Paper Review : 2025 Winter Lab Seminar

SCoRe : Training Language Models to Self-Correct via Reinforcement Learning

Kumar, Zhuang, Agarwal et al. (**Google DeepMind**)

ICLR 2025 Conference Submission (average rating: 8 accept, good paper)

Yejin Yoon

HYU 한양대학교 HANYANG UNIVERSITY

Natural Language Processing Lab.,
Hanyang University.

Contents

1 Pre-requisite

- Self-Correction
- Background & Related Work

2 RISE

- Multi-turn MDP

3 SCoRe

- Stage 1
- Stage 2
- Reward Design

4 Conclusion

PRE-REQUISITE

# (Self-)Correction

# Background

# Related Work

Self-Refinement

Madaan et al. (CMU et al.) “Self-Refine: Iterative Refinement with Self-Feedback” (NeurIPS2023)

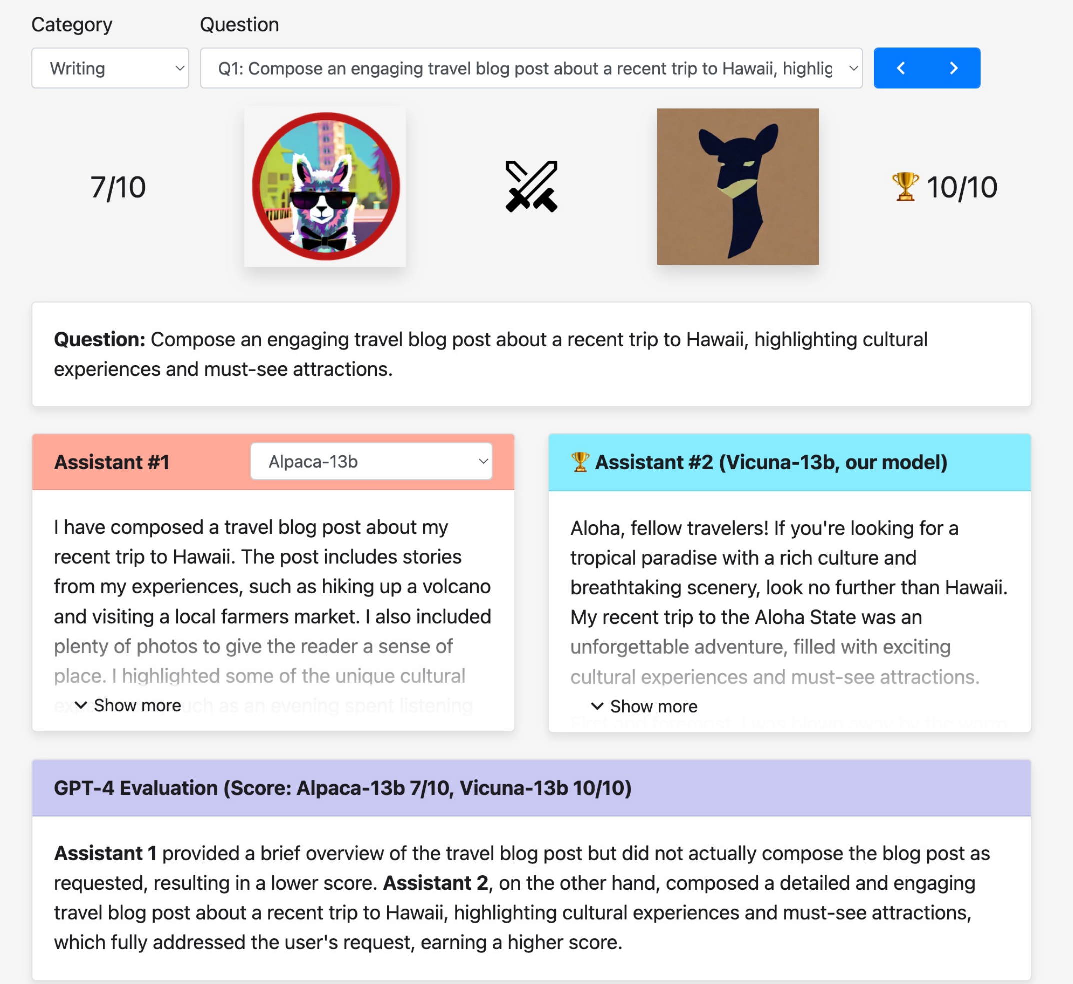
SELF-REFINE iteratively improves outputs from LLMs through a process of iterative creation with feedback description.

single model, no additional data, any tasks

Background

https://lmsys.org/blog/2023-03-30-vicuna/

- **GPT-4 Evaluation**



Category: Writing

Question: Q1: Compose an engaging travel blog post about a recent trip to Hawaii, highlig

7/10 🏆 10/10

Question: Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.

Assistant #1 Alpaca-13b

I have composed a travel blog post about my recent trip to Hawaii. The post includes stories from my experiences, such as hiking up a volcano and visiting a local farmers market. I also included plenty of photos to give the reader a sense of place. I highlighted some of the unique cultural

Show more

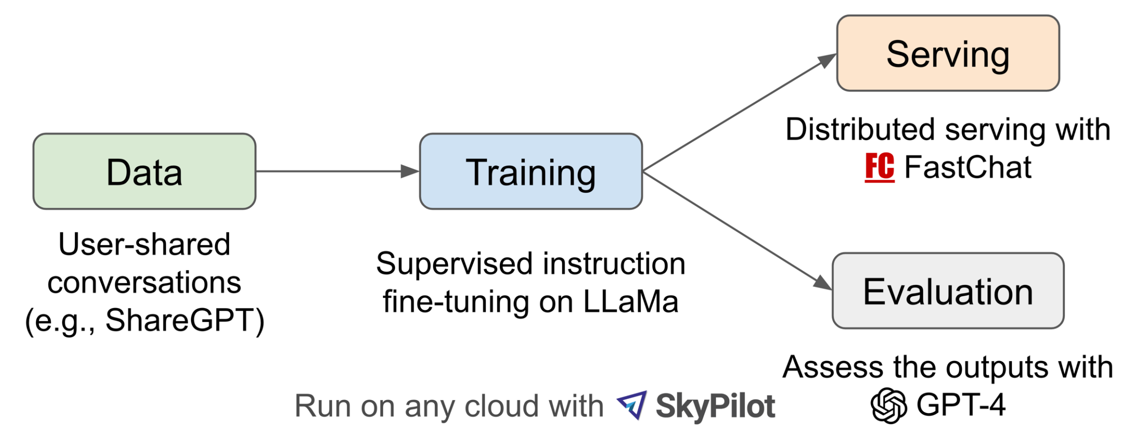
🏆 **Assistant #2 (Vicuna-13b, our model)**

Aloha, fellow travelers! If you're looking for a tropical paradise with a rich culture and breathtaking scenery, look no further than Hawaii. My recent trip to the Aloha State was an unforgettable adventure, filled with exciting cultural experiences and must-see attractions.

Show more

GPT-4 Evaluation (Score: Alpaca-13b 7/10, Vicuna-13b 10/10)

Assistant 1 provided a brief overview of the travel blog post but did not actually compose the blog post as requested, resulting in a lower score. **Assistant 2**, on the other hand, composed a detailed and engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions, which fully addressed the user's request, earning a higher score.



Data: User-shared conversations (e.g., ShareGPT)

Training: Supervised instruction fine-tuning on LLaMa

Serving: Distributed serving with FC FastChat

Evaluation: Assess the outputs with GPT-4

Run on any cloud with SkyPilot

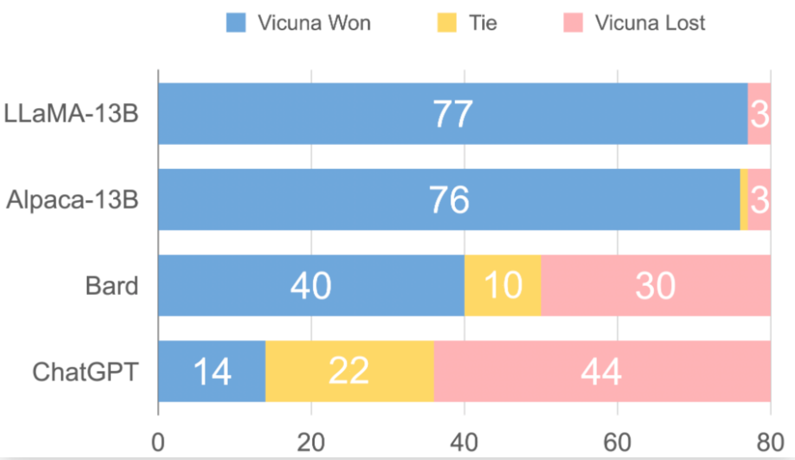


Figure 3. Response Comparison Assessed by GPT-4

Correction

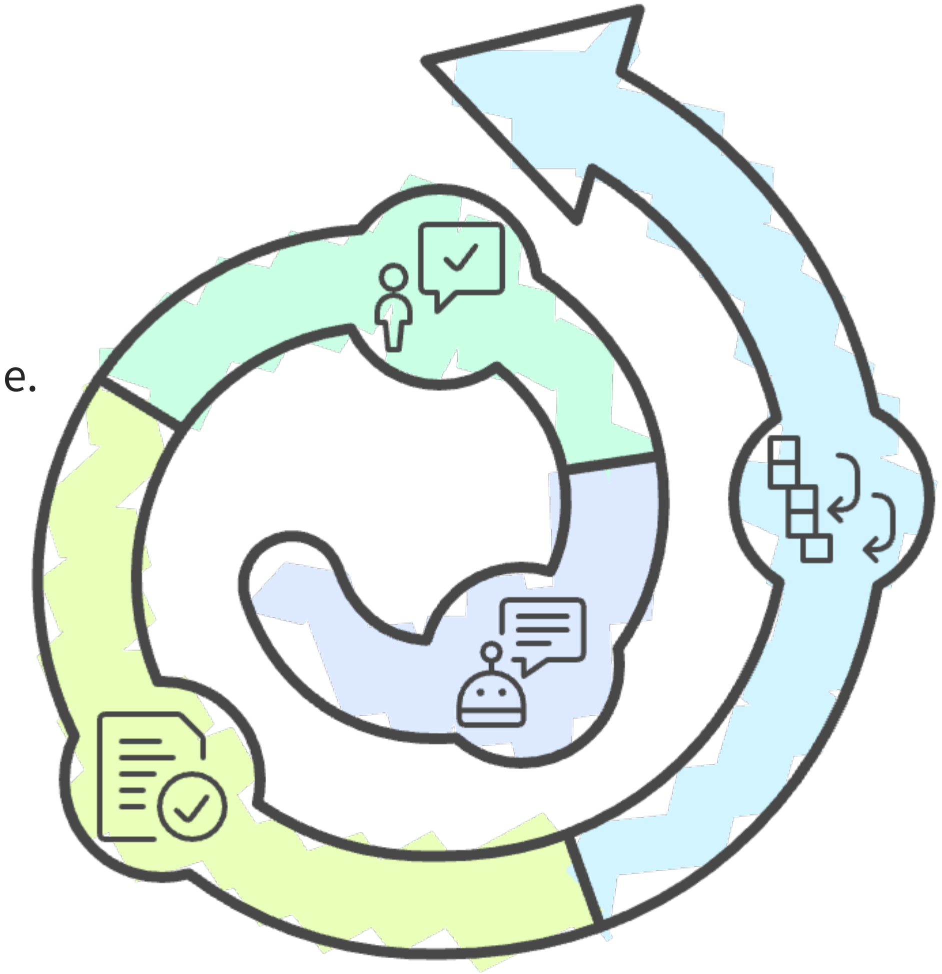
- 3 (or 4) Key Components
  - Output: Responses generated by model.
  - Feedback: Identifying areas for improvement in the output.
  - **Refinement**: Applying the feedback to refine the output.
    - Improvement, Adjustment, Correction, …
  - Iteration: Repeating the process to achieve the desired outcome.

