

Paper Review : 2025 Spring Lab Seminar

Self-Correction & Self-Verification

: A Comparative Review of SCoRe, ReVISE and S²R

Kumar, Zhuang, Agarwal et al. (Google DeepMind | ICLR2025)

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

Lee, Oh et al. (KAIST, Yonsei Univ. | Reasoning and Planning for LLMs @ICLR2025)

“ReVISE: Learning to Refine at Test-Time via Intrinsic Self-Verification”

Ma, Wang et al. (Tencent, Tsinghua Univ., ... | arXiv 2025.02)

“S²R: Teaching LLMs to Self-verify and Self-correct via Reinforcement Learning”

Yejin Yoon

Self-Correction & Self-Verification

Contents

1 Introduction

- Motivation
- Main contributions of each paper

2 Comparison of Methodologies

- SCoRe, ReVISE and S²R

3 Experimental Results Comparison

- Performance analysis

4 Discussion and Conclusion

- Conclusion
- Future research directions

Introduction

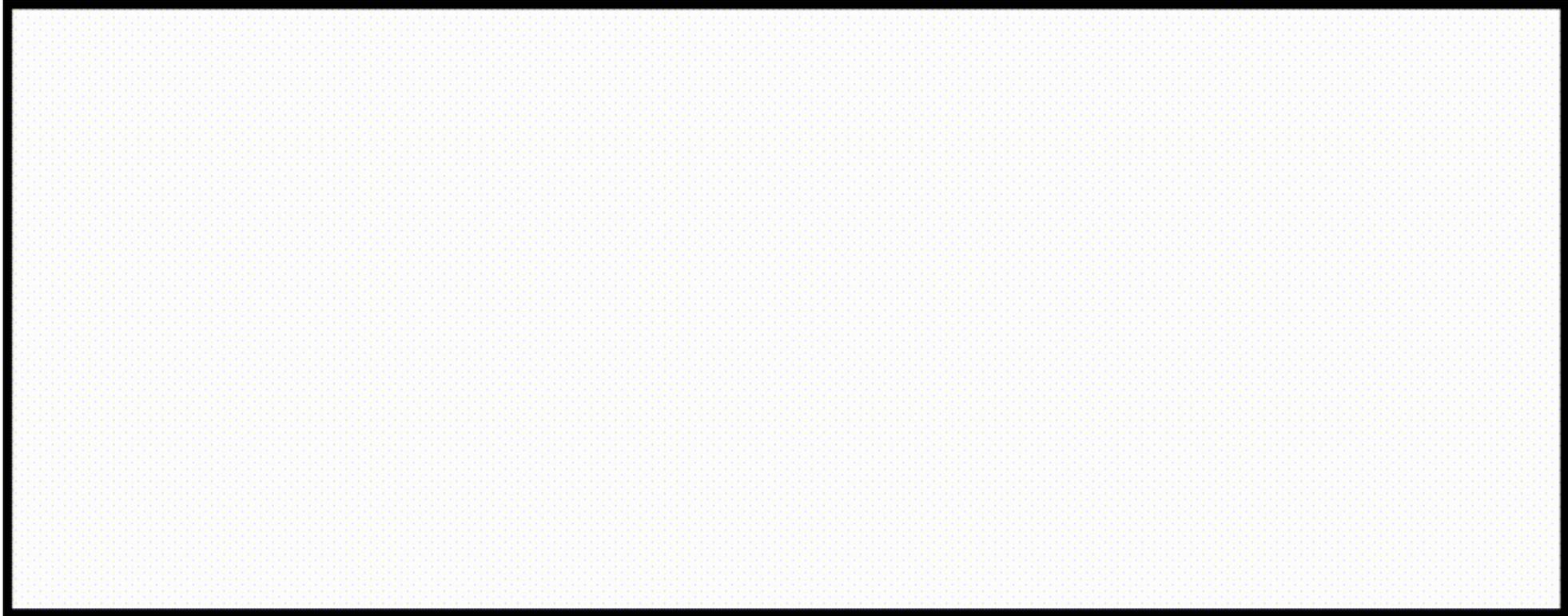
Self-Correction

Motivation

Main contributions of each paper

Self-Correction

📄 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

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

03

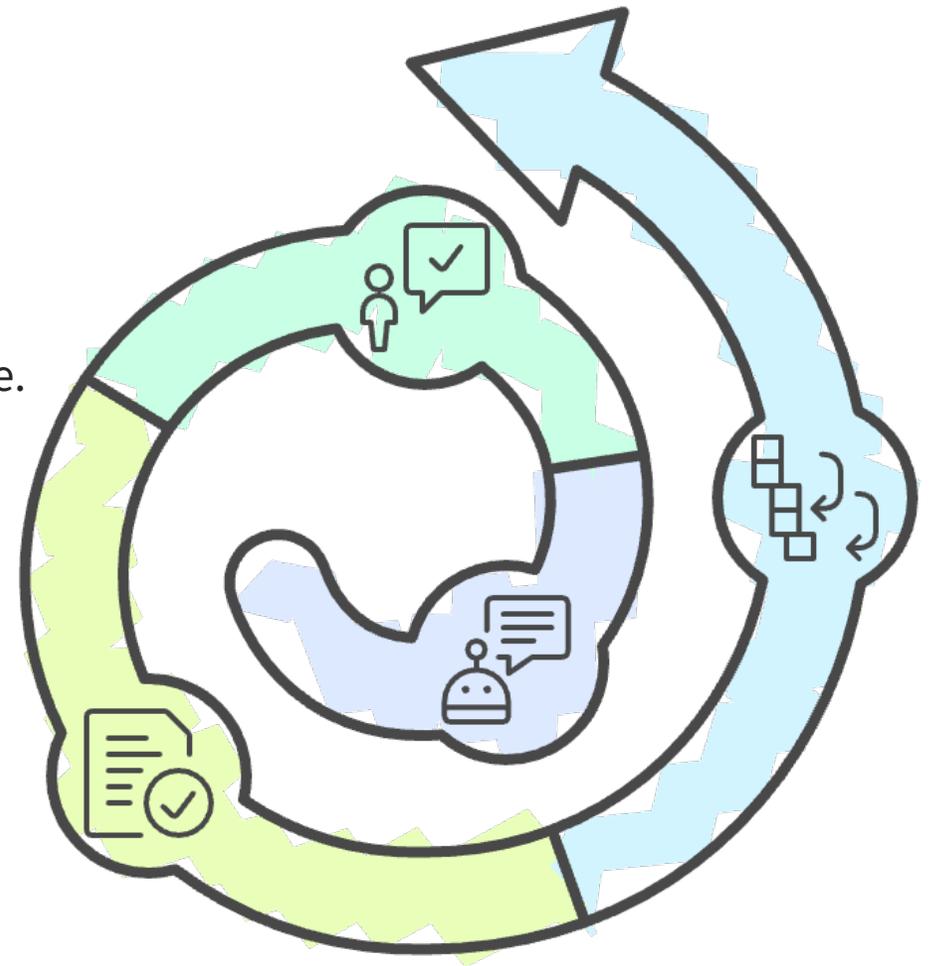
Output Correction

02

Feedback Collection

04

Iteration Continuation

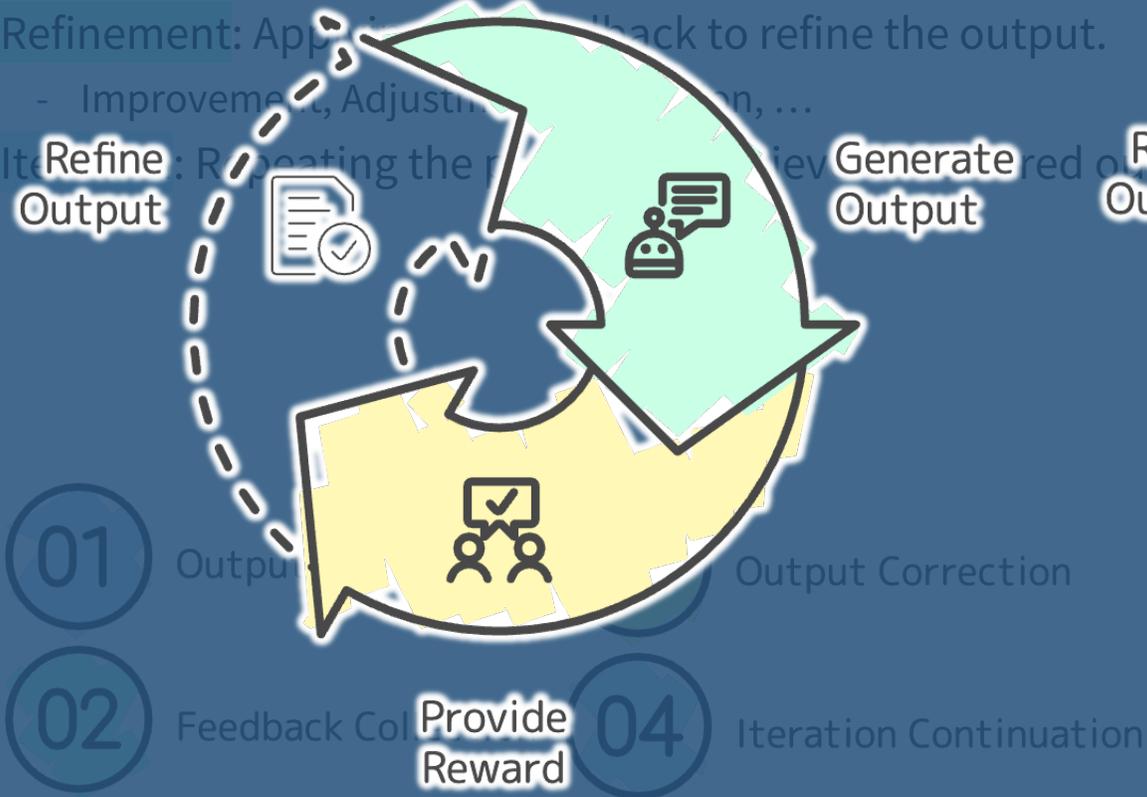


Iterative Correction Process

Implicit vs. Explicit Correction

3 (or 4) Key Components

- Output: **Implicit Correction**
- Feedback: Identifying areas for improvement in the output.
