this paper, we develop **RISE: Recursive IntroSpEction**, an approach for fine-tuning LLMs to introduce this ability. Our approach prescribes an iterative fine-tuning procedure, which attempts to teach the model how to alter its response after having seen previously unsuccessful attempts to solve a problem with additional environment feedback. **RISE** poses fine-tuning for a single-turn problem as solving a multi-turn Markov decision process (MDP), where the initial state is the prompt. Inspired by principles in online imitation learning, we derive effective strategies to dictate multi-turn data collection and training so as to imbue an LLM with the capability to recursively detect and correct its previous mistakes in subsequent iterations. Our experiments show that **RISE** enables 7B Llama2 and Mistral models to improve themselves with more turns on math reasoning tasks, outperforming several single-turn strategies given an equal amount of inference-time computation. Our analysis shows that **RISE** makes meaningful improvements to responses to arrive at the correct solution for challenging prompts, without disrupting one-turn abilities.

1. Introduction

A promising approach for utilizing and deploying foundation models, and in particular, large language models (LLMs) is to treat them as general-purpose decision-making machines, or "agents", as it is commonly referred to in literature. To be successful, an LLM agent must not just

provide plausible completions for input text, but rather it must exhibit interactive, goal-directed behavior to accomplish a given task. This requires mastering two qualities: (a) producing responses that explicitly seek out information about the task, followed by (b) making decisions and improving them by "thinking" and verifying them, by accounting for external input. For instance, to succeed at using a new coding library, an effective LLM agent should first synthesize programs, then try the most promising subset against a compiler, use the resulting feedback to improve the program, and repeat the process for multiple turns. Being able to successfully improve a response over multiple turns is equivalent to a form of "self-improvement", over the course of interaction with the external world.

To enable inference-time self-improvement, recent approaches attempt to re-purpose the knowledge already stored in pre-trained models via few-shot prompting (Gou et al., 2023; Madaan et al., 2023; Wei et al., 2022; Zhang et al., 2024; Chen et al., 2023b). While prompt tuning in conjunction with feedback is effective in eliciting improved responses from capable models, it falls short in producing models that can succeed in complex tasks by correcting their own mistakes, such as those that require logical reasoning (Huang et al., 2023b; Yang et al., 2023b). To address this issue, some work fine-tunes LLMs with data from the downstream task (Schick et al., 2023; Chen et al., 2023a; Liu et al., 2023). While fine-tuning does utilize "oracle" responses to in-domain prompts, it still does not induce an improvement strategy (see Section 6).

Can we train models to be capable of improving their own responses? If done correctly and on a diverse set of problems and scenarios, this could introduce in an LLM, a general procedure for test-time self-improvement. While one straightforward approach to do to induce this into a model would be to generate data showcasing improvements over multiple sequential turns (potentially from highly capable models), we find that simply imitating this data is not sufficient at enabling this capability (Section 6.3). Largely, this is due to two reasons: first, multi-turn interaction from different models does not showcase improvements on the kinds of errors that the learner would make thereby being irrelevant or undecipherable to the learner (Kang et al., 2024), and second, proprietary models are typically not good at proposing meaningful improvements to their own

¹Carnegie Mellon University ²UC Berkeley ³Multion Inc.. Correspondence to: Yuxiao Qu <yuxiaoq@andrew.cmu.edu>. Accepted by the *Structured Probabilistic Inference & Generative Modeling workshop* of ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

errors (Huang et al., 2023b), but can still provide useful responses to the problem at hand. Therefore, we need a different strategy to endow models with a self-improvement capability. Our key insight is to supervise improvements to the learner’s own responses in an iterative fashion, taking inspiration from methods in online imitation learning (Ross et al., 2011). This supervision can be oracle responses to the prompt, or be generated from the learner itself.

Our contribution is an algorithm, **RISE: Recursive Introspection**, utilizing these insights to improve an LLM’s self-improvement ability over multiple interaction turns. In each iteration, our approach bootstraps on-policy rollouts from the learner with better responses at the next turn obtained by running best-of-N (using a success indicator on the task) on multiple revision candidates obtained by sampling from the learner itself or using responses from a capable model, whichever is more convenient. This way, we are able to construct rollouts that demonstrate the learner how it can improve its responses under its own distribution. Then, we fine-tune the learner on this data using a reward-weighted regression (RWR (Peters & Schaal, 2007)) objective, that is able to learn from both high- and low-quality parts of such rollouts. By iteratively repeating this procedure, we are able to instill a general self-improvement algorithm into an LLM. Our results show that LLMs trained via RISE can produce correct responses on more prompts, improving over turns for more challenging prompts.

Even though we find that strong LLMs (Jiang et al., 2023; Yuan et al., 2024a) often fail to revise their own responses over multiple turns of interaction, **RISE** successfully endows similarly-sized LLMs with self-improvement capabilities, resulting in monotonically increasing task performance after each turn. Specifically, on a common mathematical reasoning benchmark, GSM8K (Cobbe et al., 2021), RISE improves the performance of a fine-tuned LLaMa2-7B model by 17.7% over the course of 5-turn introspection, improving over parallel sampling at the first turn; a Mistral-7B model by 23.9%; whereas GPT-3.5 only improves by 4.6% over five turns. Similar trends hold on MATH (Hendrycks et al., 2021), where RISE improves the LLaMa2-7B model by 4.6% and a Mistral-7B model by 11.1% over five turns, with no oracle guidance.

2. Related Work

Several prior works build techniques to improve reasoning and thinking capabilities of foundation models for downstream applications. Typically these works focus on building prompting techniques for effective multi-turn interaction with external tools (Yang et al., 2023a; Chen et al., 2023b; Charalambous et al., 2023; Nijkamp et al., 2023; Yao et al., 2022; Wang et al., 2023a; Gao et al., 2023), sequentially refining predictions by reflecting on actions (Chen et al., 2023b; Gou et al., 2023; Zhang et al., 2022), asking the

model to verbalize its thoughts (Wei et al., 2022; Nye et al., 2021; Zhou et al., 2023), asking the model to critique and revise itself (Madaan et al., 2023; Shinn et al., 2023) or by using other models to critique a primary model’s responses (Du et al., 2023; Yang et al., 2023a; Bai et al., 2022; Huang et al., 2023a). While a subset of this work does improve self-improvement abilities, this self-correction ability often requires access to detailed error traces (e.g., execution traces from code compilers (Madaan et al., 2023; Chen et al., 2023b)) in order to succeed. In fact, (Huang et al., 2023b) and Table 1 both indicate that self-improvement guided by the LLM itself (i.e., “intrinsic self-correction”) is often infeasible for off-the-shelf LLMs, **but fine-tuning with RISE induces this capability** as we show in this paper.

Beyond prompting, prior works also attempt to fine-tune LLMs to obtain self-improvement capabilities (Schick et al., 2023; Chen et al., 2023a; Zeng et al., 2023). These works attempt to improve reasoning performance by training on self-generated responses (Yuan et al., 2024a; Yue et al., 2023; Luo et al., 2023; Toshniwal et al., 2024; Yu et al., 2023). To achieve this, these works use a combination of learned verifiers (Wang et al., 2023b; Lightman et al., 2023; Uesato et al., 2022), search (Lehnert et al., 2024; Nye et al., 2021; Gandhi et al., 2024; Saha et al., 2023), contrastive prompting on negative data (Chia et al., 2023; Wang et al., 2022a), and iterated supervised or reinforcement learning (RL) (Chen et al., 2024; Yuan et al., 2024b; Rosset et al., 2024). While our approach also trains on model-generated data, we aim to introduce a complementary capability of improving performance over sequential turns of interaction, instead of improving single-turn performance alone. Other work fine-tunes LLMs for multi-turn interaction directly via RL (Snell et al., 2022; Zhou et al., 2024): while this is indeed related to us, single-turn problems posed in multi-turn scenarios require addressing distinct challenges than generic multi-turn RL: (i) sample-efficiency is not a concern since the entire MDP is fully characterized by the training dataset of prompts and oracle answers, and (ii) we need to generalize to novel test prompts. Multi-turn RL focuses on sample efficiency, which is not critical in our setting. More generally, our main focus is to show that it is possible to train models for self-improvement via appropriately designing multi-turn fine-tuning objectives. This is orthogonal from the choice of RL, contrastive or supervised learning approach used for optimizing the said objective.

Perhaps the most related to our work, are GLoRE (Havrilla et al., 2024) and Self-Correct (Welleck et al., 2023), which train separate models to identify errors and refine incorrect answers of other LLMs. Unlike these works, our approach trains a single model to produce answers and improve them over more than two turns, which is the maximal number of turns studied in these works. We show that doing so successfully requires a careful design choices: an iterative (i.e.,

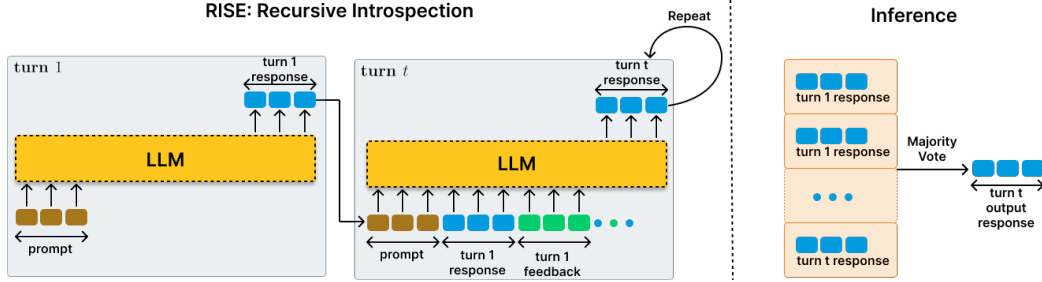


Figure 1: *Recursive Introspection (RISE)*. Using iterative multi-round training on on-policy rollouts and supervision from a reward function, RISE trains models that are capable of improving themselves over multiple turns. At inference, we run majority voting on candidate outputs from different turns to obtain the final response.

STaR (Zelikman et al., 2022)-like) on-policy data generation strategy along with a training objective that can learn from both successful and unsuccessful rollouts.