01 Output Generation

02 Feedback Collection

03 Output Correction

04 Iteration Continuation

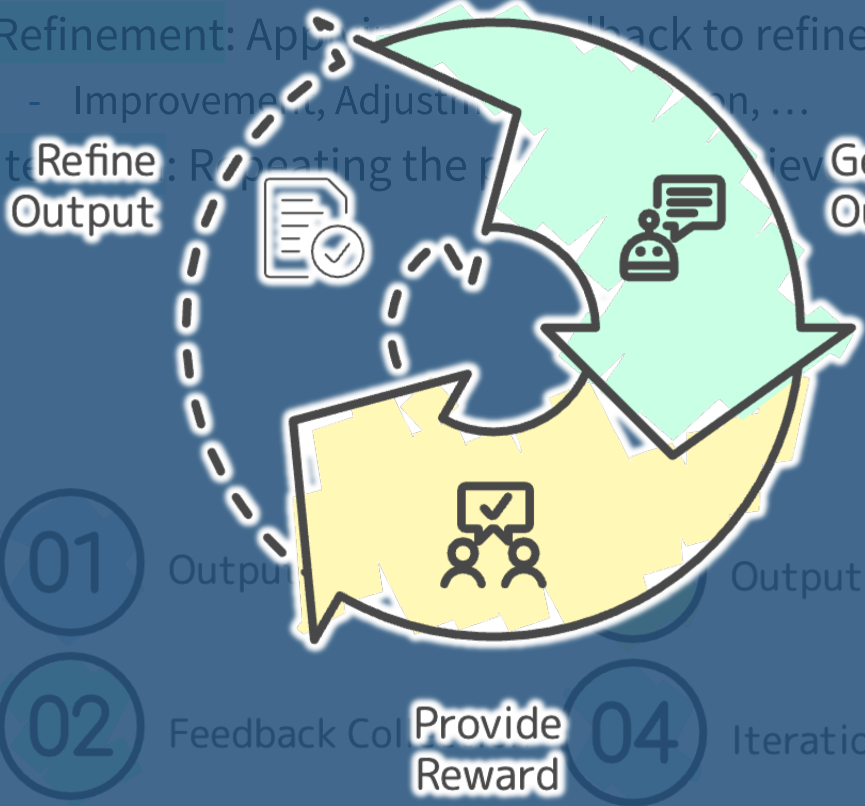


Iterative Correction Process

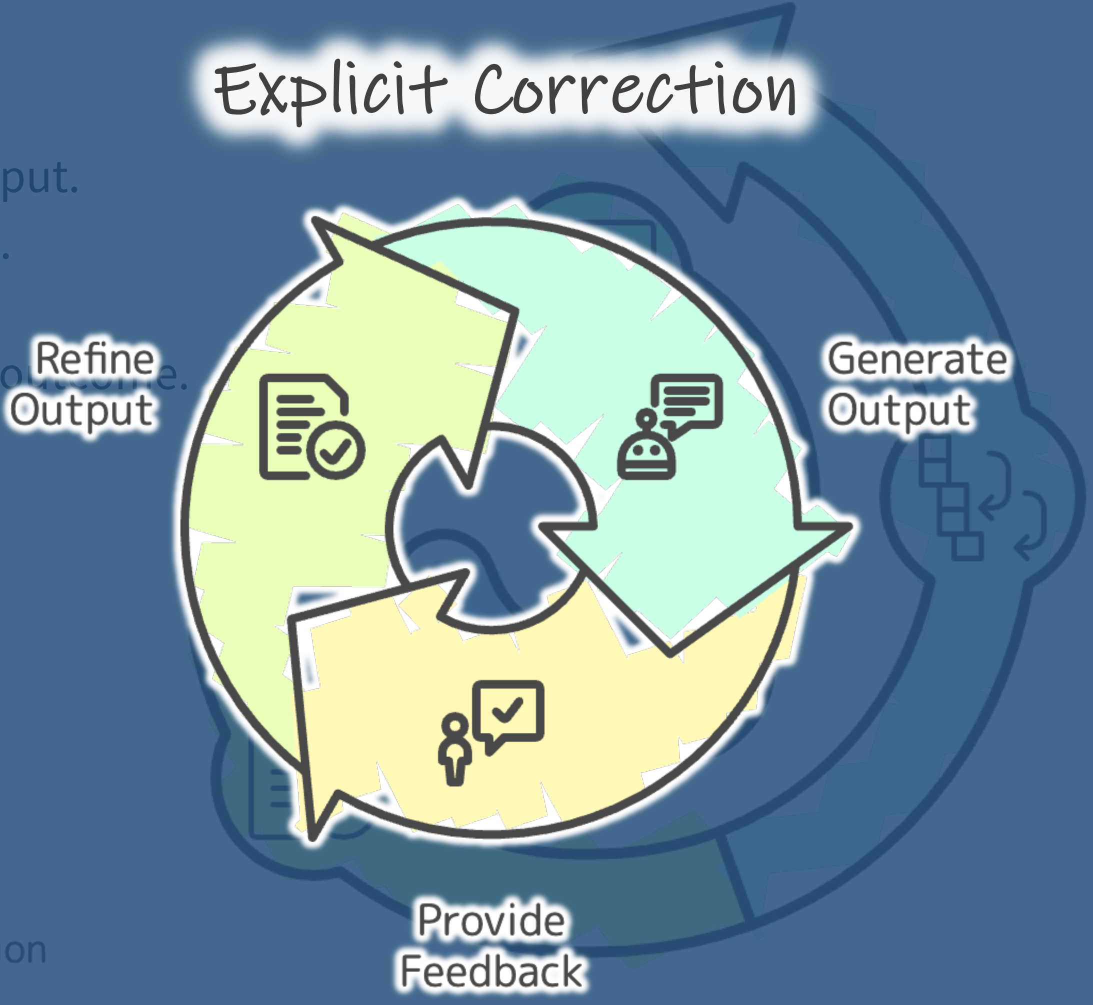
Implicit vs. Explicit Correction

- 3 (or 4) Key Components
  - Output:
  - Feedback: Identifying areas for improvement in the output.
  - Refinement: App[illegible]back to refine the output.
    - Improveme[illegible], Adjust[illegible]on, …
  - Ite[illegible]: R[illegible]ting the p[illegible]iev[illegible]red outcome.

Implicit Correction



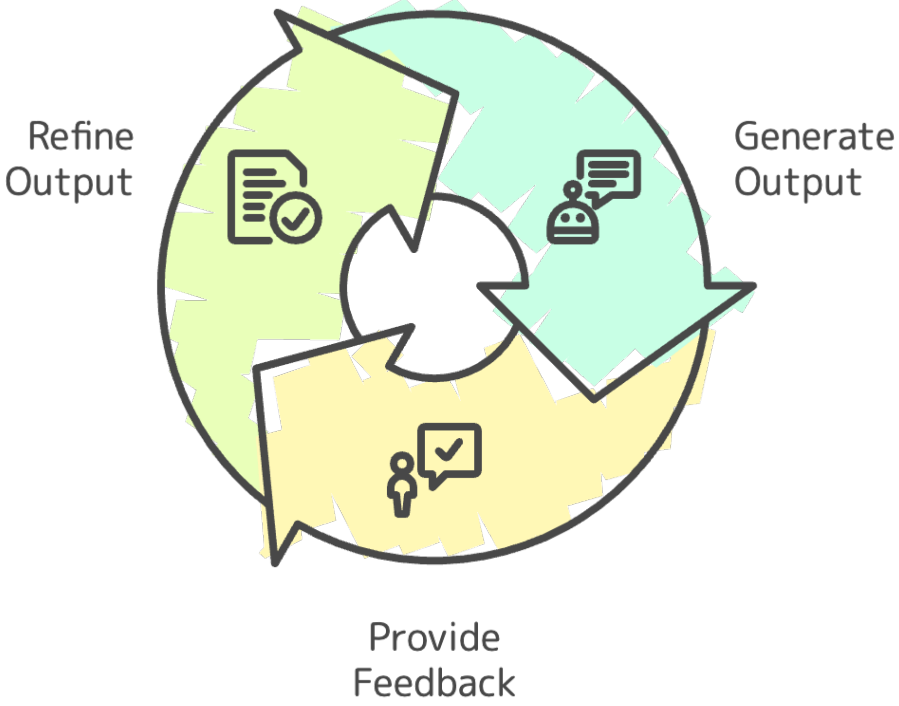
Explicit Correction



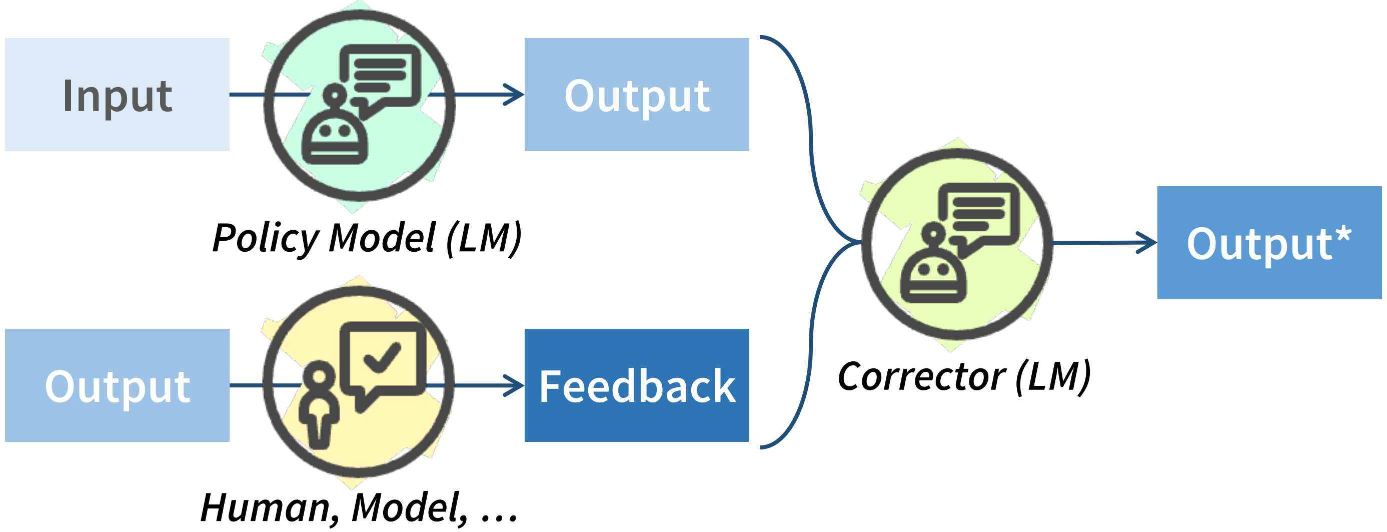
Iterative Correction Process

Explicit Correction

- **Explicit Correction: A Corrector is required to perform Refinement process**
  - #1 Trained Corrector
    - The Corrector is trained using Feedback-Refinement data in a supervised learning manner.
    - Large-scale supervision or human-annotated data is required.

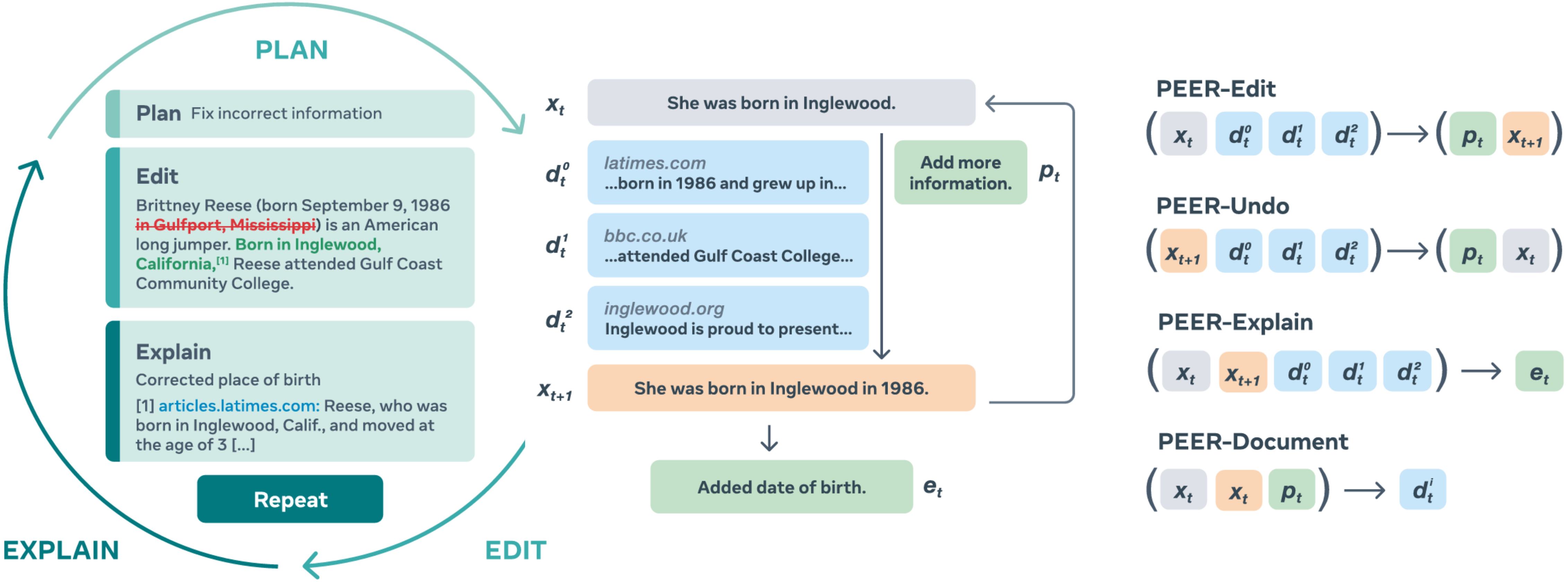


SFT Corrector



Explicit Correction

Schick et al. (Meta AI, CMU et al.) “PEER: A Collaborative Language Model” (ICLR 2023)


PLAN
EDIT
EXPLAIN
Repeat
x_t
d_t^0
d_t^1
d_t^2
x_{t+1}
p_t
e_t
PEER-Edit
$(x_t\ d_t^0\ d_t^1\ d_t^2) \longrightarrow (p_t\ x_{t+1})$
PEER-Undo
$(x_{t+1}\ d_t^0\ d_t^1\ d_t^2) \longrightarrow (p_t\ x_t)$
PEER-Explain
$(x_t\ x_{t+1}\ d_t^0\ d_t^1\ d_t^2) \longrightarrow e_t$
PEER-Document
$(x_t\ x_t\ p_t) \longrightarrow d_t^i$

Plan Fix incorrect information

Edit

Brittney Reese (born September 9, 1986 ~~in Gulfport, Mississippi~~) is an American long jumper. Born in Inglewood, California,[1] Reese attended Gulf Coast Community College.

Explain

Corrected place of birth

[1] articles.latimes.com: Reese, who was born in Inglewood, Calif., and moved at the age of 3 [...]

She was born in Inglewood.

latimes.com
...born in 1986 and grew up in...

bbc.co.uk
...attended Gulf Coast College...

inglewood.org
Inglewood is proud to present...

Add more information.

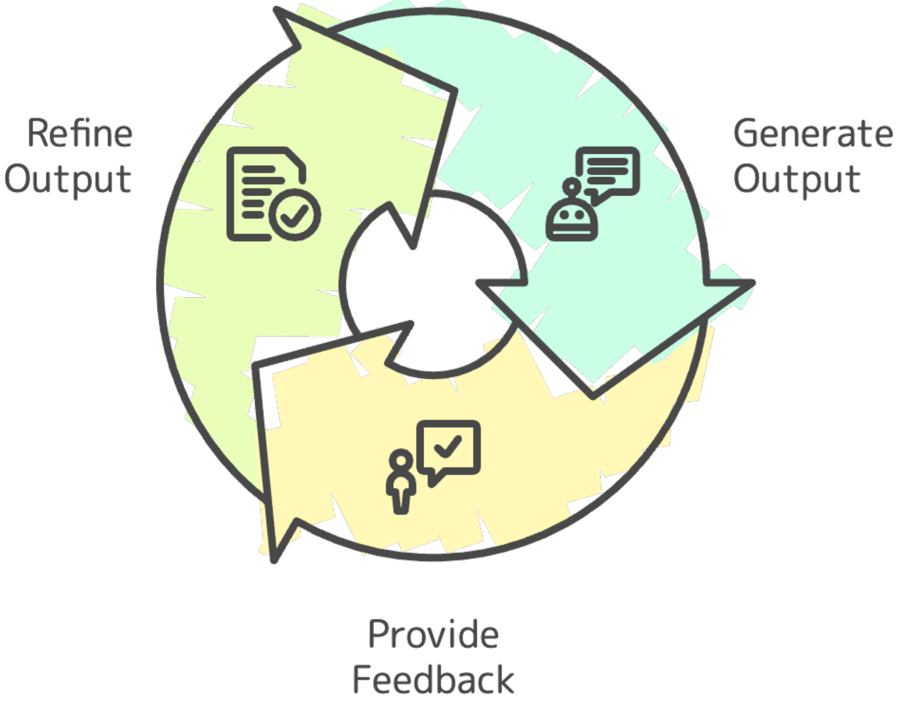
She was born in Inglewood in 1986.

Added date of birth.