- Refinement: Applying feedback to refine the output.
 - Improvement, Adjustment, ...
- Iteration: Repeating the process to refine the output.



Explicit Correction



Iterative Correction Process

Why is Self-Correction Important?

• Limitations of Previous Approaches

MATH Example 1: Arithmetic Self-Correction

Problem. If $n \equiv 2 \pmod{7}$, then find the remainder when $(n + 2)(n + 4)(n + 6)$ is divided by 7.

SCoRe turn 1 solution (incorrect). Since $n \equiv 2 \pmod{7}$, we can write $n = 7k + 2$ for some integer k . Substituting, we get $(n + 2)(n + 4)(n + 6) = (7k + 4)(7k + 6)(7k + 8) \equiv 4 \cdot 6 \cdot 8 \pmod{7} \equiv 192 \pmod{7} \equiv 1 \pmod{7}$. Final Answer: The final answer is 1. I hope it is correct.

Self-correction instruction. There might be an error in the solution above because of lack of understanding of the question. Please correct the error, if any, and rewrite the solution.

SCoRe turn 2 solution (correct). Since $n \equiv 2 \pmod{7}$, we can write $n = 7k + 2$ for some integer k . Substituting, we get $(n + 2)(n + 4)(n + 6) = (7k + 4)(7k + 6)(7k + 8) \equiv 4 \cdot 6 \cdot 8 \pmod{7} \equiv 4 \cdot 6 \cdot 1 \pmod{7} \equiv 24 \pmod{7} \equiv 3 \pmod{7}$. Final Answer: The final answer is 3. I hope it is correct.

- LLMs struggle to correct their own reasoning errors.
 - Zero-shot & Few-shot prompting \rightarrow Limited Self-Correction ability.