From an algorithmic standpoint, our work resembles online imitation learning (Ross et al., 2011; Sun et al., 2018), in that it queries expert supervision on states attained by on-policy rollouts. On-policy distillation for LLMs (Agarwal et al., 2023; Chang et al., 2024) utilizes this principle, but queries an expert to provide completions on partially-generated responses instead of sequentially proposing revisions with the reward signal that we utilize. We also emphasize that our goal is not necessarily to propose a novel learning objective, but to show that even the simple reward-weighted RL can endow the model with self-improvement capabilities.

3. Problem Setup and Preliminaries

The goal of our work is to improve LLM performance over sequential turns. Concretely, given a dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i^*)\}_{i=1}^N$ of problems \mathbf{x}_i and oracle responses \mathbf{y}_i^* , our goal is to obtain an LLM $\pi_\theta(\cdot | [\mathbf{x}, \hat{\mathbf{y}}_{1:t}, p_{1:t}])$ that, given the problem \mathbf{x} , previous model attempts $\hat{\mathbf{y}}_{1:t}$ at the problem, and auxiliary prompts $p_{1:t}$ (e.g., instruction to find a mistake and improve the response; or additional compiler feedback from the environment) solves the problem in the fewest number of turns. Formally,

$$\max_{\pi_\theta} \sum_{i=1}^L \mathbb{E}_{\mathbf{x}, \mathbf{y}^* \sim \mathcal{D}, \hat{\mathbf{y}}_i \sim \pi_\theta(\cdot | [\mathbf{x}, \hat{\mathbf{y}}_{1:i-1}, p_{1:i-1}])} \mathbb{I}(\hat{\mathbf{y}}_i == \mathbf{y}^*) \quad (3.1)$$

Note that, unlike standard supervised fine-tuning that trains the model π to produce a single response $\hat{\mathbf{y}}$ given \mathbf{x} , Equation 3.1 trains π to also appropriately react to a given history of responses coming its own previous attempts $\hat{\mathbf{y}}_{1:i-1}$. Equation 3.1 most closely resembles an RL objective and we will indeed develop our approach by converting a single-turn problem into a multi-turn MDP. Finally note that prompting-based methods such as Self-Refine (Madaan et al., 2023) can still be viewed as training π to optimize $\pi(\mathbf{y}^* | \mathbf{x})$ but when only allowed to modulate the prompt p_i to optimize Equation 3.1. Naturally, since the parameters θ are unchanged, this would not be effective at optimizing the objective fully.

4. RISE: Recursive Introspection

Having seen that even strong off-the-shelf models do not exhibit an effective ability of improving themselves when

provided with sequential attempts at a given problem, a natural next step is to ask how to train for a self-improvement ability. In this section, we will develop our approach, **RISE**, for fine-tuning foundation models towards improving their own predictions over multiple turns. Our approach will first convert a problem into a multi-turn MDP, then collect data, and finally run offline reward-weighted supervised learning in this multi-turn MDP to induce this capability.

4.1. Converting Single-Turn Problems into a Multi-Turn Markov Decision Process (MDP)

The first step in building our approach **RISE** involves procedurally constructing a multi-turn MDP using a single-turn dataset of prompts and oracle responses. Given a dataset, $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i^*)\}$, consisting of prompts \mathbf{x}_i and corresponding oracle responses \mathbf{y}_i^* (e.g., math questions and natural language responses to those questions), we will construct an *induced* MDP \mathcal{M} from \mathcal{D} , and then learn policies that optimize multi-step return in this MDP. An initial state in this MDP is a possible prompt $\mathbf{x}_i \in \mathcal{D}$. We denote the output response from the foundation model as action \mathbf{a} . Given a state s , the next state can be obtained by concatenating the tokens representing s with the action \mathbf{a} proposed by the model, and an additional fixed prompt \mathbf{f} that asks the model to introspect, e.g., “*this response is wrong, please introspect and correct your answer.*”. The reward function is a sparse binary indicator of success for a state s , $r([\mathbf{x}_i, \dots], \mathbf{a}) = 1$ if and only if $\mathbf{a} = \mathbf{y}_i^*$. The discount factor γ can be adjusted based on the number of turns T in the conversation. This construction $\mathcal{D} = (\mathbf{x}_i, \mathbf{y}_i^*) \rightarrow \mathcal{M}$ is formalized as:

$$\rho(s_0) = \text{Unif}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N), \quad (4.1)$$

$$P(s' | s, \mathbf{a}) = \delta(s' = \text{concat}[s, \mathbf{a}, \mathbf{f}]), \quad (4.2)$$

$$r(s, \mathbf{a}) = \mathbf{1}(\mathbf{a} = \mathbf{y}_i^* \text{ if } \mathbf{x}_i \in s). \quad (4.3)$$

4.2. Learning in the Multi-Turn MDP

With the MDP construction in place, the next step involves training a model to improve itself over the course of a rollout. We subscribe to an offline approach for learning that we describe below.

Step 1: Data collection for self-improvement. To ensure

that the offline rollout data from this multi-turn MDP is useful for imbuing a self-improvement capability into the model, it must satisfy a few desiderata: **(1)** it must illustrate the mistakes that the learner makes and showcase how to improve upon them in the next turn, **(2)** the data must illustrate responses that are relevant to the model given the problem and previous attempts in context, and **(3)** it must not contain any rollout that degrades in a subsequent turn. Our data collection strategy that satisfies these desiderata.

At a given round k , for a given problem x_i , we unroll the *current* model $\pi_{\theta_k}(\cdot|\cdot)$ to produce multiple sequential attempts, denoted by $\mathbf{y}_t^i \sim \pi_{\theta_k}(\cdot|s_t^i)$. In problems, where external input is available, we also observe a variable-length, natural language external input, f_t^i (e.g., in math problems we ask the model to correct itself). We also observe a scalar reward value $r(s_t^i, \mathbf{y}_t^i)$, denoted as r_t^i in short. Let us denote this dataset of “on-policy” model rollouts as $\mathcal{D}_{\text{on-policy}} := \{(s_t^i, \mathbf{y}_t^i, f_t^i, r_t^i)_{t=1}^T\}$.

For each time-step, we construct an improved version of the response \mathbf{y}_t^i that we will denote by $\tilde{\mathbf{y}}_t^i$. We also record the reward score associated with this improved response as $r(s_t^i, \tilde{\mathbf{y}}_t^i)$, or \tilde{r}_t^i in short. To obtain an improved version of a response \mathbf{y}_t^i , we can employ several strategies. Perhaps the most straightforward approach is to query an off-the-shelf more capable model to provide a correct response given the prompt x_i , previous response \mathbf{y}_t^i , and an optional external feedback f_t^i . We refer to this as the **distillation** variant of our approach, since it uses a strong “teacher” model to guide self-improvement (and hence, distills knowledge from this teacher model into the learner).

$$\tilde{\mathcal{D}}_{\text{on-policy} + \text{distill}} := \left\{ \left\{ (s_t^i, \tilde{\mathbf{y}}_t^i, f_t^i, \tilde{r}_t^i) \right\}_{t=1}^T \right\}_{i=1}^{|\mathcal{D}|}. \quad (4.4)$$

The second variant of our approach which alleviates the need for a teacher model, involves constructing an improved response by sampling multiple times from the learner itself. Concretely, for each state in the dataset, $s_t^i \in \mathcal{D}_{\text{on-policy}}$, we sample N responses $\tilde{\mathbf{y}}_t^i[0], \tilde{\mathbf{y}}_t^i[1], \dots, \tilde{\mathbf{y}}_t^i[N] \sim \pi_{\theta}(\cdot|s_t^i)$, and use the best response from these N candidates (as measured by the associated reward values $\tilde{r}_t^i[0], \dots, \tilde{r}_t^i[N]$) to relabel the model response at the next step $t + 1$ in an improvement trajectory. Formally, say $\tilde{\mathbf{y}}_t^i[m] = \arg \max_{j \in [N]} r(s_i, \tilde{\mathbf{y}}_t^i[j])$, then, we relabel the responses in the dataset $\mathcal{D}_{\text{on-policy}}$ at step $t + 1$ with the improved response and its associated reward value $\tilde{r}_t^i[m]$:

$$\tilde{\mathcal{D}}_{\text{on-policy} + \text{self-improvement}} := \left\{ \left\{ (s_{t+1}^i, \tilde{\mathbf{y}}_t^i[m], f_{t+1}^i, \tilde{r}_t^i[m]) \right\}_{t=0}^{T-1} \right\}_{i=1}^{|\mathcal{D}|}. \quad (4.5)$$

Step 2: Policy improvement. With the aforementioned data construction schemes, we can now train a model on these datasets. While in general, any offline RL approach can be used to train on this data, in our experiments we adopt an approach based on weighted supervised learning (Peters &

Schaal, 2007) due to ease of experimentation. In particular, we run weighted imitation learning, where weights are given by the exponential transformation of the reward values in either $\tilde{\mathcal{D}}$:

$$\max_{\theta} \mathbb{E}_{x_i \sim \tilde{\mathcal{D}}} \left[\sum_{t=1}^T \log \pi_{\theta}(\tilde{\mathbf{y}}_t^i | s_t^i) \cdot \exp(r_t^i / \tau) \right] \quad (4.6)$$

τ is a temperature parameter to further expand or narrow the difference between good and bad actions. In our preliminary experiments we found that Equation 4.6 can often induce a bias towards increasing log likelihoods of responses where rewards are high, prioritizing updates on easy problems where rewards are already high. To address this issue, we apply a slight modification to Equation 4.6 and center the exponentiated rewards around the mean value averaged across all attempts on a given prompt, akin to advantage-weighted regression (Peng et al., 2019). We find that the use of advantages in place of rewards helps us avoid the “rich-gets-richer” phenomenon with easy problems.

4.3. Inference at Deployment Time

RISE can be run in two modes at inference time. Perhaps most straightforward to run the policy $\pi_{\theta}(\cdot|\cdot)$ trained by RISE is within a multi-turn rollout, where the model samples a new response conditioned on the past context (i.e., state in the multi-turn MDP). This past context consists of the external feedback p_i^{test} concerning the response $\mathbf{y}_i^{\text{test}}$ and the rollout terminates as soon as $r(x, \mathbf{y}_i^{\text{test}}) = r(x, \mathbf{y}^*)$. This protocol invokes queries to the reward function after each turn in the rollout. Since several reward function queries are performed, we refer to this approach as “**with oracle**”.