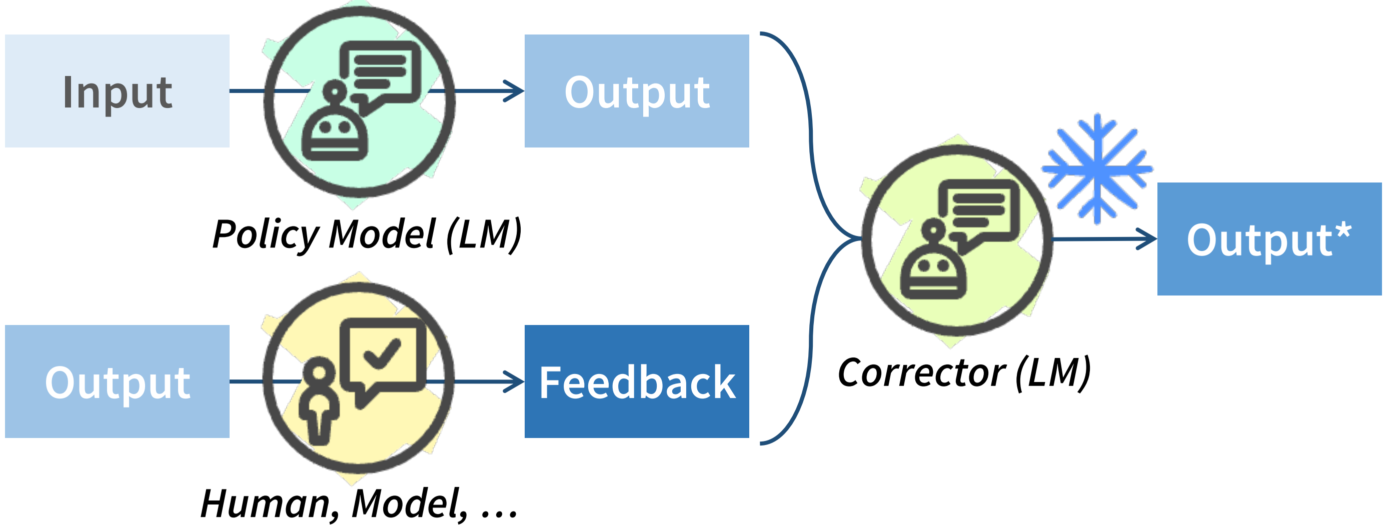
Plan→ Edit → Explain→ Repeat

Explicit Correction

- **Explicit Correction: A Corrector is required to perform Refinement process**
  - #2 Prompted Corrector
    - The Corrector is NOT trained.
    - Correction is performed through **model prompting**.

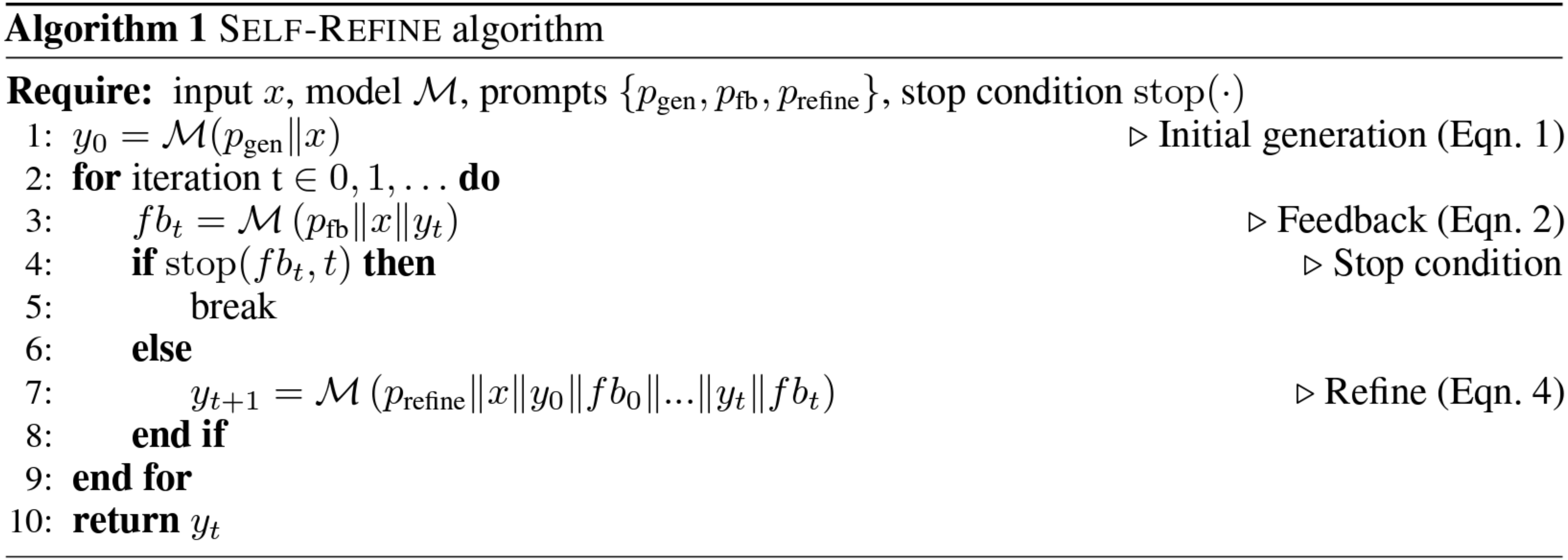


ICL Refinder



Explicit Correction

Madaan et al. (CMU et al.) “Self-Refine: Iterative Refinement with Self-Feedback” (NeurIPS2023)



Algorithm 1 SELF-REFINE algorithm

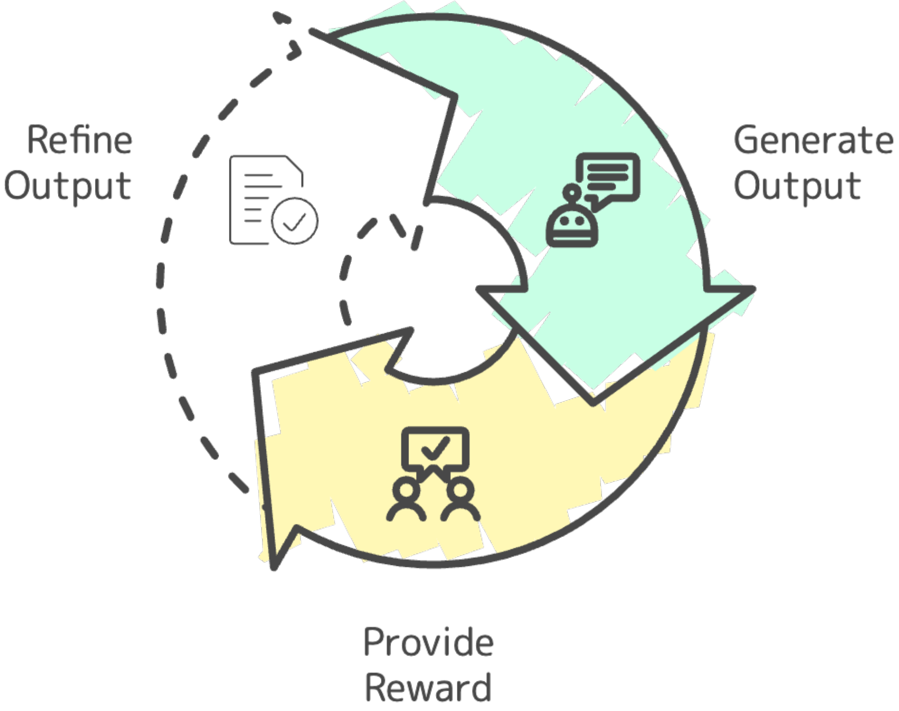
```
Require: input $x$, model $\mathcal{M}$, prompts $\{p_{gen}, p_{fb}, p_{refine}\}$, stop condition $stop(\cdot)$
$y_0 = \mathcal{M}(p_{gen} \| x)$                                ▷ Initial generation (Eqn. 1)
for iteration t $\in 0, 1, \ldots$ do
    $fb_t = \mathcal{M}(p_{fb} \| x \| y_t)$                      ▷ Feedback (Eqn. 2)
    if $stop(fb_t, t)$ then                                       ▷ Stop condition
        break
    else
        $y_{t+1} = \mathcal{M}(p_{refine} \| x \| y_0 \| fb_0 \| \ldots \| y_t \| fb_t)$    ▷ Refine (Eqn. 4)
    end if
end for
return $y_t$
```

SELF-REFINE iteratively improves outputs from LLMs through a process of iterative creation with feedback description.

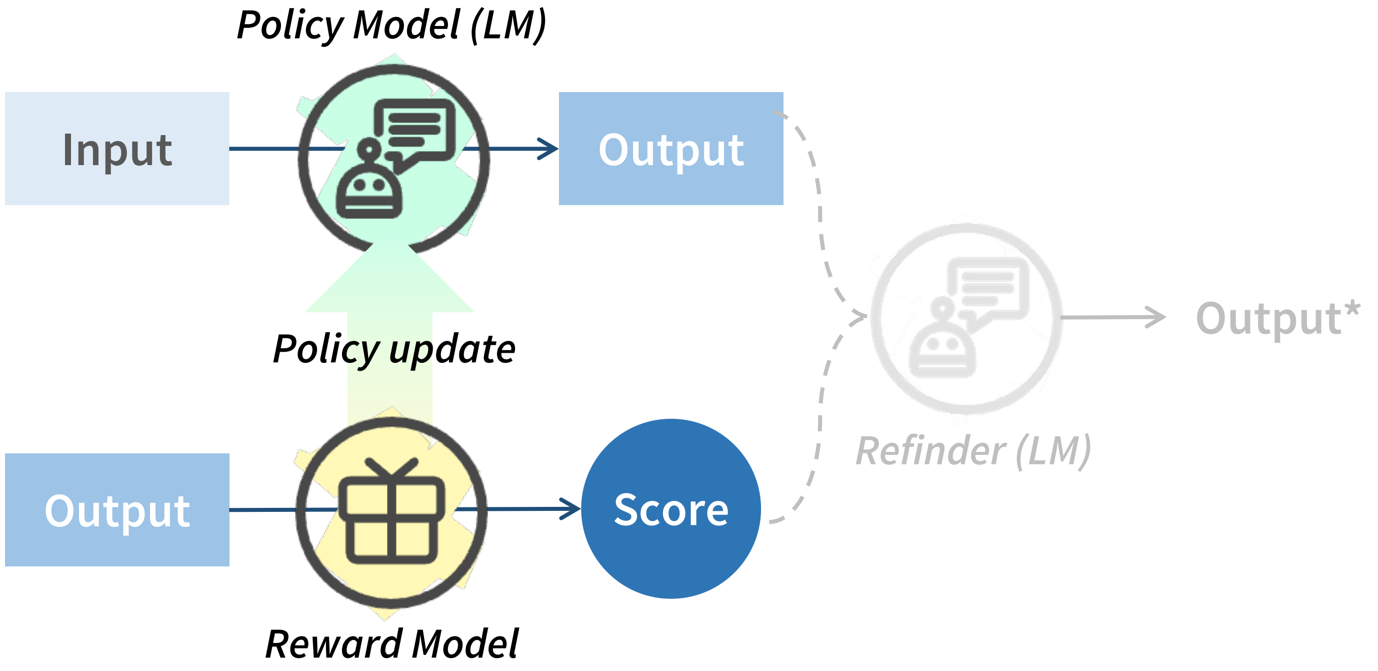
Output → Feedback → Refine → … : Online, On-Policy

Implicit Correction

- **Implicit Correction: Refinement does not explicitly exist**
  - #3 RL Manner
    - Responses are optimized through **reward function optimization**.
    - Training a **reward model** is required.



Reinforcemnet Learning



SCoRe

Key Challenges of Self-Correction

A. *Oracle Feedback (or Ground Truth)*

- Oracle feedback requires significant resources like expert labeling and validation.
- It’s often unavailable during inference, where real-world conditions differ from training.

B. *Context Dependency*

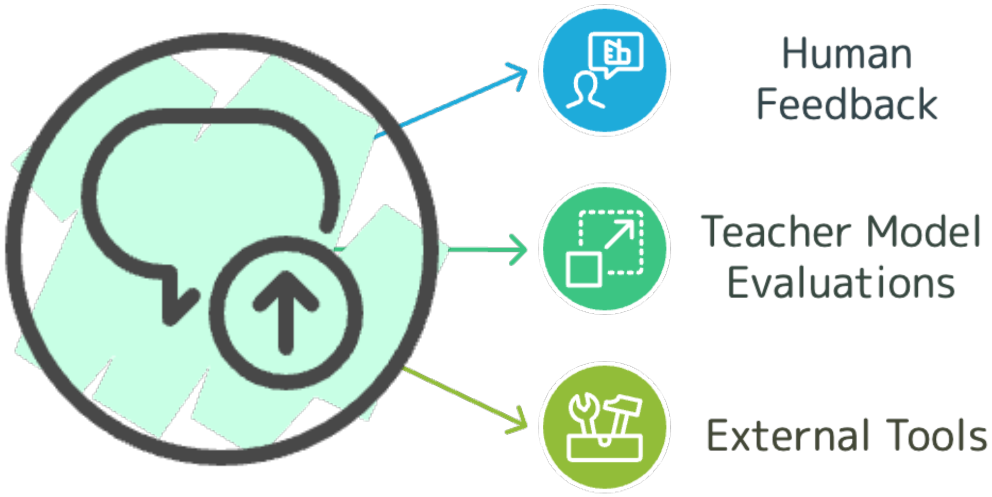
- Feedback tailored to specific tasks often lacks generalization to diverse contexts or real-world scenarios.

C. *Unclear Criteria & Complex Correction Methods*

- The criteria for evaluating outputs are often unclear.
- It’s also hard to create methods for models to find and fix their mistakes.

Intrinsic vs. Extrinsic Self-Correction

Huang, Chen et al. (Google DeepMind et al.) “Large Language Models Cannot Self-Correct Reasoning Yet” (ICLR2024)



Intrinsic Self-Correction

LLM corrects its initial response using only its own abilities:
it relies solely on the knowledge and parameters
inside the model to improve its response,
w/o any external information or feedback.

Extrinsic Self-Correction

LLM corrects by utilizing external inputs to modify its output.
: feedback from humans, evaluations from other models, and
information from external tools and knowledge sources
(search engines, calculators, etc.).