- Existing methods (Fine-Tuning, Prompting) have limitations.
 - Supervised Fine-Tuning (SFT) \rightarrow Distribution mismatch & behavior collapse.

Research Questions in These Papers

- **RQ1) Can LLMs autonomously perform Self-Verification & Self-Correction?**
- **RQ2) Is Supervised Fine-Tuning sufficient?**
- **RQ3) Does Reinforcement Learning improve Self-Correction?**
- **TL;DRs of each paper**
 - **SCoRe**: Employs multi-turn RL to enhance self-correction
 - **ReVISE**: Uses self-verification to enable self-correction w/o RL
 - **S²R**: Integrates self-verification and self-correction and optimizes w/RL

Similarities

- All 3 papers focus on self-correction techniques for LLMs to improve their reasoning abilities.
- **The main commonalities include:**
 1. **Need for self-correction** – 3 works agree that LLMs detect and fix reasoning errors.
 2. **Test-time scaling** – additional computation at inference improves reasoning accuracy
 3. **Beyond SFT → use additional mechanisms** – SFT alone leads to distribution mismatch & behavior collapse, making additional methods necessary (e.g., RL, preference learning, ...)
 4. **Evaluation on Math & Code Tasks** – experiments on datasets like MATH, GSM8K, and Human Eval to assess reasoning performance
 5. **Iterative Reasoning** – rather than a single-pass answer, models should re-evaluate and refine their reasoning over multiple steps

#1 Need for Self-Correction

💡 Why it matter?