RISE can also be run in a mode that avoids the need for querying the reward function within a rollout. In this case, we run full-length rollouts and utilize a self-consistency mechanism (Wang et al., 2022b) based on majority voting to decide the candidate response at the end of each turn. Concretely, at the end of each turn j , we identify the response by running a majority vote over all response candidates from the previous turns ($\text{maj}(\mathbf{y}_0^{\text{test}}, \mathbf{y}_1^{\text{test}}, \dots, \mathbf{y}_j^{\text{test}})$), including turn j . We call this “**without oracle**”.

4.4. Practical Algorithm and Implementation Details

We trained 7B models via RISE and found that these models often could not adhere to response style and instructions for improving their responses when generating on-policy data. As a result, before running on-policy data collection, we find it often useful to run an initial phase of supervised fine-tuning on in-domain, multi-turn rollouts generated from a capable model to provide style and instruction-following information to the learner. We call this the “knowledge boosting” stage. We then run on-policy rollouts starting from a boosted model. In each iteration, we generate 1 trajectory for each unique problem. We then run fine-tuning, with hyperparameters and details in Appendix C. We adopt

the scheme of Zelikman et al. (2022) and iteratively fine-tune starting from the *base* model.

5. When and Why is Self-Improvement Over Turns Possible?

A natural question to ask is why is self-improvement with **RISE** even possible. One might surmise that the model may simply not have enough knowledge to correct its *own* mistakes if it is unable to correctly answer the problem in the first turn. Then, why is it possible to teach the model to correct its own mistakes? In this section, we provide two reasons why this kind of self-improvement is possible, supported with empirical evidence to justify our hypotheses.

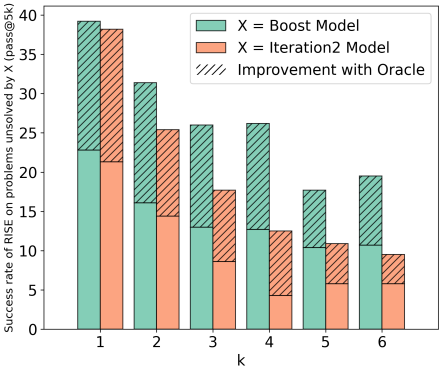


Figure 2: **Fraction of problems unsolved by pass@B at the first turn that sequential 5-turn sampling from RISE solves**, where $B = 5 \times k$ (k is the x-axis). RISE can solve several challenging problems that sampling at the first turn with much larger budgets cannot solve.

Reason 1: Bounded-capacity models & flexible test-time computation. Iteratively teaching a model how to make updates on a given response is important when representing the target distribution $p^*(\mathbf{y}|\mathbf{x})$ requires more capacity than what the model π_θ has. In this case, learning a sequence of conditionals, $\pi_\theta(\mathbf{y}_{i+1}|\mathbf{x}, \mathbf{y}_{0:i})$ followed by marginalization is expected to induce a more flexible marginal distribution over \mathbf{y}_T given \mathbf{x} . We tracked the training perplexity (loss) of directly fitting oracle answers and compared it to the perplexity of fitting a sequence of answers in Figure 8, and found that the latter indeed does reduce the loss more than the oracle response directly, thus supporting our hypothesis. Of course, in problems which require “knowledge-based” question answering, it is not possible for the model to produce any meaningful improvements because learning $p^*(\mathbf{y}|\mathbf{x})$ is not bounded by insufficient capacity of $\pi_\theta(\mathbf{y}|\mathbf{x})$, but rather due to the absence of features that are critical to learn a mapping from \mathbf{x} to \mathbf{y} . We expect that training with RISE would only incentivize hallucinations in this case (Kang et al., 2024). However, this is not the failure mode on reasoning problems (Li et al., 2024), where maj@K rates at turn 1 tend to be higher than pass@1. In fact, Figure 2 shows that the sequential procedure learned by RISE can even solve a significant fraction of problems that were unsolved by pass@B for much larger B in the first turn.

Reason 2: Implicitly learning a model of correctness. Unlike Huang et al. (2023b), the reason why self-improvement over turns is possible for us is because training with RISE can *implicitly* instill a verification procedure into the learner: training on data in Equation 4.4 should in principle make the model capable of deducing whether or not a given input response is correct using simple heuristics, for example, evaluating the edit distance between the input and output. The emergence of such test-time elimination or verification strategies by training on on-policy data has been conceptualized under the notion of learning generalizable strategies in the multi-turn MDP (Ghosh et al., 2021), and indeed, such test-time procedures learned by RISE can generalize to out-of-distribution prompts (see Appendix A.2).

6. Experimental Evaluation

The goal of our experiments is to demonstrate the efficacy of RISE in imbuing language models with the ability to self-improve their responses over turns. Our experiments answer the following questions: (1) How effectively can RISE improve performance over turns? (2) Does the self-improvement behavior learned by RISE generalize to novel problems that are out of the training domain? (3) Does the performance of RISE improve with iterative training? and finally, (4) How to best collect data to train RISE? To this end, we compare RISE to other approaches on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021).

Baselines, comparisons, and evaluation. We compare RISE to several prior methods that attempt to induce similar self-improvement capabilities: (a) GloRE (Havrilla et al., 2024), which trains a separate reward model to locate errors and a refinement model to improve responses of a base LLM; (b) self-refine (Madaan et al., 2023; Huang et al., 2023b) that prompt a base model to critique and revise its mistakes, and (c) self-consistency (Wang et al., 2022b) on multiple responses from the first turn. We tried to construct fair comparisons between RISE and these works by using a similar-sized model (Yuan et al., 2024a; Jiang et al., 2023), but differences in the base model, training data, and evaluation setups prohibit us from performing an apples-to-apples comparison. Nonetheless, we can still hope to understand the ballpark of improvement by contextualizing our results with these prior works. We also compare to V-STaR (Hosseini et al., 2024), but since this is not a fair comparison, we defer it to Appendix A.

We evaluate RISE in both modes at inference time: with and without an oracle (Section 4.3), at the end of five turns: while the run with the oracle terminates a rollout as soon as the response is correct (“p1@t5”), w/o oracle does not terminate a rollout earlier than five turns (“m1@t5”). We also compare maj@K performance at the first turn for all the models we train (“m1@t1”, “m5@t1”).

Table 1: **RISE vs. other approaches (Self-Refine, GLoRE) and baselines.** Observe that RISE attains the biggest performance improvement (in brown) between 1-turn (m5@t1) and 5-turn (m1@t5) performance w/o an oracle on both GSM8K and MATH. This performance gap is even larger when oracle early termination is allowed (p1@t5 w/ oracle). Self-Refine (Madaan et al., 2023) largely degrades performance across the board. GLoRE trains a separate refinement model, but still performs worse than RISE; more details about it are in Appendix A. Using RISE on top of a better base model (Mistral-7B) is also effective (positive improvements with multiple turns), and note the m1@t5 performance of Mistral-7B exceeds even state-of-the-art math models such as Eurus-7B-SFT (Yuan et al., 2024a). Color coding indicates performance and improvement numbers that can be compared to each other.

Approach	GSM8K (Cobbe et al., 2019)				MATH (Hendrycks et al., 2021)			
	w/o oracle			w/ oracle	w/o oracle			w/ oracle
	m1@t1	→ m5@t1	→ m1@t5	p1@t5	m1@t1	→ m5@t1	→ m1@t5	p1@t5
RISE (Ours)								
Llama2 Base	10.5	22.8 (+12.3)	11.1 (+0.6)	13.9 (+3.4)	1.9	5.1 (+3.2)	1.4 (-0.5)	2.3 (+0.4)
+Boost	32.9	45.4 (+12.5)	39.2 (+6.3)	55.5 (+22.6)	5.5	6.8 (+1.3)	5.5 (0.0)	14.6 (+9.1)
+Iteration 1	35.6	49.7 (+14.1)	50.7 (+15.1)	63.9 (+28.3)	6.3	8.8 (+2.5)	9.7 (+3.4)	19.4 (+13.1)
+Iteration 2	37.3	51.0 (+13.7)	55.0 (+17.7)	68.4 (+31.1)	5.8	10.4 (+4.6)	10.4 (+4.6)	19.8 (+14.0)
RISE (Ours)								
Mistral-7B	33.7	49.4 (+15.7)	39.0 (+5.3)	46.9 (+13.2)	7.5	13.0 (+5.5)	8.4 (+0.9)	13.0 (+5.5)
+ Iteration 1	35.3	50.6 (+15.3)	59.2 (+23.9)	68.6 (+33.3)	6.7	9.5 (+2.8)	18.4 (+11.1)	29.7 (+22.4)
7B SoTA								
Eurus-7B-SFT	36.3	66.3 (+30.0)	47.9 (+11.6)	53.1 (+16.8)	12.3	19.8 (+7.5)	16.3 (+4.0)	22.9 (+10.6)
Self-Refine			→ m1@t3	→ p1@t3			→ m1@t3	→ p1@t3
Base	10.5	22.4 (+11.9)	7.1 (-3.4)	13.0 (+2.5)	1.9	5.1 (+3.2)	1.9 (0.0)	3.1 (+1.2)
+Iteration 2	37.3	50.5 (+13.2)	33.3 (-4.0)	44.5 (+7.2)	5.8	9.4 (+3.6)	5.7 (-0.1)	9.5 (+3.7)
GPT-3.5	66.4	80.2 (+13.8)	61.0 (-5.4)	71.6 (+5.2)	39.7	46.5 (+6.8)	36.5 (-3.2)	46.7 (+7.0)
Mistral-7B	33.7	48.5 (+14.8)	21.2 (-12.5)	37.9 (+4.2)	7.5	12.3 (+4.8)	7.1 (-0.4)	11.4 (+3.9)
Eurus-7B-SFT	36.3	65.9 (+29.6)	26.2 (-10.1)	42.8 (+6.5)	12.3	19.4 (+7.1)	9.0 (-3.3)	15.1 (+2.8)
GLoRE			→ m1@t3	→ p1@t3				
+ORM	48.2		49.5 (+1.3)	57.1 (+8.9)	Not studied in (Havrilla et al., 2024)			
+SORM	48.2		51.6 (+3.4)	59.7 (+11.5)				
+Direct	48.2		47.4 (-0.8)	59.2 (+11.0)				

6.1. Does RISE improve performance over multiple turns compared to other approaches?

We present the comparisons in Table 1. First, note that RISE (“Iteration 1” and “Iteration 2”) boosts up the Llama2-based model’s five-turn performance by 15.1% and 17.7% respectively on GSM8K and 3.4% and 4.6% on MATH w/o any oracle. Interestingly, we found using prompting-only Self-Refine (Madaan et al., 2023) largely degrades performance across the board, even with a strong proprietary model, GPT-3.5. The strongest 7B base models, Mistral-7B and Eurus-7B-SFT (Yuan et al., 2024a), when coupled with standard prompting are also only able to improve their performance, but only by 5.3% / 11.6% and 0.9% / 4.0% respectively on GSM8K and MATH, which is significantly lower than our approach. The performance of GLoRE improves by only 3.4% on GSM8K (over two turns), but this is still lower than our approach that improves by 6.3% in two turns and 13.4% in three turns. This indicates that RISE is effective in attaining in teaching models how to improve their own errors.