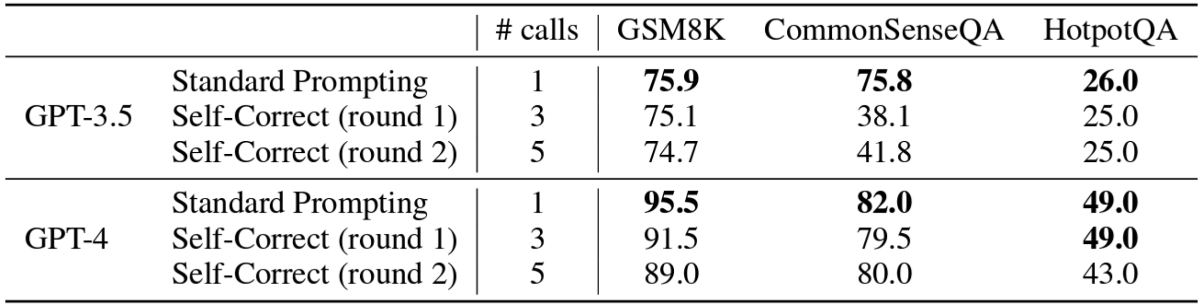
Effectively leveraging external feedback is crucial for improving LLMs' self-correction capabilities.

Intrinsic vs. Extrinsic Self-Correction

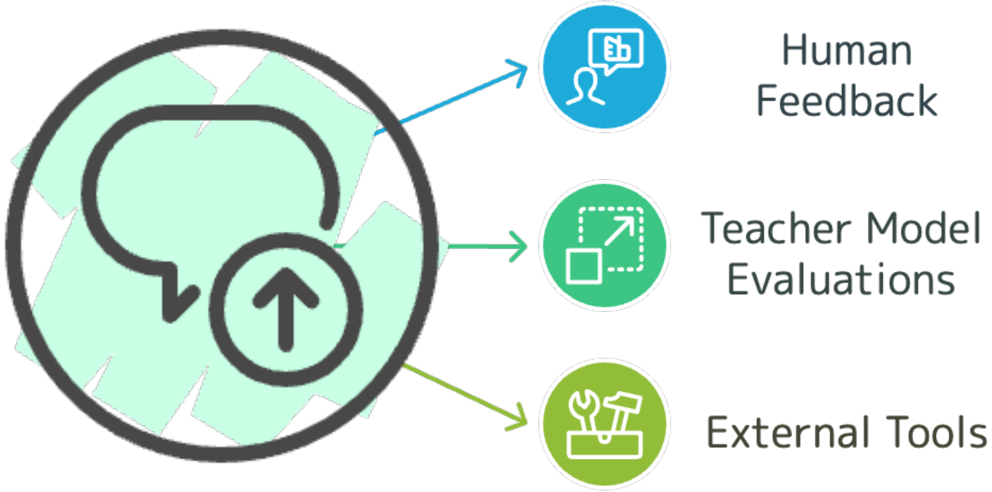
Huang, Chen et al. (Google DeepMind et al.) “Large Language Models Cannot Self-Correct Reasoning Yet” (ICLR2024)

LLMs are NOT YET
capable of **accurately evaluating**
the correctness of **their own responses.**

Table 3: Results of GPT-3.5 and GPT-4 on reasoning benchmarks with intrinsic self-correction.



		# calls	GSM8K	CommonSenseQA	HotpotQA
GPT-3.5	Standard Prompting	1	**75.9**	**75.8**	**26.0**
	Self-Correct (round 1)	3	75.1	38.1	25.0
	Self-Correct (round 2)	5	74.7	41.8	25.0
GPT-4	Standard Prompting	1	**95.5**	**82.0**	**49.0**
	Self-Correct (round 1)	3	91.5	79.5	**49.0**
	Self-Correct (round 2)	5	89.0	80.0	43.0



Human Feedback

Teacher Model Evaluations

External Tools

Extrinsic Self-Correction

LLM corrects by utilizing external inputs to modify its output.
: feedback from humans, evaluations from other models, and information from external tools and knowledge sources (search engines, calculators, etc.).

Effectively leveraging external feedback is crucial for improving LLMs' self-correction capabilities.

Paper Review : 2025 Winter Lab Seminar

Recursive IntroSpEction: Teaching Language Model Agents How to Self-Improve

Yuxiao Qu, Tianjun Zhang, Naman Garg, Aviral Kumar (CMU, UC Berkeley, MultiOn)

Accepted to NeurIPS 2024

Yejin Yoon

Natural Language Processing Lab.,
Hanyang University.

Problem States

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Problem Setup and Preliminaries**
  - Given Dataset : $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i^*)\}_{i=1}^N$ $\rightarrow$ $\mathcal{M}:$ $\rho(\boldsymbol{s}_0) = \mathrm{Unif}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$
    - $\boldsymbol{x}_i$: problems
    - $\boldsymbol{y}_i^*$: oracle responses

$$P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = \delta\left(\boldsymbol{s}' = \mathrm{concat}[\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{f}]\right)$$

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

  - Policy : $\mathrm{LLM}\ \pi_\theta(\cdot|[\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:t}, p_{1:t}])$
    - $\hat{\boldsymbol{y}}_{1:t}$: previous model attempts at the problem
    - $p_{1:t}$: auxiliary instructions

      e.g. instruction to find a mistake and improve the response; or additional compiler feedback from the environment

  - Objective

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

Problem States

📄 Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Multi-turn MDP**
  - **M**arkov **D**ecision **P**rocess: a model for sequential decision making when outcomes are uncertain.

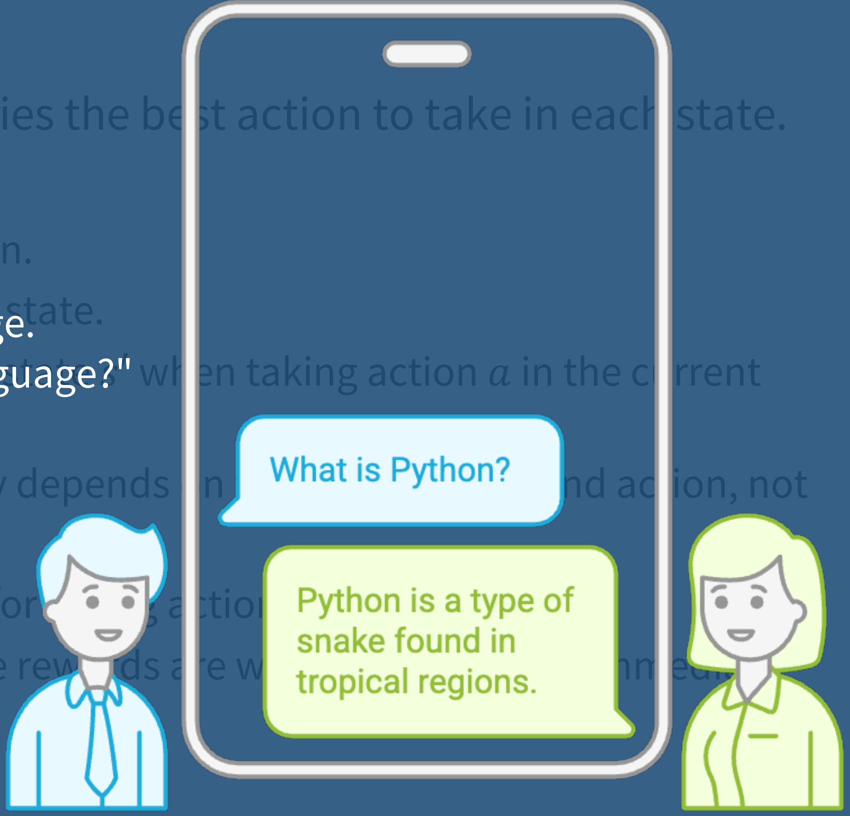
  - Goal: Agent’s goal is to find the optimal policy $\pi(a|s)$, which specifies the best action to take in each state.
  - Components
    1. State Space S : The set of all possible states the environment can be in.
    2. Action Space A : The set of all actions the agent can take in any given state.
    3. Transition Probability $P(s'|s,a)$: The probability of moving to a new state s' when taking action a in the current state s.
       - This satisfies the ***Markov property***, which means the next state only depends on the current state and action, not on the sequence of past states: $P(s'|s,a) = P(s'|s,a,\text{past states})$
    4. Reward Function $R(s,a)$: The immediate reward the agent receives for taking action a in state s.
    5. Discount Factor $\gamma \in [0,1]$: A factor that determines how much future rewards are worth compared to immediate rewards.

Problem States

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Multi-turn MDP**

1. **Current State: S_0**
    - Dialogue context: The user asked about Python, and the model gave a wrong response (snake-related).
    - Error detected implicitly or through user feedback.
2. **Action: A**
    - A_1: Leave the response as-is.
    - A_2: Correct the response to refer to Python as a programming language.
    - A_3: Clarify with the user: "Did you mean Python the programming language?"
3. **Reward: $R(s, a)$**
    - $R(A_1) = -1$: Negative reward for maintaining the wrong response.
    - $R(A_2) = 2$: Positive reward for correcting the mistake.
    - $R(A_3) = 1$: Moderate reward for asking for clarification.
4. **State Transition**
    - If A_2 (correction) is taken, the new state S_1 might include user satisfaction and a positive reaction.
    - If A_1 is taken, S_1 could involve user frustration or explicit correction.



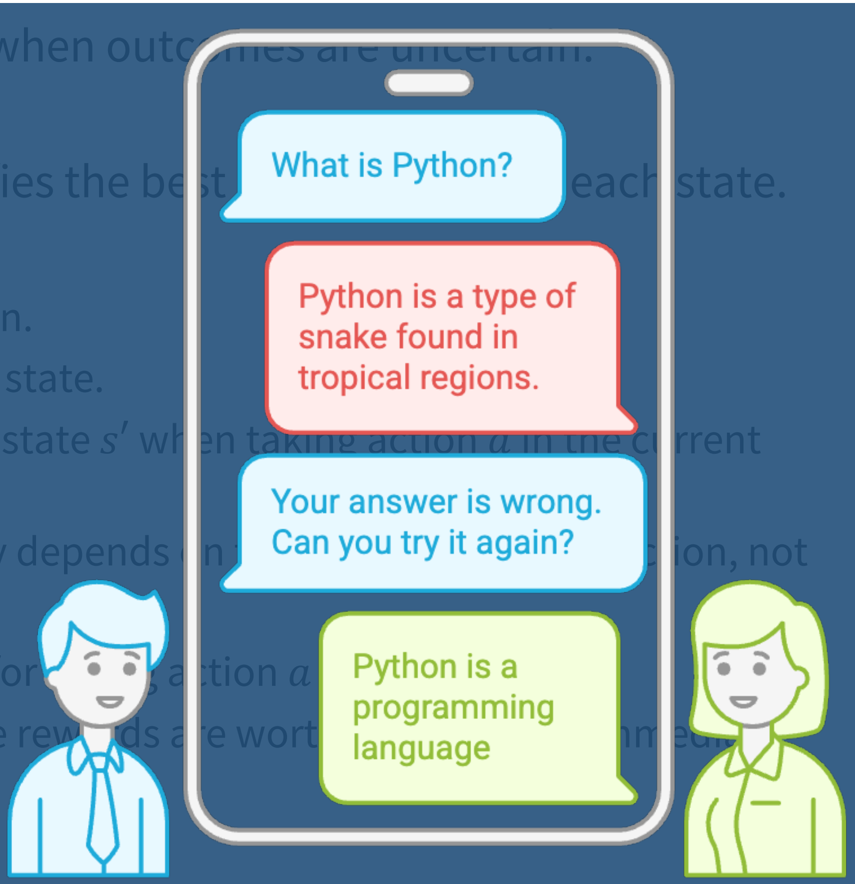
Problem States

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Multi-turn MDP**

5. Optimal Policy: π^*

The agent learns through reinforcement to choose actions that maximize rewards, favoring corrections when needed.



What is Python?

Python is a type of snake found in tropical regions.

Your answer is wrong. Can you try it again?

Python is a programming language

Problem States

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Problem Setup and Preliminaries**
  - Giv
  - Poli
  - Obj

nment

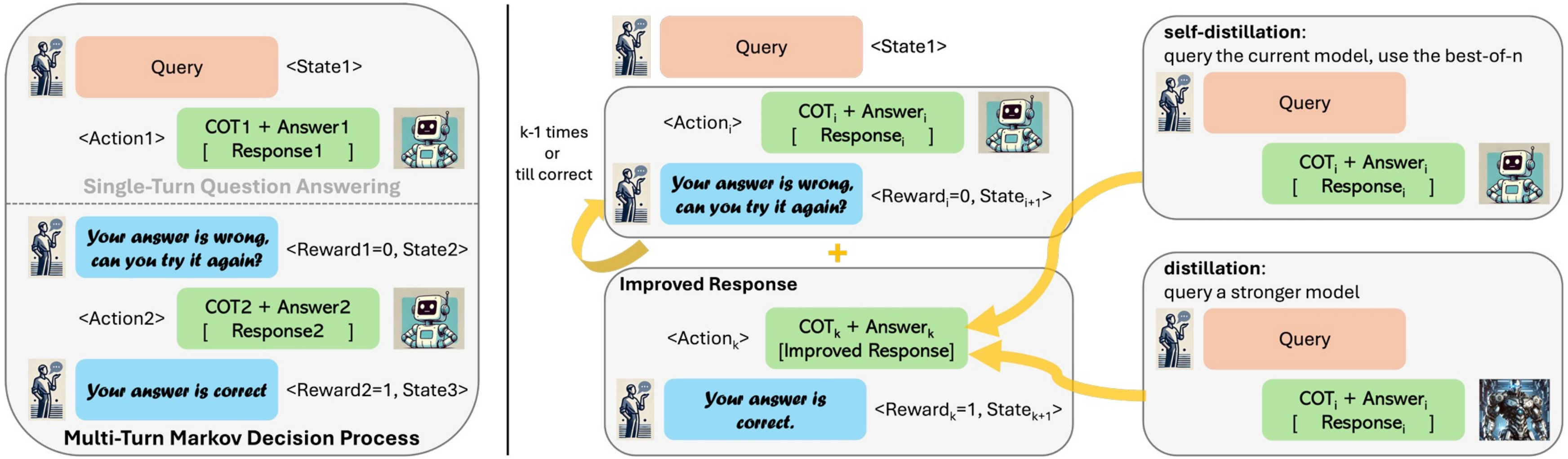


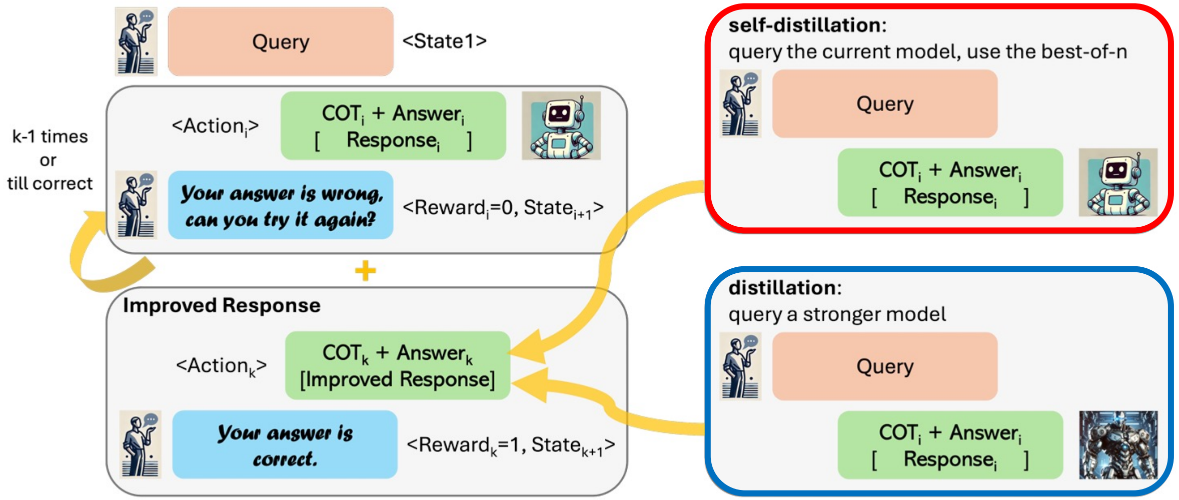
Figure 2: ***Left: Problem formulation.*** We convert single-turn problems into multi-turn MDPs as discussed in Section 3.1. The state is given by the prompt, history of prior attempts, and optional feedback from the environment. An action is a response generated from the LLM given the state of multi-turn interaction so far. ***Right: Data collection.*** We collect data by unrolling the current model $k-1$ times followed by an improved version of the response, which is obtained by either (1) **self-distillation**: sample multiple responses from the current model, and use the best response, or (2) **distillation**: obtain oracle responses by querying a more capable model. In either case, RISE then trains on the generated data.