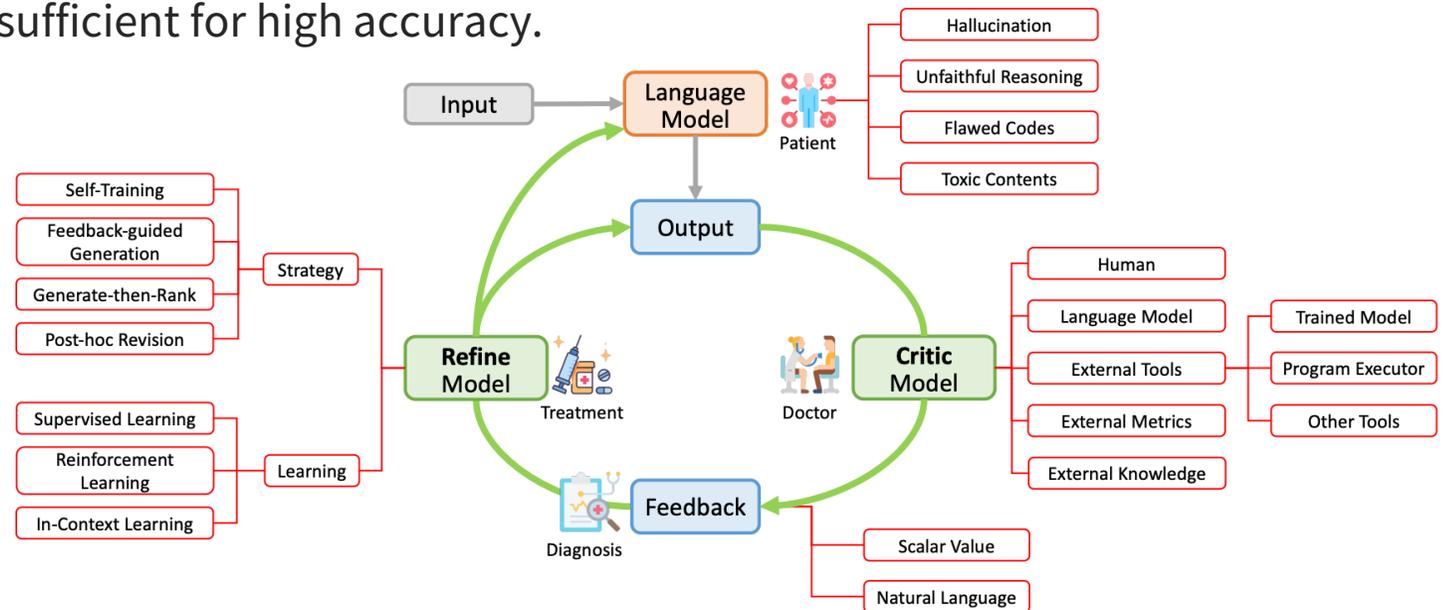
Reasoning errors tend to propagate in **auto-regressive models**, leading to poor final outputs. A self-correction mechanism allows LLMs to refine their reasoning and improve answer accuracy.

• LLMs make reasoning errors!

- Simply generating a single output is insufficient for high accuracy.

• Self-correction mechanisms should be integrated into LLMs

- It enables models to identify and fix their own mistakes rather than relying solely on *external feedback* or *human supervision*.



#2 Test-Time Scaling

What is Test-Time Scaling?

(definition) the practice of dynamically increasing computational resources or processing steps **at inference time** to improve the accuracy and reliability of model outputs.

Instead of relying solely on a model's first generated response, it leverages additional **verification, iteration, or selection strategies** to refine outputs, making reasoning more robust and error-resistant.

- **Additional computation at inference improves reasoning accuracy.**

- Instead of improving models only through pre-training & SFT, recent works highlight test-time computation as a viable approach to enhance reasoning.

e.g., best-of-N, Self-Consistency Decoding, Self-refinement, external verification (reward model, verifier,), ...

How many r in raspberry?

Question

Let's break down the process of counting the letter 'r' in the word "raspberry" ...