Can RISE effectively make use of mistakes and correct them? One concern that arises in prior works is whether the model can truly correct itself over turns or whether the improvement comes from the effect of sampling more answers and picking the best one. In Table 1, we see that sequentially improving responses via RISE (“maj@1@turn5”)

outperforms sampling 5 responses in parallel at the 1-turn and applying a majority vote on them (“maj@5@turn1”). In particular, comparing maj@5 performance at the end of ‘1-turn and ‘5-turn, we observe a consistent 4% to 8% improvement on GSM8K and an 6.5% improvement on MATH (with Mistral-7B model). This means that RISE can imbue models with a self-improvement ability while the simple SFT, followed by parallel sampling cannot endow the same ability. In addition, we also observe that running multiple iterations of RISE still preserves the first turn performance while improving the 5-turn performance.

How does the base model affect RISE? The performance of RISE with Llama2-7B on an absolute scale is lower than the best models specifically fine-tuned on math data (e.g., Eurus-7B-SFT or Mistral-7B). However, we find that RISE is still effective on top of Mistral-7B base model. In fact, *our performance at the end of 5-turns outperforms one of the best 7B SFT models, customized to reasoning.* Compare the m1@t5 performance of Eurus-7B-SFT and Mistral-7B in RISE (ours).

Self-improvement version of RISE. We also compare the performance of RISE with entirely self-generated data and supervision (Equation 4.4, $N = 16$) after one iteration directly on top of the more capable Mistral-7B model on GSM8K in Table 2, without any boosting phase. We find

Table 2: *RISE with self-generated data on GSM8K*. RISE is able to improve 5-turn maj@1 performance of the model with entirely self-generated data and supervision, despite the base Mistral-7B model not producing correct answers for several problems.

RISE (Self)	w/o oracle			w/ oracle
	m1@t1	→ m5@t1	→ m1@t5	p1@t5
Mistral-7B	33.7	49.4 (+15.7)	39.0 (+5.3)	46.9 (+13.2)
+ Iteration 1	36.8	44.4 (+7.6)	39.5 (+6.6)	48.7 (+15.9)

that this variant is also able to improve five-turn performance of the base Mistral-7B model. Though, with no boosting, this approach is limited by its ability to generate sufficiently diverse rollouts since the base Mistral-7B model is unable to answer more than 66% of the GSM8K training questions correctly. Nonetheless, training via RISE does enhance multi-turn self-improvement capabilities, entirely on its own.

6.2. Does the Performance of RISE Improve with Iterative Training?

Next, we attempt to understand if RISE improves with multiple rounds of training on on-policy data. As shown in Tables 1 and 2, the performance of RISE improves from iteration to iteration constantly. The 5-turn performance of RISE, both with and without an oracle, exhibits a clear improvement with more rounds across all domains and models. This implies that iterative self-training procedures of the form of STaR (Zelikman et al., 2022) can also be combined with RISE for training models for self-improvement.

6.3. What Data Compositions and Data Quantity are Crucial for RISE?

We now study how different data compositions affect the performance of RISE: should we collect on-policy error correction data like DAgger (Ross et al., 2011) or should we bias towards high-quality off-policy data.

There are three key aspects: (a) use multi-turn rollout data for fine-tuning, (b) use weighted supervised fine-tuning compared to naïve supervised learning, which only utilizes successful rollouts for fine-tuning; and (c) use on-policy rollouts and self-generated or oracle data. We will study these aspects one by one.

(a) **Data composition for fine-tuning.** We first study the necessity of using the interaction of error correction history for training RISE in Figure 3 (Left). We compare two approaches: model trained with oracle answers shown right after the first turn and oracle answers shown after intermediate failed attempts in Figure 3 (Left). Even though the latter trains on intermediate responses that may not always be correct, it attains a higher performance than simply training on the correct response for a given prompt. This highlights the importance of training on contexts that include a multi-turn interaction history depicting mistakes from the learner to

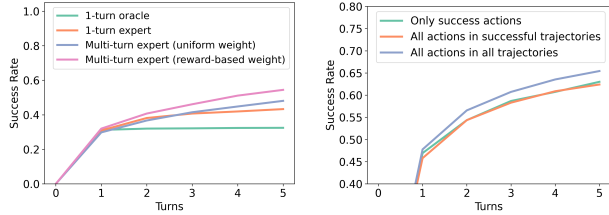


Figure 3: *Left: The importance of multi-turn interaction history and unweighted objectives for training RISE*. Note that training with multi-turn data leads to better self-improvement performance at the end of 5 turns, than one-turn data obtained from the original dataset with human answers or with oracle answers from another model; also observe that using a weighted objective performs better. *Right: The importance of using all rollouts for learning*, instead of only successful rollouts or only successful responses in the data. Using all data performs best.

improve self-improvement capabilities.

(b) **Weighted supervised learning vs unweighted supervised learning.** Next, we investigate the effect of reward-weighted RL on multi-turn data in RISE as opposed to simply imitating filtered successful data. We find that using all the data leads to improved performance over simply filtering good data in Figure 3 (Right), which reduces sample size. In Figure 3 (Left), we find that reward-weighted training improves performance, allowing us to better leverage all the sub-optimal data.

(c) **On-policy vs off-policy data; self-generated vs expert data.** RISE runs on-policy rollouts and seeks improvements on responses that the learner produces. As shown in Figure 4 (Left), a “DAGger (Ross et al., 2011)”-style approach that seeks improvements on responses appearing in on-policy rollouts improves performance (orange) compared to using the expert data alone (blue/pink). Conceptually, this addresses the train-test mismatch between the distribution of context tokens, enabling imitation learning methods to now target the correct distribution. In addition, recent work (Kang et al., 2024) has shown that LLMs often memorize “unfamiliar” examples generated by oracle models; by training on on-policy rollouts, we are able to eliminate such cases. Thus, while the model trained via offline imitation is able to reduce loss, these improvements do not generalize to new problems. In addition, we find in Figure 4 (Left) that while utilizing oracle responses from an expert is the most effective, training on self-generated data (“best-of-n”) is also effective. Even with $N = 16$, we are able to improve multi-turn performance of the learner.

6.4. Pass@N vs Sequential Sampling via RISE

We now study the performance of sequential sampling with oracle feedback in GSM8K, unlike relying on the model’s notion of self-consistency as in Table 1. Specifically, we compare the performance of RISE with early termination of evaluation rollouts against pass@5 performance of the RISE model at the first turn (which makes an equal number of queries to the ground-truth correctness indicator). Access

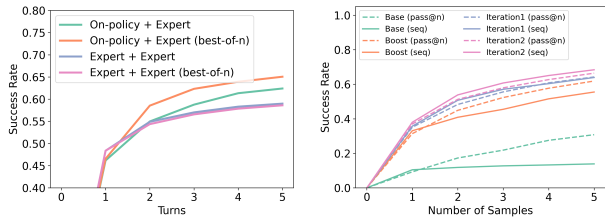


Figure 4: *Left: The importance of the data sources used for training.* We study the performance of the iteration 1 of RISE on GSM8K with different data sources. “Expert” refers to the use of an oracle model, “On-policy” corresponds to sampling from the learner, and “Best-of-N” means using the best sample out of N from the learner (here $N = 16$). *Right: Comparing RISE with oracle error feedback (pass@1 @ turn k ; solid lines) to parallel sampling of 5 responses at turn 1 (pass@ k @ turn 1; dashed lines) over number of turns k on the x-axis on GSM8K.* Observe that sequential sampling with Iteration 1 and Iteration 2 RISE models consistently outperforms parallel sampling for all values of turn k ; and the gap grows as the number of iterations increases. In contrast, this trend is absent for base and SFT models.

to ground-truth correctness indicator is expected to improve performance for both parallel and sequential sampling as expected, but we see in Figure 4 (Right) that RISE is able to improve performance more beyond simply sampling more samples at the first turn and computing pass@ K .

6.5. Error Analysis of RISE over Turns

Following the protocol of Huang et al. (2023b), in this section, we perform an error analysis of the improvement performed by RISE (without any oracle feedback) to understand how the fraction of incorrect and correct responses changes over turns, when **no oracle** is used for early termination. We demonstrate this in the form of Venn diagrams in Figure 5. First note that there is a consistent increase in the portion of problems that stay correct and a consistent decrease in the portion of problems that stay incorrect, which means that the model is able to answer more and more problems as we increase the number of turns. Second, there is a consistent decrease in the number of problems that transition from being correct to incorrect, which is often also not the case for strong proprietary LLMs such as GPT in Huang et al. (2023b). Finally, while we also note that there is a decrease in the total number of incorrect problems that become correct in the subsequent turn, this is a direct consequence of the size of the incorrect response set shrinking as more problems become correct over turns. This indicates that one can induce “intrinsic” self-improvement (per the terminology of Huang et al. (2023b)) via fine-tuning with RISE, even though no external environment input is provided during evaluation.