Suggestions

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Suggestions**
  - Step 1: Data Collection for Self-improvement

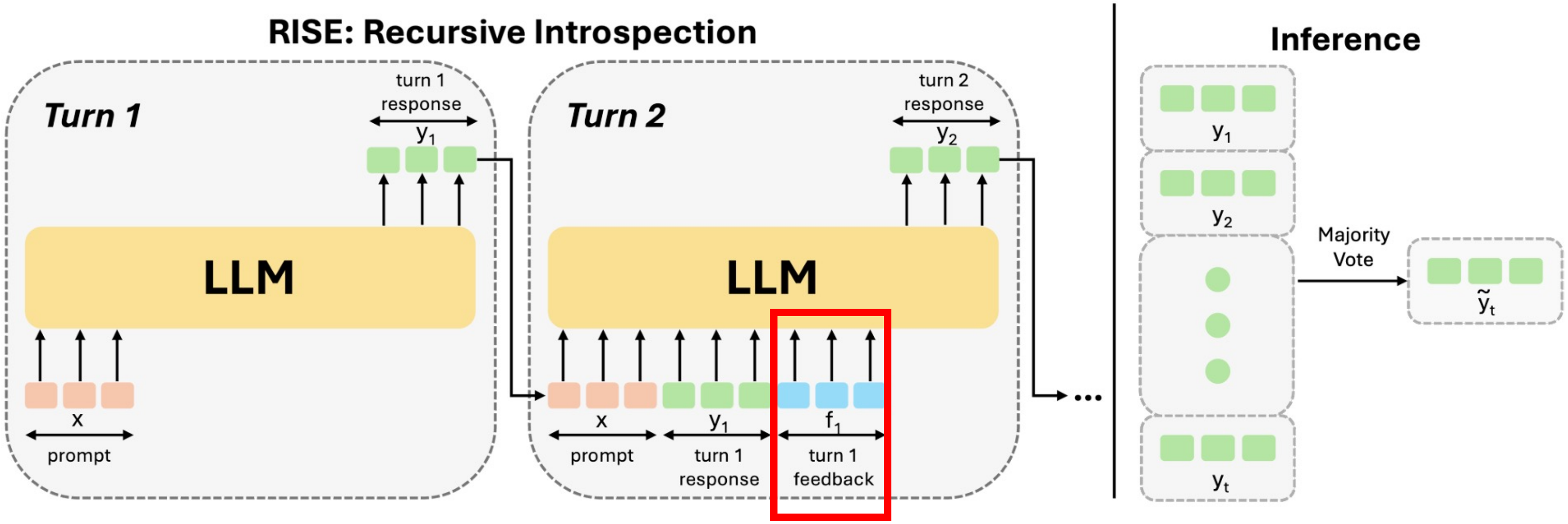
1. Distillation $\tilde{\mathcal{D}}_{\text{on-policy + distill}} := \left\{ \left\{ \left(\boldsymbol{s}_t^i, \tilde{\boldsymbol{y}}_t^i, f_t^i, \tilde{r}_t^i \right) \right\}_{t=1}^{T} \right\}_{i=1}^{|\mathcal{D}|}.$
2. Self-Distillation $\tilde{\mathcal{D}}_{\text{on-policy + self-distillation}} := \left\{ \left\{ \left(\boldsymbol{s}_{t+1}^i, \tilde{\boldsymbol{y}}_t^i[m], f_{t+1}^i, \tilde{r}_t^i[m] \right) \right\}_{t=0}^{T-1} \right\}_{i=1}^{|\mathcal{D}|}.$



Suggestions

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Suggestions**
  - Step 1: Data Collection for Self-improvement
  - Step 2: Policy Improvement

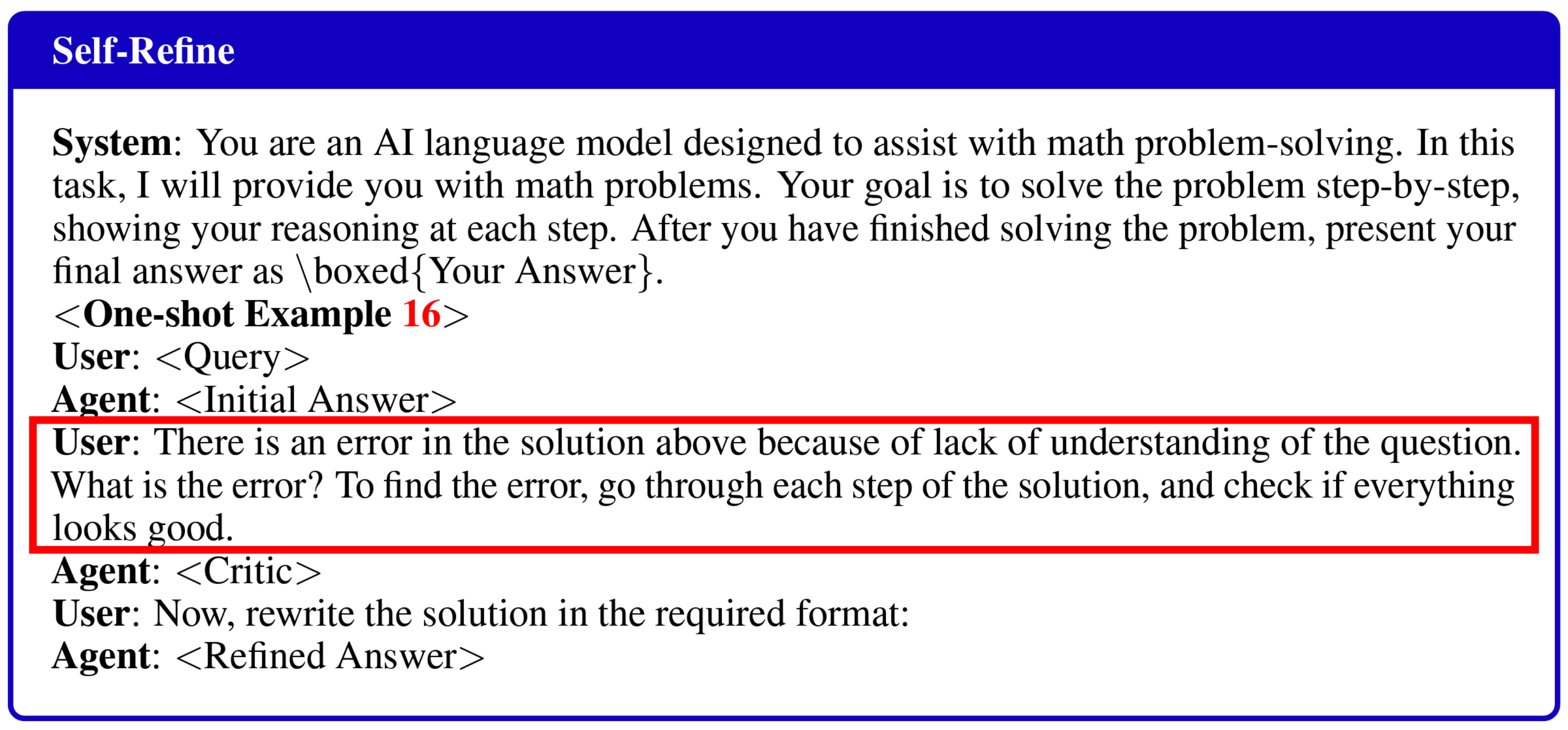


$$\text{Reward-weighted RL:}\quad \max_{\theta}\ \mathbb{E}_{\boldsymbol{x}_i\sim\tilde{\mathcal{D}}}\left[\sum_{t=1}^{T}\log\pi_\theta(\tilde{\boldsymbol{y}}_t^i|\boldsymbol{s}_t^i)\cdot\underbrace{\exp(r_i^t/\tau)}_{\text{weight}}\right]$$

Suggestions

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Suggestions**
  - Step 1: Data Collection for Self-improvement
  - Step 2: Po



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{Your Answer}.
<**One-shot Example 16**>
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>

Results

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Suggestions**
  - Step 1: Data Collection for Self-improvement

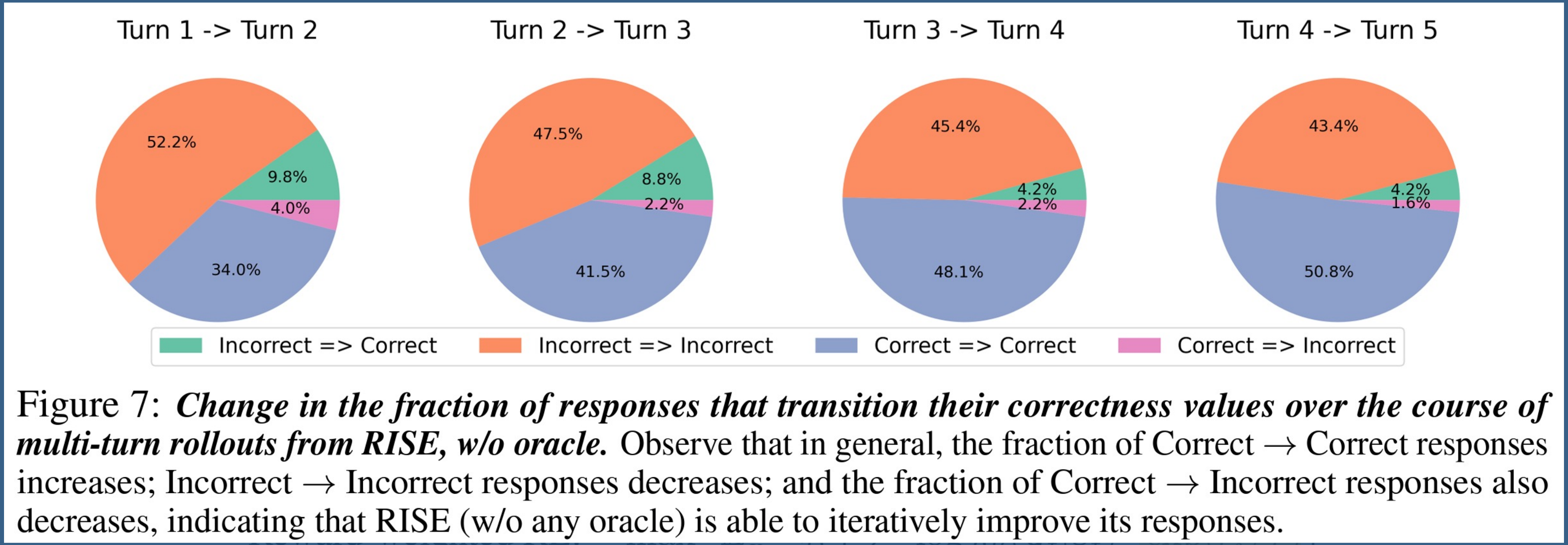
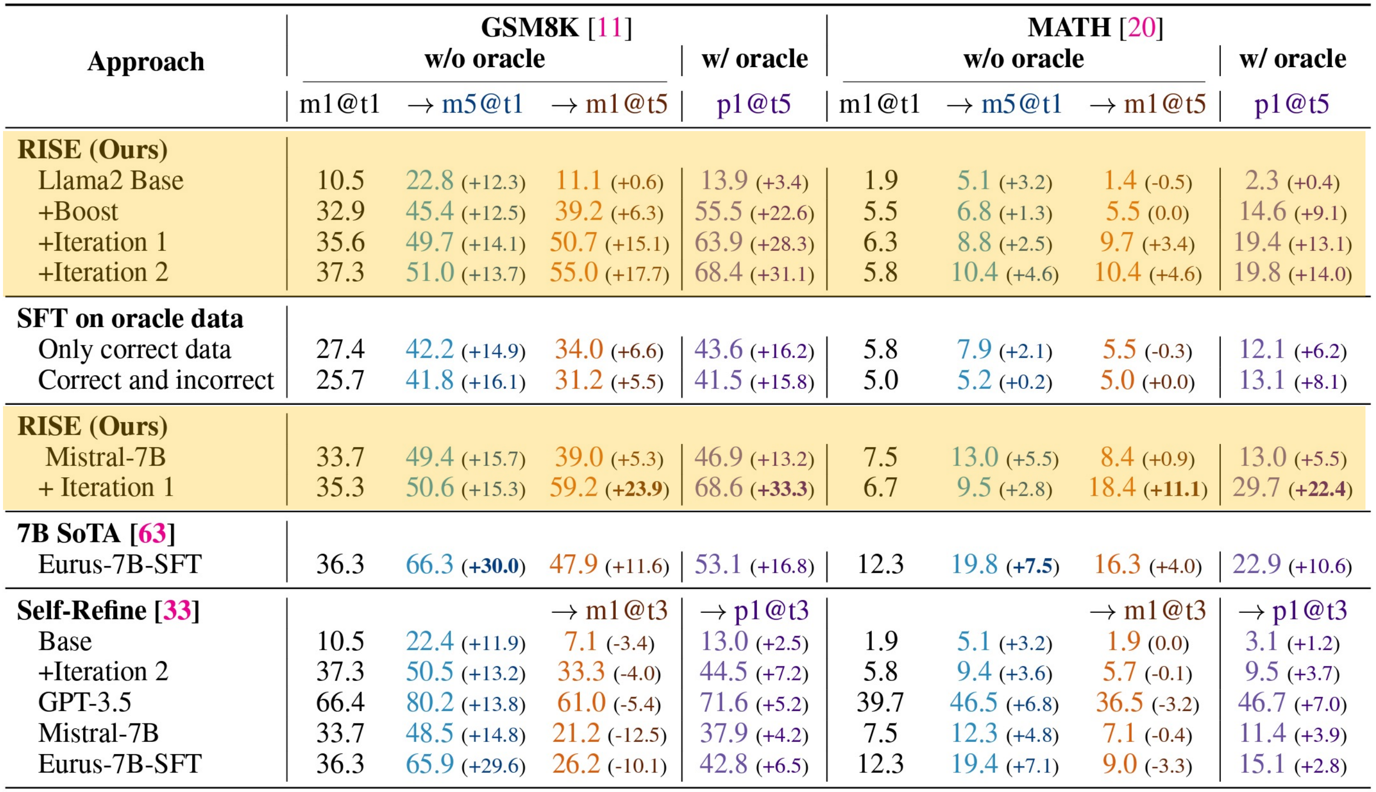


Figure 7: ***Change in the fraction of responses that transition their correctness values over the course of multi-turn rollouts from RISE, w/o oracle.*** Observe that in general, the fraction of Correct → Correct responses increases; Incorrect → Incorrect responses decreases; and the fraction of Correct → Incorrect responses also decreases, indicating that RISE (w/o any oracle) is able to iteratively improve its responses.