* First letter: 'r' - This is an 'r', count = 1.

* Second letter: 'a' - Not an 'r', count remains 1 ...

* Sixth letter: 'e' - Not an 'r', count remains 1.

* Seventh letter: 'r' - This is an 'r', count = 2.

* Eighth letter: 'y' - Not an 'r', count remains 2 ...

The number of 'r's in "raspberry" is 2.

Wait, let's re-read the question carefully. It asks "How many r in raspberry?" ... * r - a - s - p - b - e - r - r - y ... * First 'r' ... *

Second 'r' ... * Third 'r' ... Count = 3 ... **Reasoning trace**

My initial answer of 2 was incorrect due to a quick reading of the word. **Final Answer:** The final answer is **3** **Response**

#2 Test-Time Scaling

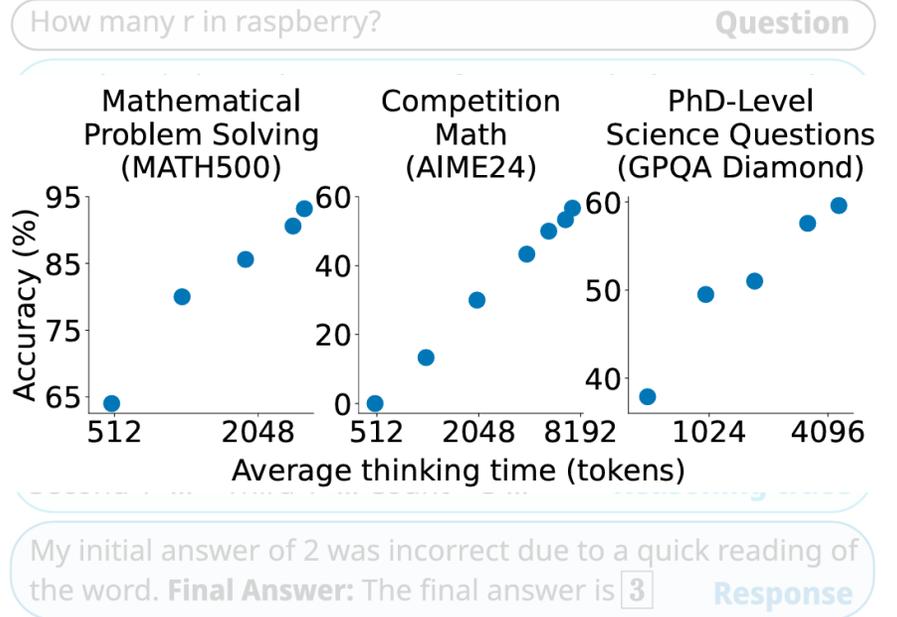
💡 *What is Test-Time Scaling?*

(definition) a practice of dynamically increasing computational resources or processing steps **at inference time** to improve the accuracy and reliability of model outputs.

Instead of relying solely on a model's first generated response, it leverages additional **verification, iteration, or selection strategies** to refine outputs, making reasoning more robust and error-resistant.

- **Additional computation at inference improves reasoning accuracy.**

- Instead of improving models only through pre-training & SFT, recent works highlight test-time computation as a viable approach to enhance reasoning.
- LLMs dynamically adjust their reasoning steps during inference, **refining answers iteratively** rather than relying solely on the first-generation attempt.



#3 Shortcomings of SFT → Using Additional Mechanisms

- ✓ Supervised learning alone struggles to generalize self-correction beyond the dataset.
- ✓ This is why additional mechanisms like RL, preference learning, or intrinsic verification are needed.

- **Supervised fine-tuning (SFT) alone is insufficient for teaching self-correction.**
 - **Distribution Mismatch:** When models are trained on human-annotated corrections, they may learn to correct errors made by the base model, but not necessarily errors they generate themselves.
 - **Behavior Collapse:** Models may overfit to the correction patterns seen in the training data, leading to superficial corrections or even reverting correct answers to incorrect ones.
- **Recent Works move beyond standard supervised learning.**
 - They all emphasize iterative correction rather than single-step fixes, allowing models to learn more complex reasoning pathways.