Qualitative examples. We also inspect several examples from the GSM8K test set to qualitatively understand the behavior of RISE over turns and observe different behavior patterns, that we show in Appendix B. For instance, the trained model may choose to completely rewrite its previous response if it is totally incorrect in order to get to the correct

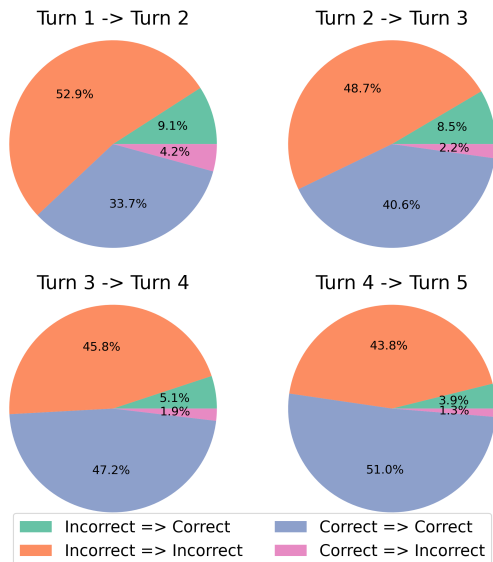


Figure 5: *Change in the fraction of responses that transition their correctness values over the course of multi-turn rollouts from RISE, w/o oracle.* Observe that in general, the fraction of Correct \rightarrow Correct responses increases; Incorrect \rightarrow Incorrect responses decreases; and the fraction of Correct \rightarrow Incorrect responses also decreases, indicating that RISE (w/o any oracle) is able to iteratively improve its responses.

answer or make small edits if the previous response is mostly correct. Another interesting pattern we note is that the model implicitly has the ability to locate errors in previous responses and only refine the erroneous steps. Additionally, the model is tolerant of noisy environmental feedback when there is no oracle-assisted early termination.

7. Discussion, Future Directions, and Limitations

We presented RISE, an approach for fine-tuning LLMs to be able to improve their own responses over multiple turns sequentially. RISE prescribes an iterative RL recipe on top of on-policy rollout data, with expert or self-generated supervision to steer self-improvement. RISE significantly improves self-improvement abilities of 7B models on reasoning tasks (GSM8K and MATH), attaining an improvement over turns that past work (Huang et al., 2023b) has not observed in strong proprietary models. In addition, while RISE outperforms prior approaches that attempt to tackle similar problems of refinements and correction, it is simpler in that it does not require running multiple models at once and can work well with just one model. Despite these good results, there are still many open questions and limitations. Due to computational constraints, we were not able to perform more than two iterations of training with RISE, and no more than one iteration when the supervision comes from the learner itself. Improving with self-generated supervision will likely require compute. RISE requires running manual iterations and hence, a more “online” variant of RISE is likely the solution in the long run.

Impact Statement

Our work has no special societal implications than any other work attempting to build learning methods for fine-tuning LLMs. That said, very capable LLMs can have significant implications on society and human life, and hence must be deployed cautiously, though this is not unique to our work.

References

- Agarwal, R., Vieillard, N., Stanczyk, P., Ramos, S., Geist, M., and Bachem, O. Gkd: Generalized knowledge distillation for auto-regressive sequence models. *arXiv preprint arXiv:2306.13649*, 2023.
- Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*, 2022.
- Chang, J. D., Shan, W., Oertell, O., Brantley, K., Misra, D., Lee, J. D., and Sun, W. Dataset reset policy optimization for rlhf. *arXiv preprint arXiv:2404.08495*, 2024.
- Charalambous, Y., Tihanyi, N., Jain, R., Sun, Y., Ferrag, M. A., and Cordeiro, L. C. A new era in software security: Towards self-healing software via large language models and formal verification. *arXiv preprint arXiv:2305.14752*, 2023.
- Chen, B., Shu, C., Shareghi, E., Collier, N., Narasimhan, K., and Yao, S. Fireact: Toward language agent fine-tuning, 2023a.
- Chen, X., Lin, M., Schärli, N., and Zhou, D. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*, 2023b.
- Chen, Z., Deng, Y., Yuan, H., Ji, K., and Gu, Q. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.
- Chia, Y. K., Chen, G., Tuan, L. A., Poria, S., and Bing, L. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*, 2023.
- Cobbe, K., Hesse, C., Hilton, J., and Schulman, J. Leveraging procedural generation to benchmark reinforcement learning. *arXiv preprint arXiv:1912.01588*, 2019.
- Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.
- Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., and Goodman, N. D. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.
- Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.
- Ghosh, D., Rahme, J., Kumar, A., Zhang, A., Adams, R. P., and Levine, S. Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability. *NeurIPS*, 2021.
- Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Duan, N., and Chen, W. Critic: Large Language Models can Self-Correct with Tool-Interactive Critiquing. *arXiv preprint arXiv:2305.11738*, 2023.
- Havrilla, A., Raparthy, S., Nalmpantis, C., Dwivedi-Yu, J., Zhuravinskyi, M., Hambro, E., and Railneau, R. Glore: When, where, and how to improve llm reasoning via global and local refinements. *arXiv preprint arXiv:2402.10963*, 2024.
- Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- Hosseini, A., Yuan, X., Malkin, N., Courville, A., Sordoni, A., and Agarwal, R. V-star: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.
- Huang, D., Bu, Q., Zhang, J. M., Luck, M., and Cui, H. Agentcoder: Multi-agent-based code generation with iterative testing and optimisation. *arXiv preprint arXiv:2312.13010*, 2023a.
- Huang, J., Chen, X., Mishra, S., Zheng, H. S., Yu, A. W., Song, X., and Zhou, D. Large language models cannot self-correct reasoning yet. *arXiv preprint arXiv:2310.01798*, 2023b.
- Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pp. 9118–9147. PMLR, 2022.
- Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., Casas, D. d. l., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
- Kang, K., Wallace, E., Tomlin, C., Kumar, A., and Levine, S. Unfamiliar finetuning examples control how language models hallucinate, 2024.

- Langley, P. Crafting papers on machine learning. In Langley, P. (ed.), *Proceedings of the 17th International Conference on Machine Learning (ICML 2000)*, pp. 1207–1216, Stanford, CA, 2000. Morgan Kaufmann.
- Lehnert, L., Sukhbaatar, S., Mcvay, P., Rabbat, M., and Tian, Y. Beyond a*: Better planning with transformers via search dynamics bootstrapping. *arXiv preprint arXiv:2402.14083*, 2024.
- Li, C., Wang, W., Hu, J., Wei, Y., Zheng, N., Hu, H., Zhang, Z., and Peng, H. Common 7b language models already possess strong math capabilities. *arXiv preprint arXiv:2403.04706*, 2024.
- Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let’s verify step by step. *arXiv preprint arXiv:2305.20050*, 2023.
- Liu, X., Yu, H., Zhang, H., Xu, Y., Lei, X., Lai, H., Gu, Y., Ding, H., Men, K., Yang, K., et al. Agentbench: Evaluating llms as agents. *arXiv preprint arXiv:2308.03688*, 2023.
- Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*, 2023.
- Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N., Prabhunoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *arXiv preprint arXiv:2303.17651*, 2023.
- Nijkamp, E., Pang, B., Hayashi, H., Tu, L., Wang, H., Zhou, Y., Savarese, S., and Xiong, C. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. *ICLR*, 2023.
- Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.
- Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.
- Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750. ACM, 2007.
- Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Gordon, G., Dunson, D., and Dudík, M. (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL <http://proceedings.mlr.press/v15/ross11a.html>.
- Rosset, C., Cheng, C.-A., Mitra, A., Santacroce, M., Awadallah, A., and Xie, T. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*, 2024.
- Saha, S., Levy, O., Celikyilmaz, A., Bansal, M., Weston, J., and Li, X. Branch-solve-merge improves large language model evaluation and generation. *arXiv preprint arXiv:2310.15123*, 2023.
- Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*, 2023.
- Shinn, N., Labash, B., and Gopinath, A. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366*, 2023.
- Snell, C., Kostrikov, I., Su, Y., Yang, M., and Levine, S. Offline rl for natural language generation with implicit language q learning. *arXiv preprint arXiv:2206.11871*, 2022.
- Sun, L., Peng, C., Zhan, W., and Tomizuka, M. A fast integrated planning and control framework for autonomous driving via imitation learning. In *Dynamic Systems and Control Conference*, volume 51913, pp. V003T37A012. American Society of Mechanical Engineers, 2018.
- Toshniwal, S., Moshkov, I., Narenthiran, S., Gitman, D., Jia, F., and Gitman, I. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*, 2024.
- Uesato, J., Kushman, N., Kumar, R., Song, F., Siegel, N., Wang, L., Creswell, A., Irving, G., and Higgins, I. Solving math word problems with process-and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.
- Wang, B., Min, S., Deng, X., Shen, J., Wu, Y., Zettlemoyer, L., and Sun, H. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*, 2022a.
- Wang, G., Xie, Y., Jiang, Y., Mandlkar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023a.

- Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. *CoRR, abs/2312.08935*, 2023b.
- Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022b.
- Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *NeurIPS*, 2022.
- Welleck, S., Lu, X., West, P., Brahman, F., Shen, T., Khashabi, D., and Choi, Y. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=hH36JeQZDa0>.
- Yang, H., Yue, S., and He, Y. Auto-gpt for online decision making: Benchmarks and additional opinions. *arXiv preprint arXiv:2306.02224*, 2023a.
- Yang, K., Swope, A. M., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. *arXiv preprint arXiv:2306.15626*, 2023b.
- Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.
- Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023.
- Yuan, L., Cui, G., Wang, H., Ding, N., Wang, X., Deng, J., Shan, B., Chen, H., Xie, R., Lin, Y., et al. Advancing llm reasoning generalists with preference trees. *arXiv preprint arXiv:2404.02078*, 2024a.
- Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *arXiv preprint arXiv:2401.10020*, 2024b.
- Yue, X., Qu, X., Zhang, G., Fu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.
- Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.
- Zeng, A., Liu, M., Lu, R., Wang, B., Liu, X., Dong, Y., and Tang, J. Agenttuning: Enabling generalized agent abilities for llms. *arXiv preprint arXiv:2310.12823*, 2023.
- Zhang, T., Wang, X., Zhou, D., Schuurmans, D., and Gonzalez, J. E. Tempera: Test-time prompting via reinforcement learning. *arXiv preprint arXiv:2211.11890*, 2022.
- Zhang, T., Madaan, A., Gao, L., Zheng, S., Mishra, S., Yang, Y., Tandon, N., and Alon, U. In-context principle learning from mistakes. *arXiv preprint arXiv:2402.05403*, 2024.
- Zhou, A., Yan, K., Shlapentokh-Rothman, M., Wang, H., and Wang, Y.-X. Language agent tree search unifies reasoning acting and planning in language models. *arXiv preprint arXiv:2310.04406*, 2023.
- Zhou, Y., Zanette, A., Pan, J., Levine, S., and Kumar, A. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

A. Additional Results

A.1. Complete Comparisons and Discussion: Extended Version of Table 1

We provide an extended version of Table 1, with a clear explanation of how we implement baselines and a discussion of comparisons.