Results

Qu et al. (CMU et al.) “**R**ecursive **I**ntro**S**p**E**ction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

- **Results**
  - Baseline
    - Self-Refine: prompts a base model to critique and revise its mistakes
    - GloRE: trains a sparate reward model to locate errors and a refinement model to improve responses of a base LLM
  - Metrics
    - maj@N: majority voting
    - pass@K: proportion of inputs for which at least one of the k outputs is correct



Approach	GSM8K [11]				MATH [20]			
	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)
SFT on oracle data								
Only correct data	27.4	42.2 (+14.9)	34.0 (+6.6)	43.6 (+16.2)	5.8	7.9 (+2.1)	5.5 (-0.3)	12.1 (+6.2)
Correct and incorrect	25.7	41.8 (+16.1)	31.2 (+5.5)	41.5 (+15.8)	5.0	5.2 (+0.2)	5.0 (+0.0)	13.1 (+8.1)
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 [63]								
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 [33]			→ 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)

* Boost: knowledge boosting

“… these models often could not adhere to response style and instructions for improving their responses when generating on-policy data.”

Summary

1. Goal

- LLMs self-improve over multi-turns w/o relying on external feedback or specific prompts.

2. 3 Suggestions

1. Converts single-turn question-answer tasks into a **multi-turn Markov Decision Process (MDP)**.
2. Leverages **on-policy learning** with rollouts generated by the model itself.
3. Uses both **optimal and suboptimal data** for fine-tuning via **reward-weighted regression**.

3. Effects

- Achieves consistent improvement in model performance across multiple turns:Outperforms existing methods like chain-of-thought (CoT) and self-refinement techniques.
- Demonstrates generalizability to other tasks and datasets (e.g., mathematical reasoning benchmarks).
- Can solve problems that standard approaches fail to address, even with larger sampling budgets.

Paper Review : 2025 Winter Lab Seminar

SCoRe : Training Language Models to Self-Correct via Reinforcement Learning

Kumar, Zhuang, Agarwal et al. (Google DeepMind)

Submitted to ICLR 2025

Yejin Yoon

Natural Language Processing Lab.,
Hanyang University.

Problem States

Limitations of Previous Work (1/2)

- Supervised fine-tuning (SFT) or reinforcement learning (RL) fine-tuning trains for y^* **at a single step**.

Supervised Fine-tuning

- SFT aims to train a model to predict the correct response y^* given an input x.
- The training objective for SFT focuses on single-turn optimization:

$$\min_{\theta} \mathbb{E}_{(x,y^*)\sim\mathcal{D}} \left[\mathcal{L}(f_\theta(x), y^*)\right]$$

- Limitations
  - SFT does not teach the model to self-correct its responses.
  - If the model fails on the first attempt, it lacks the ability to revise or improve its output.

Reinforcement Learning

- RL fine-tuning trains the model to optimize its output based on a reward function.
- The objective for RL-based methods is:

$$\max_{\theta} \mathbb{E}_{x,y^*\sim\mathcal{D},y\sim\pi_\theta(\cdot|x)} \left[r(y, y^*)\right]$$

- Limitations
  - Most RL fine-tuning focuses on single-turn learning, optimizing for the correctness of a single output without considering the potential for self-correction.
  - It does not train the model to iteratively refine or improve responses over multiple turns.

SFT: Trains for single-turn correctness, lacks self-correction.
RL Fine-Tuning: Optimizes for a single high-quality response, does not handle iterative refinement.

Problem States

- **Limitations of Previous Work (2/2)**
  - Supervised fine-tuning (SFT) or reinforcement learning (RL) fine-tuning trains for y^* **at a single step**.
  - Comparison with RISE

Common Features:
1. **Use of MDP Structure**: Both approaches use an MDP (Multi-Turn) setup to facilitate multi-turn interactions.
2. **Self-Correction Learning**: Both enable the model to improve by learning from its errors in previous turns.

Differences:	**SCoRe**	**RISE**
1. Reward Definition in MDP	Cumulative quality of responses across all turns is optimized ($\sum \hat{r}$)	Binary reward based on the accuracy of responses ($r(s,a) = 1 \; or \; 0$)
2. Data Construction	On-policy	On-/Off-policy
3. Base Approach	RL	SFT

Problem States

• Problem Setup and Preliminaries

- Given Dataset : $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i^*)\}_{i=1}^{N}$ → $\mathcal{M} : \rho(\boldsymbol{s}_0) = \mathrm{Unif}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$

$$P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = \delta\left(\boldsymbol{s}' = \mathrm{concat}[\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{f}]\right)$$

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

  - $\boldsymbol{x}_i$: problems
  - $\boldsymbol{y}_i^*$: oracle responses
- Policy : $\mathrm{LLM}\ \pi_\theta(\cdot|[\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:t}, p_{1:t}])$
  - $\hat{\boldsymbol{y}}_{1:t}$: previous model attempts at the problem
  - $p_{1:t}$: auxiliary instructions

    e.g. instruction to find a mistake and improve the response; or additional compiler feedback from the environment
- Objective

$$\max_{\pi_\theta} \mathbb{E}_{\boldsymbol{x}, \boldsymbol{y}^* \sim \mathcal{D}, \hat{\boldsymbol{y}}_{l+1} \sim \pi_\theta(\cdot|[\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:l}, p_{1:l}])} \left[\sum_{i=1}^{l+1} \widehat{r}\left(\hat{\boldsymbol{y}}_i, \boldsymbol{y}^*\right) \right]$$

  - RISE Objective: $\max_{\pi_\theta} \sum_{i=1}^{L} \mathbb{E}_{\boldsymbol{x}, \boldsymbol{y}^* \sim \mathcal{D}, \hat{\boldsymbol{y}}_i \sim \pi_\theta(\cdot|[\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:i-1}, p_{1:i-1}])} \left[\mathbb{I}\left(\hat{\boldsymbol{y}}_i == \boldsymbol{y}^*\right) \right].$

Problem States

- **Problem Setup and Preliminaries**
  - Given Dataset : $\mathcal{D} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i^*)\}_{i=1}^N$ → $\mathcal{M}: \rho(\boldsymbol{s}_0) = \text{Unif}(\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N)$
    - $\boldsymbol{x}_i$: problems
    - $\boldsymbol{y}_i^*$: oracle responses

$$P(\boldsymbol{s}'|\boldsymbol{s}, \boldsymbol{a}) = \delta\left(\boldsymbol{s}' = \text{concat}[\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{f}]\right)$$
$$r(\boldsymbol{s}, \boldsymbol{a}) = \mathbf{1}\left(\boldsymbol{a} = \boldsymbol{y}_i^* \text{ if } \boldsymbol{x}_i \in \boldsymbol{s}\right).$$

  - Policy : $\text{LLM } \pi_\theta(\cdot|[\boldsymbol{x}, \hat{\boldsymbol{y}}_{1:t}, p_{1:t}])$
    - $\hat{\boldsymbol{y}}_{1:t}$: previous model attempts at the problem
    - $p_{1:t}$: auxiliary instructions

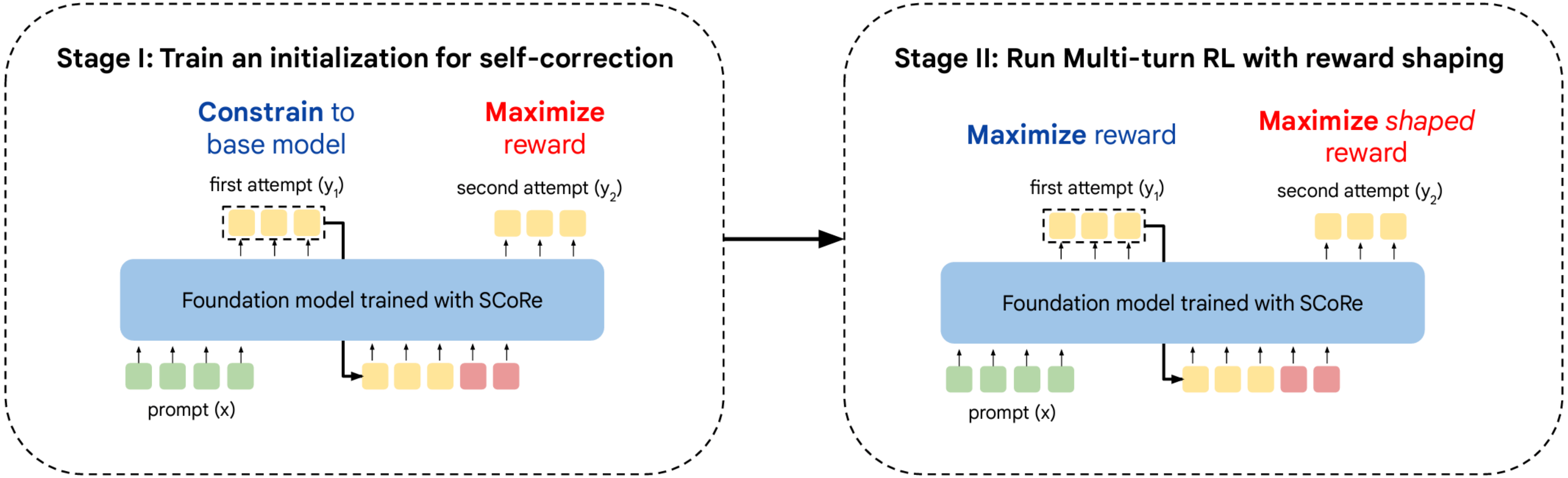
      e.g. instruction to find a mistake and improve the response; or additional compiler feedback from the environment

  - Objective: uses a REINFORCE policy gradient training approach with a **KL-divergence penalty** against a fixed model [Ahmadian et al., "Back to basics: ~"]

$$\max_\theta \; \mathbb{E}_{\boldsymbol{x}_t, \boldsymbol{y}_t \sim \pi_\theta(\cdot|\boldsymbol{x}_t)}\left[\widehat{r}(\boldsymbol{y}_t, \boldsymbol{y}^*) - \beta_1 D_{KL}(\pi_\theta(\cdot|\boldsymbol{x}_t)||\pi_{\text{ref}}(\cdot|\boldsymbol{x}_t))\right]$$

Suggestions

- **Suggestions**



Stage I: Train an initialization for self-correction

Constrain to base model

Maximize reward

first attempt (y_1)

second attempt (y_2)

Foundation model trained with SCoRe

prompt (x)

Stage II: Run Multi-turn RL with reward shaping

Maximize reward

Maximize *shaped* reward

first attempt (y_1)

second attempt (y_2)

Foundation model trained with SCoRe

prompt (x)

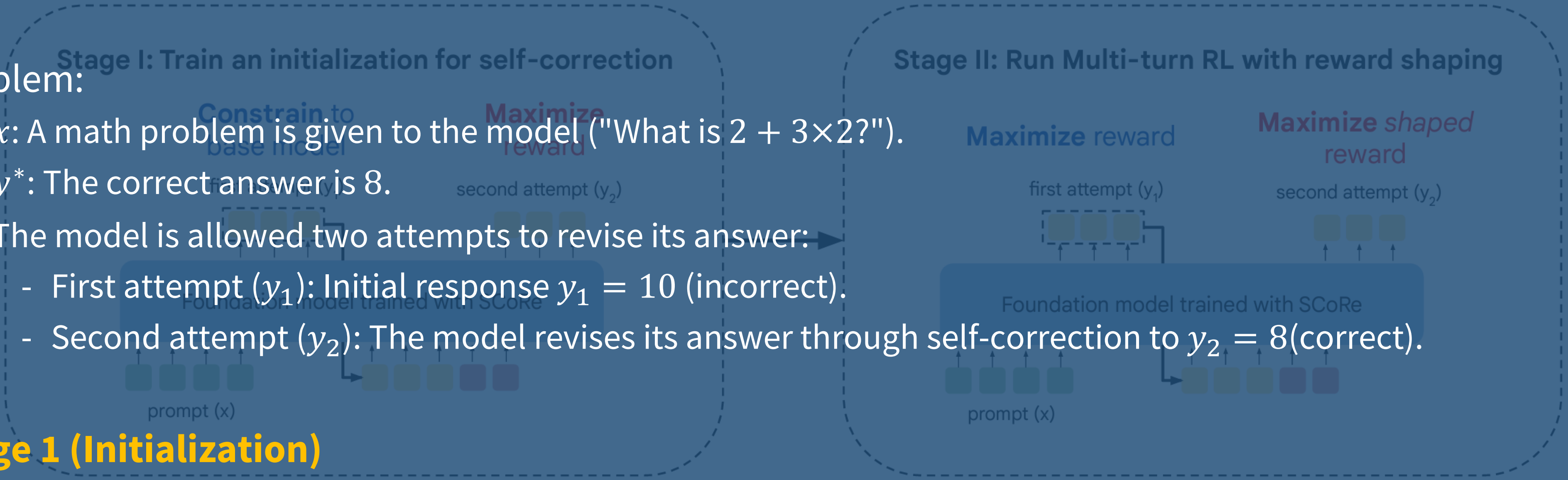
- **Stage 1 (Initialization)** : instead of running SFT (which produces pathological amplification of biases) to initialize RL training, train a good initialization that can produce high-reward responses in the second-attempt while mimicking the base model's response at the first attempt.