#4 Evaluation on Mathematical & Coding Tasks for Reasoning

💡 *How These Tasks Help Evaluate Self-Correction?*

Step-by-Step Reasoning is Required – errors in early steps lead to incorrect final answers, meaning models need self-correction to refine solutions. (Error Propagation is Common)

Clear Correctness Criteria - math and code have clear right/wrong answers → ideal for reinforcement learning (RL) and reward shaping

- **Recent researches on self-correction and test-time scaling use mathematical and coding benchmarks to evaluate LLMs' reasoning abilities.**

- these tasks require logical consistency, multi-step inference, and error detection, making them ideal for testing self-verification and self-correction mechanisms.

- **Reasoning Benchmarks**

- (Mathematical) MATH (Hendrycks et al., 2021) – a benchmark covering Olympiad-level math problems.
- (Mathematical) GSM8K (Cobbe et al., 2021) – ~8,500 grade-school math problems requiring reasoning
- (Coding) HumanEval (Chen et al., 2021) – a dataset with programming tasks requiring function synthesis

#5 Importance of Iterative Reasoning

- Reasoning should be treated as an iterative process, rather than a single-pass generation.

 **SCoRe**) Kumar, Zhuang, Agarwal et al. (Google DeepMind) “SCoRe : Training Language Models to Self-Correct via Reinforcement Learning” (ICLR2025)

- Applies multi-turn RL, forcing the model to attempt correction multiple times.

 **ReVISE**) Lee, Oh et al. (KAIST, Yonsei Univ.) “ReVISE: Learning to Refine at Test-Time via Intrinsic Self-Verification” (Reasoning and Planning for LLMs @ ICLR2025)

- Uses intrinsic self-verification to decide whether to stop or refine its response.

 **S²R**) Ma, Wang et al. (Tencent, Tsinghua Univ., ...) “Teaching LLMs to Self-verify and Self-correct via Reinforcement Learning”

- Separates self-verification and self-correction, explicitly teaching both through RL.