Table 3: **Comparing RISE with other approaches (Self-Refine, GLoRE, and V-STaR) and other baseline approaches.** Observe that RISE attains the biggest performance improvements between 1-turn and 5-turn performance without the use of an oracle on both GSM8K and MATH. This performance gap is even larger when oracle early termination is allowed (5-turn w/ oracle). Self-Refine largely degrades performance across the board. GLoRE trains a separate refinement model, but still performs worse than RISE.

Approach	GSM8K (Cobbe et al., 2019)				MATH (Hendrycks et al., 2021)			
	w/o oracle		w/ oracle		w/o oracle		w/ oracle	
	m1@t1	→ m5@t1	→ m1@t5	→ p1@t5	m1@t1	→ m5@t1	→ m1@t5	→ p1@t5
RISE (Ours)								
Llama2 Base	10.5	22.8 (+12.3)	11.1 (+0.6)	13.9 (+3.4)	1.9	5.1 (+3.2)	1.4 (-0.5)	2.3 (+0.4)
+Boost	32.9	45.4 (+12.5)	39.2 (+6.3)	55.5 (+22.6)	5.5	6.8 (+1.3)	5.5 (+0.0)	14.6 (+9.1)
+Iteration 1	35.6	49.7 (+14.1)	50.7 (+15.1)	63.9 (+28.3)	6.3	8.8 (+2.5)	9.7 (+3.4)	19.4 (+13.1)
+Iteration 2	37.3	51.0 (+13.7)	55.0 (+17.7)	68.4 (+31.1)	5.8	10.4 (+4.6)	10.4 (+4.6)	19.8 (+14.0)
RISE (Ours)								
Mistral-7B	33.7	49.4 (+15.7)	39.0 (+5.3)	46.9 (+13.2)	7.5	13.0 (+5.5)	8.4 (+0.9)	13.0 (+5.5)
+ Iteration 1	35.3	50.6 (+15.3)	59.2 (+23.9)	68.6 (+33.3)	6.7	9.5 (+2.8)	18.4 (+11.1)	29.7 (+22.4)
Baselines								
GPT-3.5	66.4	80.6 (+14.2)	71.0 (+4.6)	74.7 (+8.3)	39.7	47.8 (+8.1)	45.1 (+5.4)	54.3 (+14.6)
Mistral-7B	33.7	49.4 (+15.7)	39.0 (+5.3)	46.9 (+13.2)	7.5	13.0 (+5.5)	8.4 (+0.9)	13.0 (+5.5)
Eurus-7b-SFT	36.3	66.3 (+30.0)	47.9 (+11.6)	53.1 (+16.8)	12.3	19.8 (+7.5)	16.3 (+4.0)	22.9 (+10.6)
Self-Refine			→ m1@t3	→ p1@t3			→ m1@t3	→ p1@t3
Base	10.5	22.4 (+11.9)	7.1 (-3.4)	13.0 (+2.5)	1.9	5.1 (+3.2)	1.9 (0.0)	3.1 (+1.2)
+Boost	32.9	45.3 (+12.4)	26.5 (-6.4)	40.9 (+8.0)	5.5	6.5 (+1.0)	2.9 (-2.6)	7.2 (+1.7)
+Iteration1	35.6	49.5 (+13.9)	31.7 (-3.9)	43.7 (+8.1)	6.3	8.7 (+2.4)	5.9 (-0.4)	9.9 (+3.6)
+Iteration2	37.3	50.5 (+13.2)	33.3 (-4.0)	44.5 (+7.2)	5.8	9.4 (+3.6)	5.7 (-0.1)	9.5 (+3.7)
GPT-3.5	66.4	80.2 (+13.8)	61.0 (-5.4)	71.6 (+5.2)	39.7	46.5 (+6.8)	36.5 (-3.2)	46.7 (+7.0)
Mistral-7B	33.7	48.5 (+14.8)	21.2 (-12.5)	37.9 (+4.2)	7.5	12.3 (+4.8)	7.1 (-0.4)	11.4 (+3.9)
Eurus-7b-SFT	36.3	65.9 (+29.6)	26.2 (-10.1)	42.8 (+6.5)	12.3	19.4 (+7.1)	9.0 (-3.3)	15.1 (+2.8)
GLoRE			→ m1@t3	→ p1@t3				
+ORM	48.2		49.5 (+1.3)	57.1 (+8.9)				
+SORM	48.2		51.6 (+3.4)	59.7 (+11.5)			N/A	
+Direct	48.2		47.4 (-0.8)	59.2 (+11.0)				
V-STaR			→ m64@t1					
+STaR	28.0		46.1 (+18.1)					
+Verification	28.0		56.2 (+28.2)				N/A	
+V-STaR	28.0		63.2 (+35.2)					

Comparison with Self-Refine (Madaan et al., 2023). To build a self-refine baseline (Madaan et al., 2023) evaluation, we slightly modified our evaluation pipeline following the self-refine approach. In this setup (Figure 6), the model generates an initial response, and then the environment prompts the model to locate errors in the generated solution and refine its answer based on the initial response and the identified error.

However, our experiments show that without any oracle hint from the environment or human feedback, the self-refine approach leads to a degradation in performance across all models. Only when oracle feedback is available to assist with early termination does the self-refine approach provide a slight performance boost. This highlights the limitation of the self-refine structure in effectively improving model performance without external guidance, which is also observed in (Huang et al., 2022).

In contrast, the model trained with RISE can attain consistent performance improvements without relying on an oracle. By training the model to iteratively refine its responses, our method enables the model to self-correct and improve its performance over multiple turns. This showcases the effectiveness of our approach in comparison to the self-refine baseline, as it allows for more robust and consistent performance gains without the need for the oracle assistance.

Comparison with GLoRE (Havrilla et al., 2024). GLoRE is a multi-model system that relies on a student model to propose drafts, an Outcome-based Reward Model (ORM) or Step-wise ORM to locate errors at different granularity levels, and a Global or Local Refinement Model for adjusting these errors. Since no code was openly available for this approach, in our experiments, we compared to the numbers from the main paper (Havrilla et al., 2024). While the comparison against GLoRE is already apples-to-oranges since our method only trains a single end-to-end model, while GLoRE trains multiple models. Performance-wise, GLoRE’s global and local refinement models show little to no improvement in overall accuracy

Self-Refine

System: You are an AI language model designed to assist with math problem-solving. In this task, I will provide you with math problems. Your goal is to solve the problem step-by-step, showing your reasoning at each step. After you have finished solving the problem, present your final answer as $\boxed{\text{Your Answer}}$.

<One-shot Example 14>

User: <Query>

Agent: <Initial Answer>

User: There is an error in the solution above because of lack of understanding of the question. What is the error? To find the error, go through each step of the solution, and check if everything looks good.

Agent: <Critic>

User: Now, rewrite the solution in the required format:

Agent: <Refined Answer>

Figure 6: **Prompt for Self-Refine:** We follow the standard pipeline of the original paper, prompt the LLM to refine and correct its previous mistakes.

without an oracle, and even exhibit decreasing accuracy in some cases. However, when an oracle is used to guide the refinement process, GLoRE demonstrates a 10% improvement on the 7B model in the GSM8K dataset.

As anticipated, since we run RISE from a less advanced base model (Llama2 7B), we observe a slightly lower absolute performance compared to GLoRE. However, RISE demonstrates its effectiveness in self-improvement by sequentially enhancing its performance by an impressive 13.4% within just 3 turns without an oracle feedback, and by a remarkable 23.4% with an oracle on GSM8K. This showcase of RISE’s capabilities is particularly noteworthy considering that GLoRE utilizes 3 independent models - one for generating candidate solutions, one reward model for locating errors, and one refinement model for refinement.

Comparison with V-STaR (Hosseini et al., 2024). V-STaR requires training an additional verifier model to rank candidate answers generated by the targeted model, but it does not make any sequential revisions or improvements to a response. While comparing RISE to using a verifier for re-ranking the top 5 responses at the first turn (as a base comparison) would have been informative, we were unable to find this specific result in the original V-STaR paper. The results presented in the official table 3 for V-STaR correspond to running 64 samples, which improves the base model’s performance by 35.2% for each prompt during evaluation. In contrast, our method, RISE, after the same amount of finetuning iterations (3 iterations) and using only 5 samples, improves upon the base model by 44.5% (calculated as $55.0\% - 10.5\% = 44.5\%$). This comparison highlights RISE’s efficiency in achieving significant improvements with fewer samples and iterations compared to V-STaR’s approach of using a large number of samples without sequential refinement.

Moreover, V-STaR’s performance is inherently bounded by the candidate generator’s performance. As discussed in Section 5, if there is no correct response among the generated candidates, the problem remains unsolved. In contrast, we show in Figure 2 that RISE can also solve problems that were not solved by majority voting with a much higher budget in the first turn. Furthermore, we believe that combining V-STaR with RISE could lead to even better performance, as RISE can generate better models and a verifier can be complementarily used for filtering.

Comparison with other base models. Mistral-7B (Jiang et al., 2023) and Eurus-7B-SFT (Yuan et al., 2024a) are models that exhibit comparable performance to our method in terms of the absolute maj@5 performance. However, it is crucial to note that these base models are fine-tuned using a vast amount of data, including data specifically tuned for math reasoning performance (Yuan et al., 2024a), while our model is fine-tuned on a single domain. That said, we do show that fine-tuning with RISE can still enhance the performance of Mistral-7B models.

To summarize, our method offers several advantages over GLoRE and V-STaR, such as end-to-end error correction with a single model, superior performance with fewer samples, and the ability to solve problems that cannot be solved by random sampling in the first turn. Although our maj@1 performance is lower than GLoRE’s base model EI, which is an apples-to-oranges comparison our ultimate 5-turn performance surpasses their best absolute performance in both oracle and non-oracle scenarios. Compared to other base models like Mistral-7B and Eurus-7B-SFT, our method achieves comparable performance while being fine-tuned on a single domain, and it can be generalized to further enhance the performance of better base models.