$$\max_{\theta} \; \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{y}_1 \sim \pi_\theta(\cdot|\boldsymbol{x}),\boldsymbol{y}_2 \sim \pi_\theta(\cdot|[\boldsymbol{x}_1,p_1])}\left[\widehat{r}(\boldsymbol{y}_2,\boldsymbol{y}^*) - \beta_2 D_{KL}\left(\pi_\theta(\cdot||\boldsymbol{x}_1)||\pi_{\text{ref}}(\cdot|\boldsymbol{x}_1)\right)\right]$$

  - Base objective: $$\max_{\theta} \; \mathbb{E}_{\boldsymbol{x}_t,\boldsymbol{y}_t \sim \pi_\theta(\cdot|\boldsymbol{x}_t)}\left[\widehat{r}(\boldsymbol{y}_t,\boldsymbol{y}^*) - \beta_1 D_{KL}(\pi_\theta(\cdot|\boldsymbol{x}_t)||\pi_{\text{ref}}(\cdot|\boldsymbol{x}_t))\right]$$

Suggestions

- **Example: Solving a math problem**



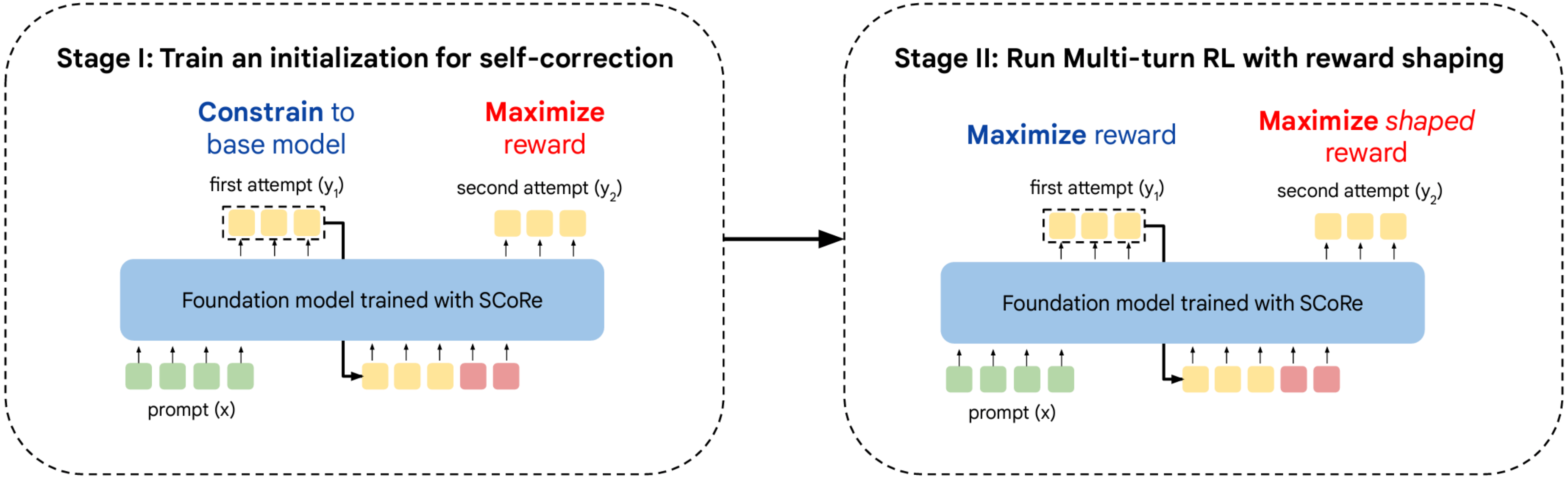
- Problem:
  - x: A math problem is given to the model ("What is $2 + 3\times2$?").
  - y^*: The correct answer is 8.
  - The model is allowed two attempts to revise its answer:
    - First attempt (y_1): Initial response $y_1 = 10$ (incorrect).
    - Second attempt (y_2): The model revises its answer through self-correction to $y_2 = 8$(correct).
- **Stage 1 (Initialization)**
  - The distribution of the first attempt (y_1) is kept close to the base model while optimizing rewards for the second attempt (y_2).
  - Goal: Decouple the first and second attempts so that the second attempt can focus on self-correction.
  - The model learns to keep $y_1 = 10$ as it is and improve $y_2 = 8$.

$$\max_{\theta} \ \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{y}_1\sim\pi_\theta(\cdot|\boldsymbol{x}),\boldsymbol{y}_2\sim\pi_\theta(\cdot|[\boldsymbol{x}_1,p_1])}\left[\widehat{r}(\boldsymbol{y}_2,\boldsymbol{y}^*) - \beta_2 D_{KL}\left(\pi_\theta(\cdot||\boldsymbol{x}_1)||\pi_{\mathrm{ref}}(\cdot|\boldsymbol{x}_1)\right)\right]$$

$$\max_{\theta} \ \mathbb{E}_{\boldsymbol{x}_t,\boldsymbol{y}_t\sim\pi_\theta(\cdot|\boldsymbol{x}_t)}\left[\widehat{r}(\boldsymbol{y}_t,\boldsymbol{y}^*) - \beta_1 D_{KL}(\pi_\theta(\cdot|\boldsymbol{x}_t)||\pi_{\mathrm{ref}}(\cdot|\boldsymbol{x}_t))\right]$$

Suggestions

• Suggestions



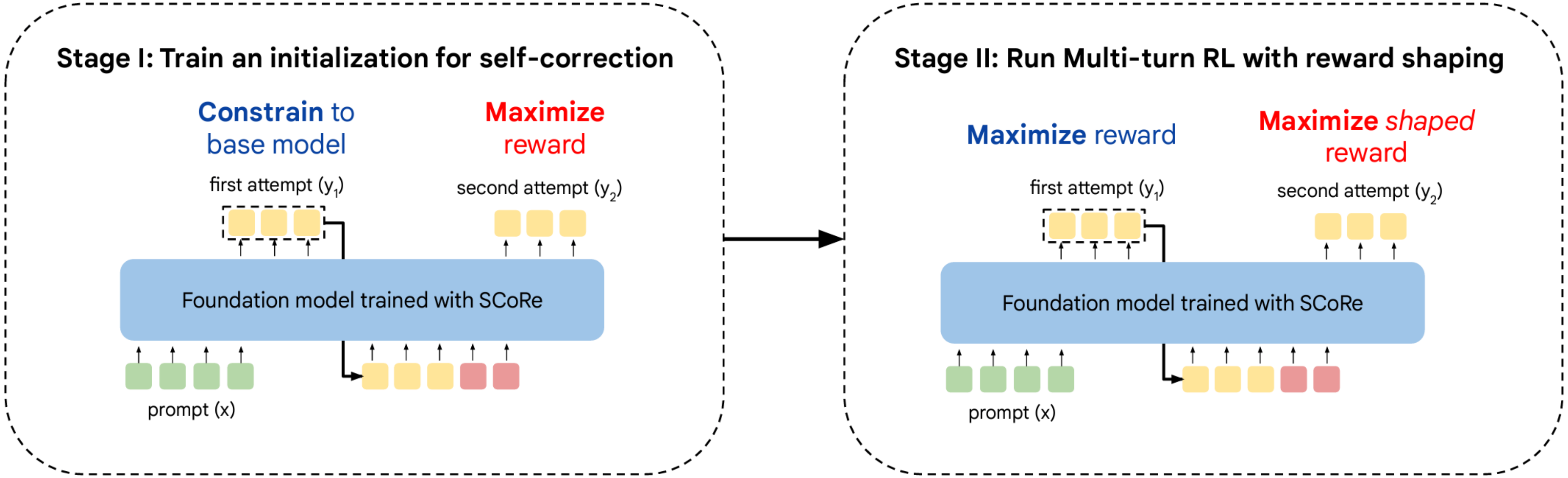
- **Stage 2 (Multi-turn Optimization)** : jointly optimizing both attempts, where the latter uses a ***shaped reward*** to incentivize the discovery of the self-correction strategy instead of the simple strategy of producing the best first response followed by making any minor edits to it in the second attempt

$$\max_{\theta} \; \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{y}_1\sim\pi_\theta(\cdot|\boldsymbol{x}),\boldsymbol{y}_2\sim\pi_\theta(\cdot|[\boldsymbol{x}_1,p_1])}\left[\sum_{i=1}^{2}\widehat{r}(\boldsymbol{y}_i,\boldsymbol{y}^*) - \beta_1 D_{KL}\left(\pi_\theta(\cdot|\boldsymbol{x}_i)||\pi_{\text{ref}}(\cdot|\boldsymbol{x}_i)\right)\right]$$

  - Stage 1 objective: $\max_{\theta} \; \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{y}_1\sim\pi_\theta(\cdot|\boldsymbol{x}),\boldsymbol{y}_2\sim\pi_\theta(\cdot|[\boldsymbol{x}_1,p_1])}\left[\widehat{r}(\boldsymbol{y}_2,\boldsymbol{y}^*) - \beta_2 D_{KL}\left(\pi_\theta(\cdot||\boldsymbol{x}_1)||\pi_{\text{ref}}(\cdot|\boldsymbol{x}_1)\right)\right]$

Suggestions

- **Suggestions**



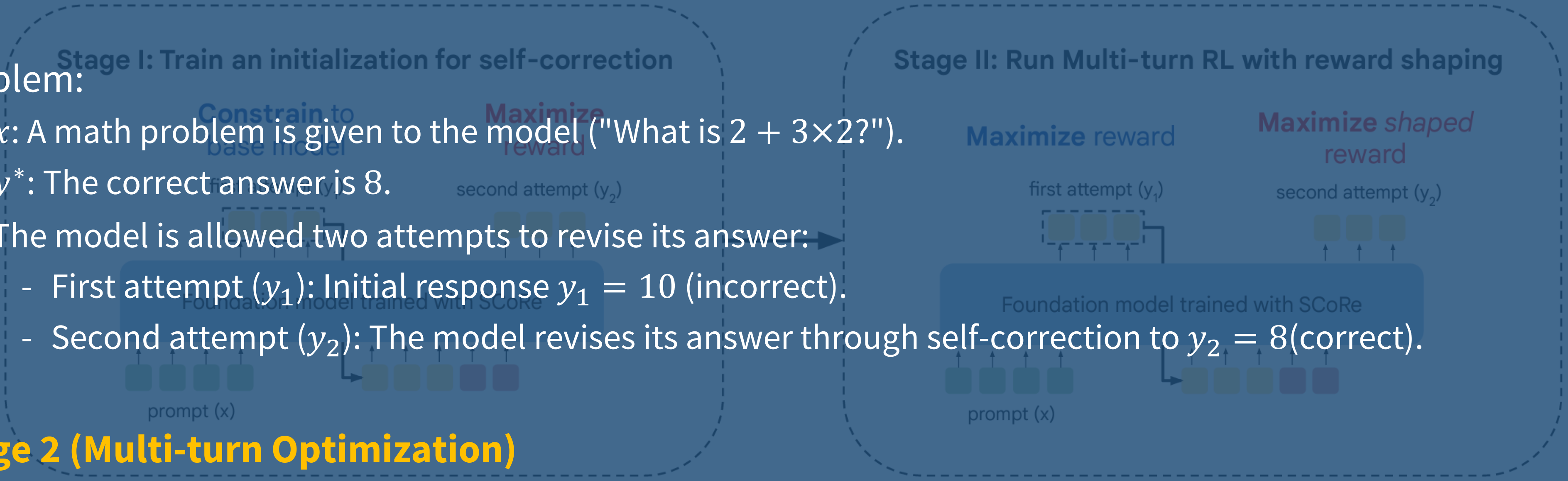
- **Stage 2 (Multi-turn Optimization)** : jointly optimizing both attempts, where the latter uses a ***shaped reward*** to incentivize the discovery of the self-correction strategy instead of the simple strategy of producing the best first response followed by making any minor edits to it in the second attempt

$$\max_{\theta} \ \mathbb{E}_{\boldsymbol{x}_1,\boldsymbol{y}_1\sim\pi_\theta(\cdot|\boldsymbol{x}),\boldsymbol{y}_2\sim\pi_\theta(\cdot|[\boldsymbol{x}_1,p_1])}\left[\sum_{i=1}^{2}\widehat{r}(\boldsymbol{y}_i,\boldsymbol{y}^*) - \beta_1 D_{KL}\left(\pi_\theta(\cdot|\boldsymbol{x}_i)||\pi_{\text{ref}}(\cdot|\boldsymbol{x}_i)\right)\right]$$

  - Reward shaping to prevent behavior collapse: $\widehat{b}(\boldsymbol{y}_2|\boldsymbol{y}_1,\boldsymbol{y}^*) := \alpha\cdot\left(\widehat{r}(\boldsymbol{y}_2,\boldsymbol{y}^*) - \widehat{r}(\boldsymbol{y}_1,\boldsymbol{y}^*)\right)$