Comparison of Methodologies

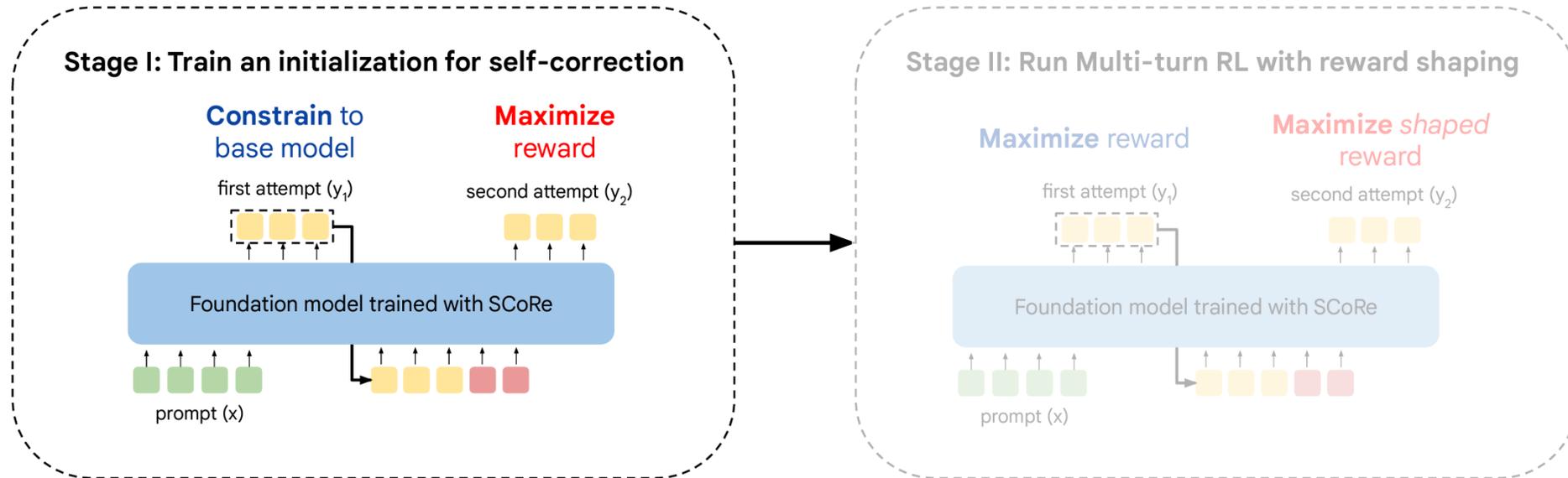
Overview

SCoRe / ReVISE / S²R

Comparison of Approaches

SCoRe

• Suggestions



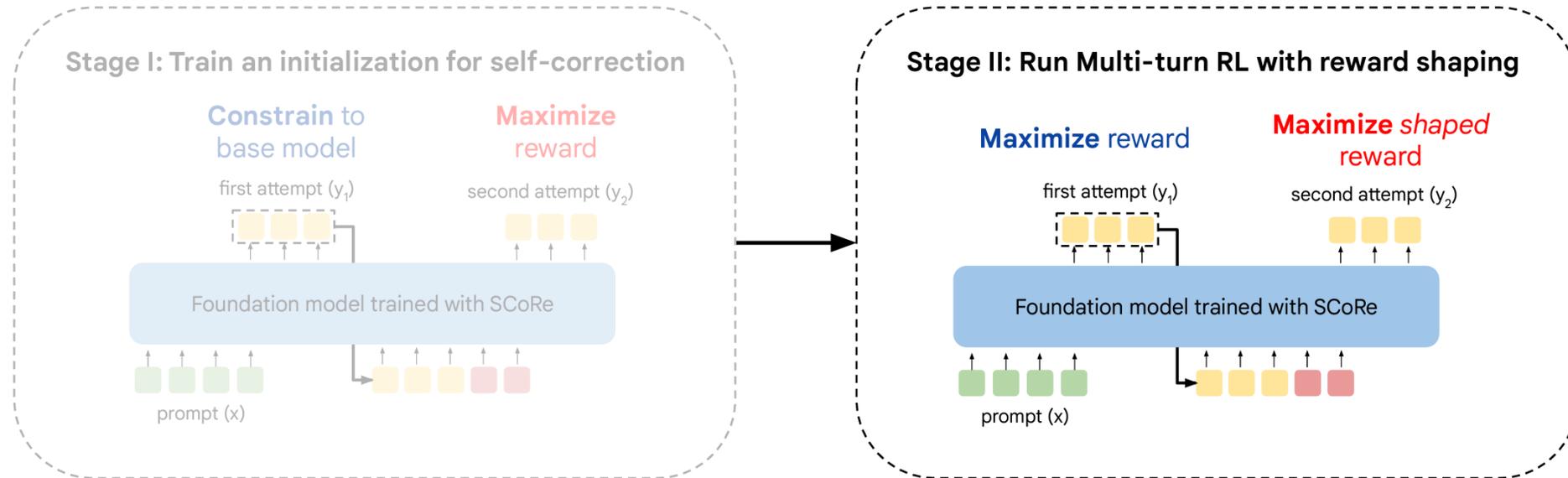
- **Stage 1 (Initialization)** : instead of running SFT (which produces pathological amplification of biases) to initialize RL training, train a good initialization that can produce high-reward responses in the second-attempt while mimicking the base model's response at the first attempt.

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

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

SCoRe

• Suggestions



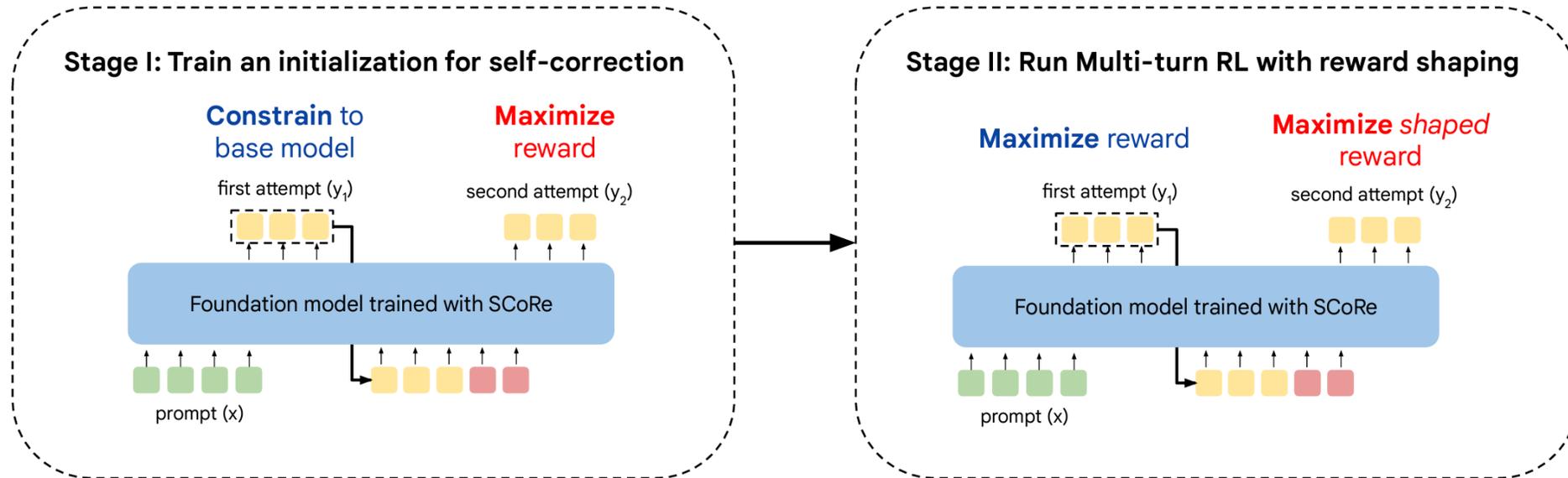
- **Stage 2 (Multi-turn Optimization)** : jointly optimizing both attempts, where the latter uses a **shaped reward** to incentivize the discovery of the self-correction strategy instead of the simple strategy of producing the best first response followed by making any minor edits to it in the second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