A.2. Does the Self-Improvement Strategy Learned by RISE Generalize to Novel Problems?

An important aspect we study is whether the learned LLM can generalize to novel problems. In Figure 7, we demonstrate the generalizability of the self-improvement procedure learned by RISE. Specifically, we compare the performance of the RISE model trained on MATH when evaluated to the test subset of the GSM8K dataset (Figure 7 Left); RISE model trained on GSM8K when evaluated on the test subset of the MATH dataset (Figure 7 Center); and the RISE model trained on both MATH and GSM8K datasets when evaluated on the SVAMP dataset (Figure 7 Right). We observe that the model trained on one dataset is still able to improve the base model’s performance on another dataset. Notably, more iterations of RISE training (i.e., from Boost → Iteration 1 → Iteration 2) not only improve performance at the first turn but also produce more effective improvements (compare “Iteration 2” against the “Boost” model from the training data), including a higher rate of improvement from turn to turn) as the number of turns grows on the x-axis. This means that even though these models have not seen queries similar to the evaluation dataset, simply training with RISE on *some* kind of mathematical prompts still boosts the efficacy of the self-improvement strategy on a new distribution of test prompts. This finding suggests that RISE is capable of instilling self-improvement procedures that can generalize effectively beyond the distribution of prompts in the fine-tuning data.

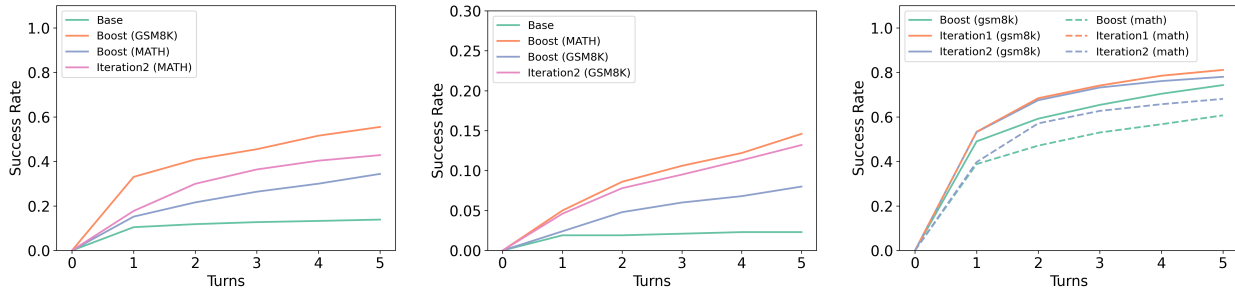


Figure 7: **Performance of models fine-tuned in GSM8K / MATH tasks on GSM8K, MATH, and SVAMP datasets.** We evaluate model fine-tuned on MATH on the GSM8K test set (Left); model fine-tuned GSM8K on MATH (Center); and the model fine-tuned on a mixture of GSM8K and MATH on the SVAMP (Right) data. Observe that with more iterations of RISE training, as the number of turns increases, the models are able to produce more effective improvements.

A.3. Iterative Training with RISE Fits the Training Distribution Flexibly

In this section, we elaborate a hypothesis posed in in Section 5 that RISE perhaps more effective because it is able to represent more flexible distributions, as it models conditionals $\pi(\mathbf{y}_i|\mathbf{x}, \mathbf{y}_{0:i-1})$ instead of attempting to model the target distribution $p(\mathbf{y}^*|\mathbf{x})$ directly. To verify if this hypothesis is true, we track the training un-weighted, negative log likelihood loss over the course of training on multi-turn data and compare it against the negative log likelihood loss attained by directly attempting to predict the final response in Figure 8. We find that for any given number of epochs (including fractional number of epochs on the x-axis), the training loss is lower when training on multi-turn data that RISE generates in comparison with oracle responses to the prompts obtained from an expert. This suggests that indeed the learning problem posed by RISE is easier.

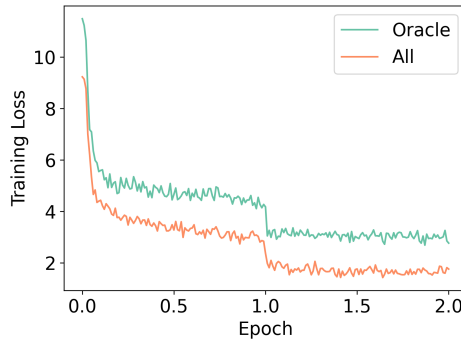


Figure 8: **The training perplexity (loss) of fitting only the oracle answer or a sequence of answers.** Observe that fitting a sequence of answers (orange) reduces the loss more than fitting only the oracle answer (green).

B. Qualitative Examples from RISE

In this section, we provide some qualitative examples showing the nature of edits made by RISE on various datasets that we train on. We utilize the model from iteration 2 of RISE.

RISE Example (Small Edits)

Query: Mike plays ping pong for 40 minutes. In the first 20 minutes, he scores 4 points. In the second 20 minutes, he scores 25% more points. How many total points did he score?

Turn-1 Response:

Step 1: In the first 20 minutes, Mike scores 4 points.

Step 2: In the second 20 minutes, Mike scores 25% more points than he did in the first 20 minutes. **So, Mike scores 4 + 4 = 8 points in the second 20 minutes.**

Step 3: To find the total points, we need to add the points scored in the first and second 20 minutes. $\text{total_points} = 4 + 8 = 12$.

Answer: ✗

Turn-2 Response:

I apologize for the mistake. Let's solve the problem step by step.

Step 1: In the first 20 minutes, Mike scores 4 points.

Step 2: In the second 20 minutes, Mike scores 25% more points than he did in the first 20 minutes. **This means he scores $4 + 4 * 25\% = 4 + 1 = 5$ points in the second 20 minutes.**

Step 3: To find the total points, we add the points scored in the first and second 20 minutes: $4 + 5 = 9$ points.

Answer: ✓

RISE Example (Big Edits)

Query: In a candy machine, there are 22 more than four times the number of pink gumballs as there are blue gumballs. If there are 12 blue gumballs how many pink ones are there?

Turn-1 Response:

Step 1: Let's assume the number of blue gumballs as x. Step 2: According to the given information, there are 22 more than four times the number of pink gumballs as there are blue gumballs. So, the number of pink gumballs can be expressed as $4x + 22$. **Step 3: We are given that there are 12 blue gumballs, so let's substitute x with 12 in the equation.**

Answer: ✗

Turn-2 Response:

Let's solve the problem step by step: Step 1: Let's denote the number of blue gumballs as x. Step 2: We are given that the number of pink gumballs is 22 more than four times the number of blue gumballs. So, the number of pink gumballs can be expressed as $4x + 22$. **Step 3: We are also given that there are 12 blue gumballs, so we can substitute $x = 12$ into the expression for the number of pink gumballs: $4 * 12 + 22 = 48 + 22 = 70$ Therefore, there are **boxed70 pink gumballs.****

Answer: ✓

Figure 9: Examples of RISE correct its previous behavior in different modes. Some only make changes to a small part (small edits), some may directly rewrite most of its previous answers (big edits) because the first step in the previous answer is wrong. The mistaken steps of different turns are highlighted in **red**, and the correct are highlighted in **green**. This demonstrates shows how RISE can correct its previous answers and finally get to a correct answer.

RISE Example (Noise Tolerance)

Query: Ryan plants 2 flowers a day in his garden. After 15 days, how many flowers does he have if 5 did not grow?

Turn-1 Response:

Step 1: Ryan plants 2 flowers a day for 15 days, so he plants a total of $2 \text{ flowers/day} * 15 \text{ days} = 30 \text{ flowers}$ in his garden. Step 2: However, 5 flowers did not grow, so we need to subtract these non-growing flowers from the total count. Step 3: Calculate the number of flowers that actually grew in Ryan's garden. $\text{flowers_grew} = 30 \text{ flowers} - 5 \text{ flowers} = 25 \text{ flowers}$.

Answer: ✓

Turn-2 Response:

Apologies for the mistake. Let's correct the solution. Step 1: In 15 days, Ryan plants 2 flowers a day, so he plants a total of $2 \text{ flowers/day} * 15 \text{ days} = 30 \text{ flowers}$ in his garden. Step 2: Since 5 flowers did not grow, we need to subtract these non-growing flowers from the total count. Step 3: Calculate the number of flowers that actually grew in Ryan's garden. $\text{flowers_grew} = 30 \text{ flowers} - 5 \text{ flowers} = 25 \text{ flowers}$.

Answer: ✓

Turn-3 Response:

Let's reevaluate the solution. Step 1: Ryan plants 2 flowers a day for 15 days, so he plants a total of $2 \text{ flowers/day} * 15 \text{ days} = 30 \text{ flowers}$ in his garden. Step 2: However, 5 flowers did not grow, so we need to add these flowers to the total count, as they were planted but did not grow. Step 3: Calculate the total number of flowers in Ryan's garden. $\text{total_flowers} = 30 \text{ flowers} + 5 \text{ flowers} = 35 \text{ flowers}$.

Answer: ✗

Turn-4 Response

Let's correct the solution. Step 1: In 15 days, Ryan plants 2 flowers a day, so he plants a total of $2 \text{ flowers/day} * 15 \text{ days} = 30 \text{ flowers}$ in his garden. Step 2: Since 5 flowers did not grow, we need to subtract these non-growing flowers from the total count. Step 3: Calculate the total number of flowers in Ryan's garden after subtracting the non-growing flowers. $\text{total_flowers} = \text{total_planted_flowers} - \text{non_growing_flowers} = 30 \text{ flowers} - 5 \text{ flowers} = 25 \text{ flowers}$.

Answer: ✓

Turn-5 Response

Let's reevaluate the solution. Step 1: In 15 days, Ryan plants 2 flowers a day, which is $2 \text{ flowers/day} * 15 \text{ days} = 30 \text{ flowers}$ in total. Step 2: However, 5 flowers did not grow, so we have to subtract these 5 flowers from the total. Step 3: Therefore, the total number of flowers Ryan has is $30 \text{ flowers (total planted)} - 5 \text{ flowers (did not grow)} = 25 \text{ flowers}$.