Suggestions

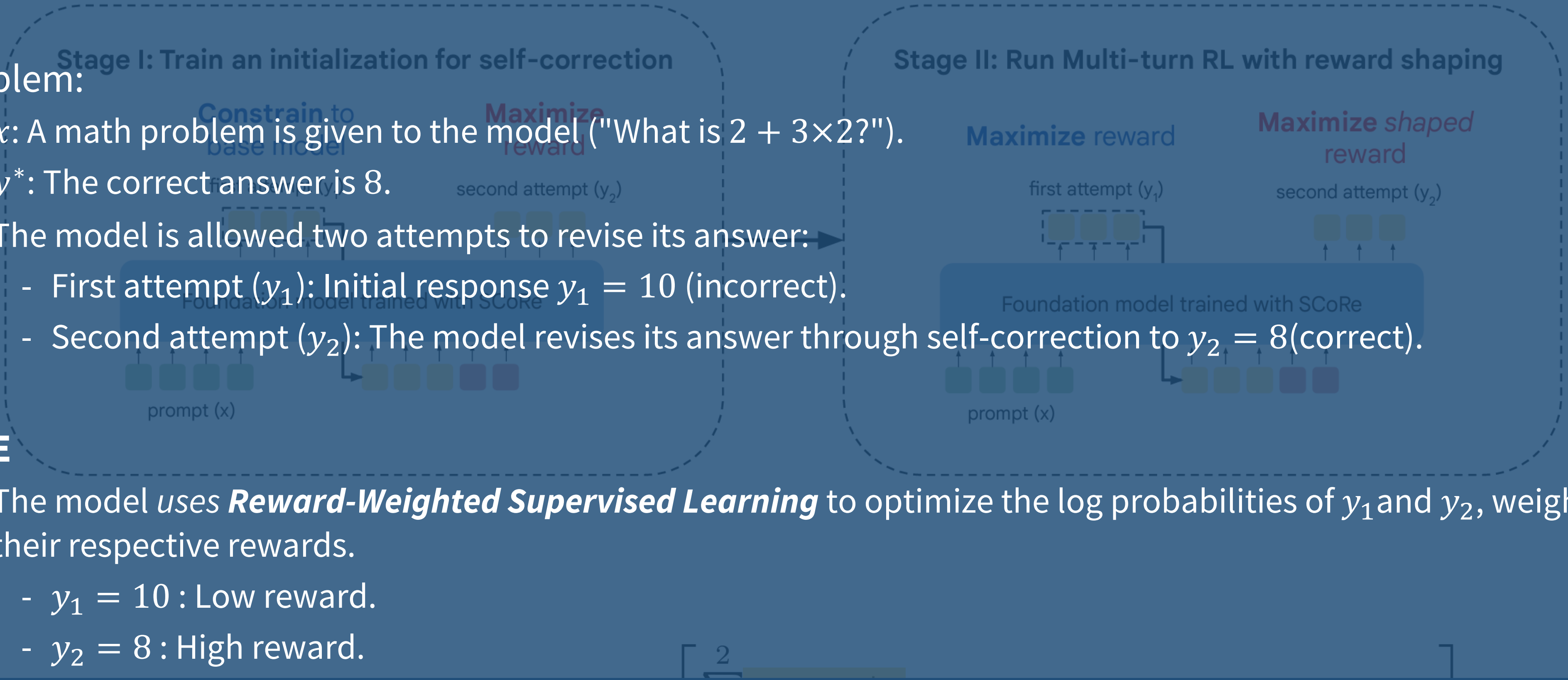
- **Example: Solving a math problem**


Stage I: Train an initialization for self-correction
Constrain to base model
Maximize reward
first attempt (y1)
second attempt (y2)
Foundation model trained with SCoRe
prompt (x)
Stage II: Run Multi-turn RL with reward shaping
Maximize reward
Maximize shaped reward
first attempt (y1)
second attempt (y2)
Foundation model trained with SCoRe
prompt (x)

  - Problem:
    - x: A math problem is given to the model ("What is 2 + 3×2?").
    - y^*: The correct answer is 8.
    - The model is allowed two attempts to revise its answer:
      - First attempt (y_1): Initial response $y_1 = 10$ (incorrect).
      - Second attempt (y_2): The model revises its answer through self-correction to $y_2 = 8$(correct).
  - **Stage 2 (Multi-turn Optimization)**
    - Simultaneously optimizes both attempts:
      - y_1 : The model reduces or eliminates the possibility of generating an incorrect response like 10.
        - The model optimizes the cumulative quality of responses $r(y_1, y^*)$ and $r(y_2, y^*)$
      - y_2: The model corrects the mistake from y_1 and improves y_2 to 8.
    - The model strengthens its self-correction ability while maintaining stability during multi-turn interactions.

$$\hat{b}(\boldsymbol{y}_2|\boldsymbol{y}_1, \boldsymbol{y}^*) := \alpha \cdot (\hat{r}(\boldsymbol{y}_2, \boldsymbol{y}^*) - \hat{r}(\boldsymbol{y}_1, \boldsymbol{y}^*))$$

Suggestions


Stage I: Train an initialization for self-correction
Constrain to base model
Maximize reward
second attempt (y2)
Foundation model trained with SCoRe
prompt (x)
Stage II: Run Multi-turn RL with reward shaping
Maximize reward
Maximize shaped reward
first attempt (y1)
second attempt (y2)
Foundation model trained with SCoRe
prompt (x)
2

- **Example: Solving a math problem**
  - Problem:
    - x: A math problem is given to the model ("What is $2 + 3\times2$?").
    - y^*: The correct answer is 8.
    - The model is allowed two attempts to revise its answer:
      - First attempt (y_1): Initial response $y_1 = 10$ (incorrect).
      - Second attempt (y_2): The model revises its answer through self-correction to $y_2 = 8$(correct).
  - **RISE**
    - The model *uses* ***Reward-Weighted Supervised Learning*** to optimize the log probabilities of $y_1$and y_2, weighted by their respective rewards.
      - $y_1 = 10$: Low reward.
      - $y_2 = 8$: High reward.

Self-correction is indirectly learned,
but the decoupling between the first and second attempts is not explicitly addressed.

Results

- **Performance of SCoRe**
  - Metrics
    - Acc.@t1: accuracy at 1st attempt
    - Acc.@t2: accuracy at 2nd attempt
    - Δ(t1, t2) = Acc.@t2-Acc.@t1
    - Δi→c(t1, t2) : Error Correction
      fraction of incorrect → correct
    - Δc→i(t1, t2) : Stability
      fraction of correct → incorrect

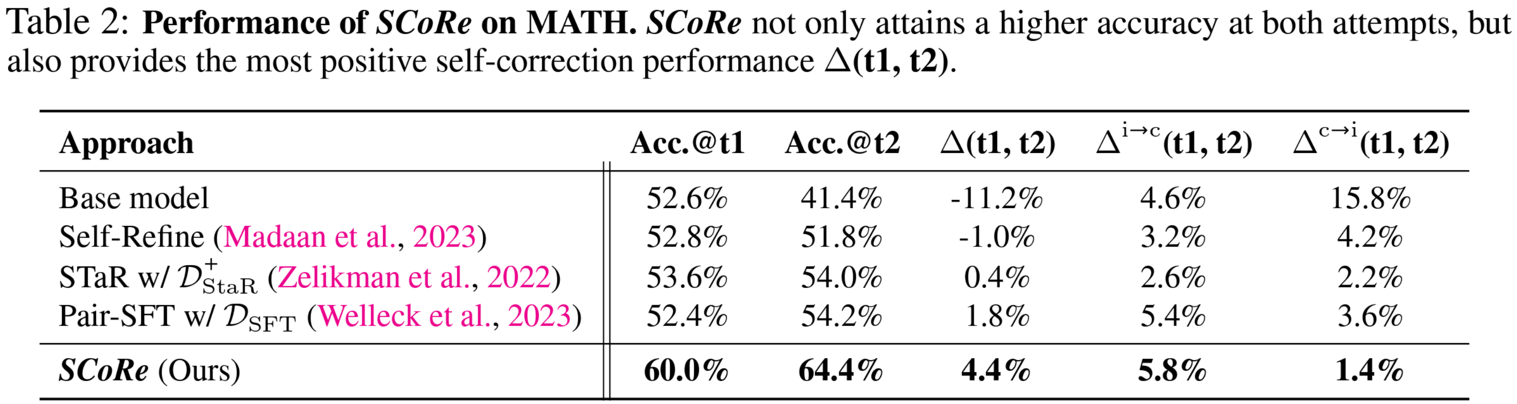


Table 2: **Performance of *SCoRe* on MATH. *SCoRe*** not only attains a higher accuracy at both attempts, but also provides the most positive self-correction performance **Δ(t1, t2)**.

Approach	**Acc.@t1**	**Acc.@t2**	**Δ(t1, t2)**	**$\Delta^{\text{i}\to\text{c}}$(t1, t2)**	**$\Delta^{\text{c}\to\text{i}}$(t1, t2)**
Base model	52.6%	41.4%	-11.2%	4.6%	15.8%
Self-Refine (Madaan et al., 2023)	52.8%	51.8%	-1.0%	3.2%	4.2%
STaR w/ $\mathcal{D}^{+}_{\text{StaR}}$ (Zelikman et al., 2022)	53.6%	54.0%	0.4%	2.6%	2.2%
Pair-SFT w/ $\mathcal{D}_{\text{SFT}}$ (Welleck et al., 2023)	52.4%	54.2%	1.8%	5.4%	3.6%
SCoRe (Ours)	**60.0%**	**64.4%**	**4.4%**	**5.8%**	**1.4%**

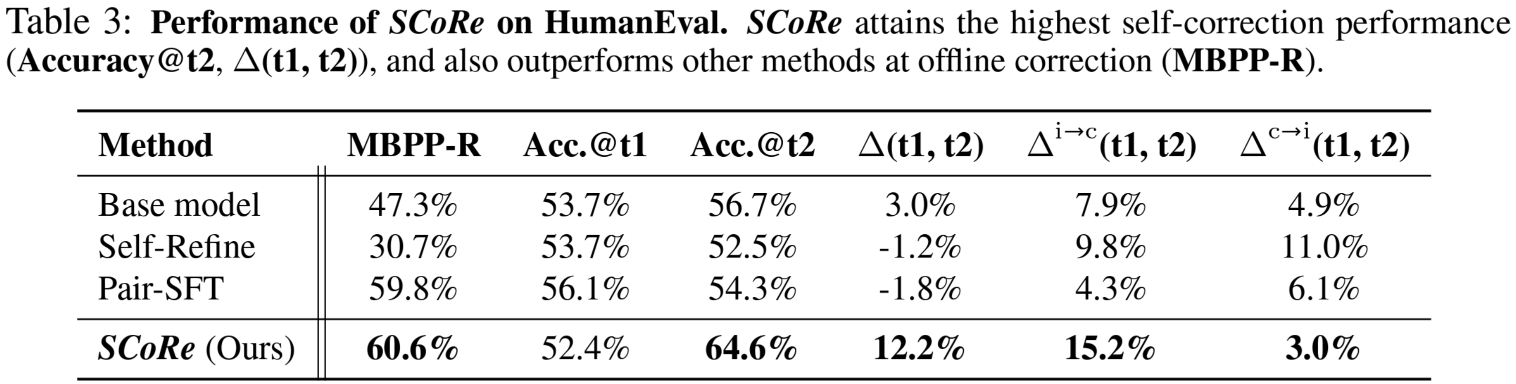


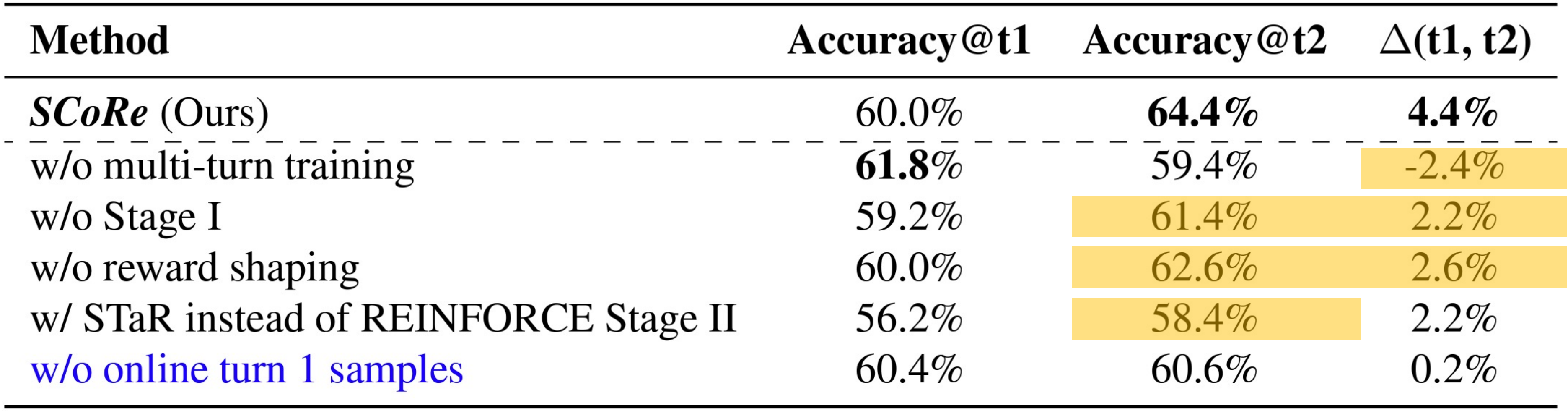
Table 3: **Performance of *SCoRe* on HumanEval. *SCoRe*** attains the highest self-correction performance (**Accuracy@t2**, **Δ(t1, t2)**), and also outperforms other methods at offline correction (**MBPP-R**).

Method	**MBPP-R**	**Acc.@t1**	**Acc.@t2**	**Δ(t1, t2)**	**$\Delta^{\text{i}\to\text{c}}$(t1, t2)**	**$\Delta^{\text{c}\to\text{i}}$(t1, t2)**
Base model	47.3%	53.7%	56.7%	3.0%	7.9%	4.9%
Self-Refine	30.7%	53.7%	52.5%	-1.2%	9.8%	11.0%
Pair-SFT	59.8%	56.1%	54.3%	-1.8%	4.3%	6.1%
SCoRe (Ours)	**60.6%**	52.4%	**64.6%**	**12.2%**	**15.2%**	**3.0%**

SCoRe achieves superior accuracy and self-correction performance

Results

- **Ablation**



Method	Accuracy@t1	Accuracy@t2	Δ(t1, t2)
SCoRe (Ours)	60.0%	**64.4%**	**4.4%**
w/o multi-turn training	**61.8%**	59.4%	-2.4%
w/o Stage I	59.2%	61.4%	2.2%
w/o reward shaping	60.0%	62.6%	2.6%
w/ STaR instead of REINFORCE Stage II	56.2%	58.4%	2.2%
w/o online turn 1 samples	60.4%	60.6%	0.2%

- Multi-Turn Training is Essential
- Stage I is Critical
- Reward Shaping Significantly Affects Performance
- On-Policy RL (REINFORCE) Outperforms STaR

SCoRe’s components are carefully designed to optimize self-correction.

Conclusion

# Main Findings

Summary

1. Goal

- Trains language models to improve their responses through multi-turn self-correction.

2. 3 Suggestions

1. 2-stage Framework

- Stage 1: Decouples the first and second attempts to avoid behavior collapse, focusing on initializing the model for effective correction.
- Stage 2: Optimizes both attempts using reinforcement learning (REINFORCE) with reward shaping.

2. Reward Shaping

- Guides the model to prioritize improvement between attempts and penalizes regressions.

3. Key Metrics

- Demonstrates improvements in second-attempt accuracy (Acc.@t2), self-correction ability (Δi→c), and stability (Δc→i).

3. Effects

- Outperforms baseline methods on benchmarks (e.g., MATH, HumanEval).
- Effectively corrects errors while minimizing regressions during self-correction.

Thank You

Yejin Yoon

HYU NLP Lab.
Hanyang University, South Korea

stillwithyou@hanyang.ac.kr