Answer: ✓

Figure 10: Examples of RISE's **tolerance of noisy feedback**. In the evaluation where there is no oracle, we use a verifier that always tells the model, "The answer is incorrect. Please try again. Here's the question: <Query>". We observe that from turn 2 to turn 5, the model is still able to output the correct answer in 3 out of 4 turns. In the response at turn 4, the model performs half of the solution correctly, and only the last step is wrong. In addition, in the turns where the model still outputs the correct answer, it doesn't directly copy the answer from the previous turn but instead comes up with a new paraphrase. This demonstrates that RISE provides the model with tolerance to noisy feedback and does not randomly change its behavior upon receiving a revisiting request. The mistaken steps of different turns are highlighted in **red**.

RISE Example (Error Localization)

Query: Josh decides to try flipping a house. He buys a house for \$80,000 and then puts in \$50,000 in repairs. This increased the value of the house by 150%. How much profit did he make?

Turn-1 Response:

Step 1: Josh buys a house for \$80,000. Step 2: He puts in \$50,000 in repairs, which increases the value of the house by 150%. Step 3: To calculate the increase in value, we multiply the cost of the house by 150% and add it to the cost of the house. $\text{increase_in_value} = \$80,000 * 150\% = \$120,000$. Step 4: To calculate the profit, we subtract the cost of the house from its new value. $\text{profit} = \text{increase_in_value} - \$80,000 = \$120,000 - \$80,000 = \$40,000$.

Answer: ✗

Turn-2 Response:

Let's solve the problem step-by-step: Step 1: Josh buys a house for \$80,000. Step 2: He puts in \$50,000 in repairs, and this increases the value of the house by 150%. Step 3: The percentage increase in value is 150% of \$80,000, which is \$120,000 ($80,000 * 1.5 = \$120,000$). Step 4: To find the actual amount that would be added to the house's value after repairs, we need to subtract the amount he spent on repairs from the increased value, which is $\$120,000 - \$50,000 = \$70,000$. Step 5: The profit Josh would make is the difference between the cost of buying the house and the increased value after repairs, which is $\$70,000 - \$80,000 = -\$10,000$ (\$10,000 less than the initial cost).

Answer: ✗

Turn-3 Response: Let's solve the problem step-by-step: Step 1: Josh buys a house for \$80,000. Step 2: He puts in \$50,000 in repairs. Step 3: This increases the value of the house by 150%. Step 4: To find the increase in value, we calculate 150% of the original price of the house: $150\% \text{ of } \$80,000 = \$120,000$. Step 5: The increase in value after repairs is the original price plus 150% of the original price: $\$80,000 + \$120,000 = \$200,000$. Step 6: Josh sells the house for \$200,000 and made a profit of $\$200,000 - \$80,000 = \$120,000$.

Answer: ✗

Turn-4 Response: Let's solve the problem step-by-step: Step 1: Josh buys a house for \$80,000. Step 2: He puts in \$50,000 in repairs, increasing the value of the house by 150%. Step 3: The increase in value is 150% of \$80,000, which is \$120,000 ($80,000 * 1.5 = \$120,000$). Step 4: The total amount spent, including the purchase of the house and the repairs, is $\$80,000 + \$50,000 = \$130,000$. Step 5: The profit Josh made is the increase in value after repairs minus the total amount spent, which is $\$120,000 - \$130,000 = -\$10,000$. Therefore, Josh made a boxed0 profit.

Answer: ✗

Turn-5 Response: Let's solve the problem step-by-step: Step 1: Josh buys a house for \$80,000. Step 2: He puts in \$50,000 in repairs. Step 3: The repairs increase the value of the house by 150%. Step 4: The increase in value is \$50,000, which is 150% of \$33,333 ($50,000 / 1.5 = \$33,333$). Step 5: The repairs cost \$50,000, and the increase in value is \$33,333. Step 6: Therefore, Josh's profit is $\$33,333 - \$50,000 = -\$16,667$. Therefore, Josh made a 16676 profit.

Answer: ✗

Figure 11: Examples of RISE not being able to correct its error within 5 steps but demonstrating meaningful **error localization behavior**. Even in turn 2, it has already solved the problem at step 4 but mistakenly takes another step and reaches the wrong answer. The following turns are unable to correct this small error. Though this problem remains unsolved, we observe that (1) the model is able to stick to the correct steps, where all responses reach an intermediate step of 12000 correctly, except for the last response, where the model tries to modify the answer from one step ahead; (2) the model doesn't repeat its responses, which is a behavior we notice when evaluating some off-the-shelf models; and (3) the model is making meaningful changes to the incorrect steps. In summary, although the final answer is still incorrect, we observe that through RISE, the model is able to locate the error and perform local computation correctly. The mistaken steps of different turns are highlighted in red, and the correct steps in turn 2 is highlighted in green.

C. Experimental Details

C.1. Hyperparameters for Fine-Tuning with RISE

For finetuning, we utilize the [FastChat](#) codebase, but we customize the loss function to be weighted by reward. The base models are directly loaded from Hugging Face: <https://huggingface.co/meta-llama/Llama-2-7b-hf> and [Llama-2-7b-chat-hf](#) and [Mistral-7B-Instruct-v0.2](#). The hyperparameters used for finetuning are specified in Table 4.

Table 4: Hyperparameters used for RISE

Hyperparameter	Values
bf16	True
epochs	2
per device train batch size	1
gpus	4xA40
gradient accumulation steps	16
learning rate	1e-5
weighted decay	0
warmup ratio	0.04
learning rate scheduler type	cosine
tf32	True
model max length	2048

C.2. Inference Hyperparameters

For API-based models, such as GPT-3.5, we directly query the official web API provided by OpenAI. In the case of open-source models, we utilize [FastChat](#) to serve the model as a web API and interact with the environment through API calls. Serving a 7B model requires a single A100 or A40 GPU. To control the randomness and length of answers generated by the LLMs, we employ the hyperparameters specified in Table 5.

Table 5: The hyperparameter settings used for generating responses from open-source and the GPT models.

Hyperparameters/Description	Open-source	GPT
temperature	1.0	0.7
top_p	1.0	1
max_new_tokens	1000	512

C.3. Datasets

The GSM8K dataset consists of 7,473 problems in the training portion and 1,319 problems in the testing portion. Similarly, the MATH dataset is divided into 7,500 problems for training and 1,000 problems for testing. The training portions of both datasets are used to generate trajectories in each iteration of the RISE method, while the testing portions are held out for evaluating the performance of the models. Additionally, the SVAMP dataset, containing 1,000 problems, is used solely for evaluation purposes to demonstrate the generalizability of our method.

For generating training trajectories, at iteration i , the current model is rolled out for a maximum of i steps. Subsequently, an expert model, such as GPT-3.5, is employed to take the conversation history between the current model and the environment and sample 5 responses. Among these responses, the one with the highest reward is chosen, with arbitrary tie-breaking in case of multiple responses with the same highest reward.

D. Prompts used for seeding RISE on the GSM8K and MATH tasks

Here we show the prompt we used for data generation and model evaluation:

RISE

System: You are an AI language model designed to assist with math problem-solving. In this task, I will provide you with math problems. Your goal is to solve the problem step-by-step, showing your reasoning at each step. After you have finished solving the problem, present your final answer as $\boxed{\text{Your Answer}}$.

<One-shot Example 13>

User: <Query>

Agent: <Answer>

User: The answer is incorrect. Please try again. Here's the question: <Query>

Agent: <Answer>

...

Figure 12: **Prompt for RISE:** We follow the standard pipeline of math problem solving, prompt the LLM to solve the problem and retry until it exceeds the maximum number of turns

D.1. One-shot Example

RISE One-shot example

User: Of the 60 students in the drama club, 36 take mathematics, 27 take physics and 20 students take both mathematics and physics. How many drama club students take neither mathematics nor physics?

Agent: Step 1: To find the number of drama club students who take neither mathematics nor physics, we first need to determine the number of students taking only math, only physics, and both math and physics.

Step 2: Out of the 60 students in the drama club, 36 take mathematics and 27 take physics, and 20 students take both mathematics and physics.

Step 3: To calculate the number of students taking only math, we subtract the number of students taking both from the total taking math: $36 - 20 = 16$.

Step 4: Similarly, to calculate the number of students taking only physics, we subtract the number of students taking both from the total taking physics: $27 - 20 = 7$.

Step 5: Now, to find the number of students taking neither math nor physics, we subtract the total number of students taking only math, only physics, and both from the total number of students in the drama club. This gives us $60 - 16 - 7 - 20 = \boxed{17}$ students taking neither.

Answer: $\boxed{17}$

Figure 13: **One-shot Example:** Demonstrates how to solve a math problem step-by-step and the expected answer format.

Self-Refine One-shot example

User: Of the 60 students in the drama club, 36 take mathematics, 27 take physics and 20 students take both mathematics and physics. How many drama club students take neither mathematics nor physics?

Agent: Step 1: To find the number of drama club students who take neither mathematics nor physics, we first need to determine the number of students taking only math, only physics, and both math and physics.

Step 2: Out of the 60 students in the drama club, 36 take mathematics and 27 take physics, and 20 students take both mathematics and physics.

Step 3: To calculate the number of students taking only math, we subtract the number of students taking both from the total taking math: $36 - 20 = 16$.

Step 4: Similarly, to calculate the number of students taking only physics, we subtract the number of students taking both from the total taking physics: $27 - 20 = 7$.

Step 5: Now, to find the number of students taking neither math nor physics, we add the number of students taking only math and only physics, then subtract this sum from the total number of students in the drama club. This gives us $60 - (16 + 7) = \boxed{37}$ students taking neither.

Answer: $\boxed{37}$

User: There is an error in the solution above because of lack of understanding of the question. What is the error? To find the error, go through each step of the solution, and check if everything looks good.

Agent: Step 1: Correct. We need to find the number of students taking neither math nor physics.

Step 2: Correct. The given information is accurately stated.

Step 3: Correct. The calculation of students taking only math is correct.

Step 4: Correct. The calculation of students taking only physics is correct.

Step 5: Incorrect. The solution adds the number of students taking only math and only physics, but this does not account for the students taking both subjects. By subtracting this sum from the total number of students, the solution double-counts the students who take both math and physics.

User: Now, rewrite the solution in the required format:

Agent: <Same as 13>

Figure 14: **One-shot Example:** Demonstrates the process of self-critique and error correction based on the self-critique.