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

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

SCoRe

• 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

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

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

Results

Qu et al. (CMU et al.) “Recursive IntroSpEction: Teaching Language Model Agents How to Self-Improve” (NeurIPS2024)

• Suggestions

- Step 1: Data Collection for Self-improvement

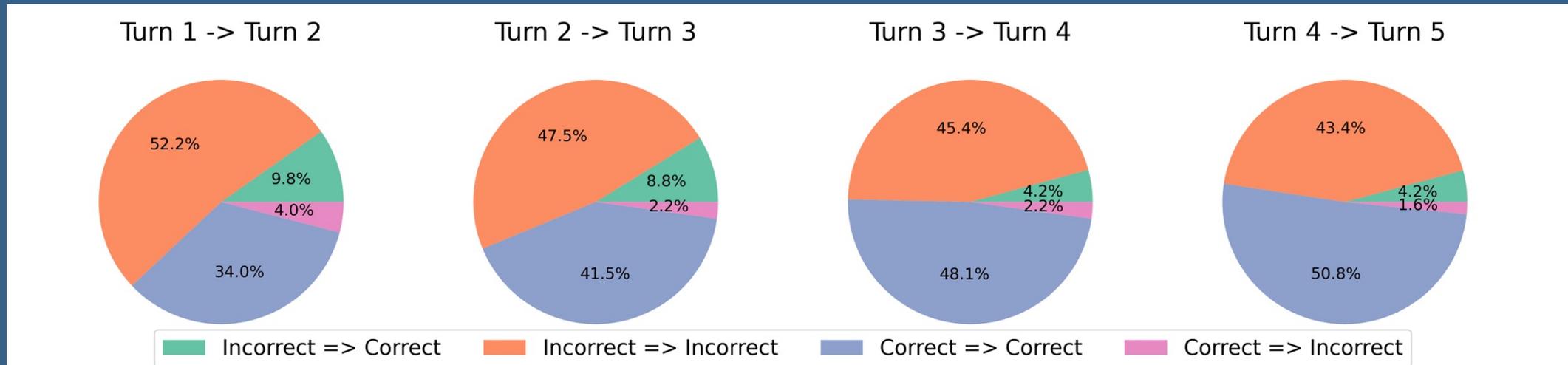


Figure 7: *Change in the fraction of responses that transition their correctness values over the course of multi-turn rollouts from RISE, w/o oracle.* Observe that in general, the fraction of Correct \rightarrow Correct responses increases; Incorrect \rightarrow Incorrect responses decreases; and the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

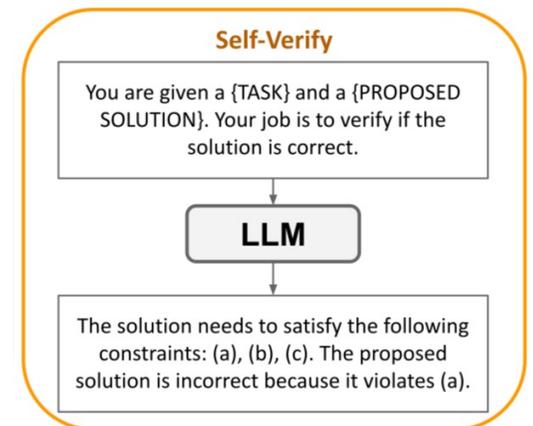
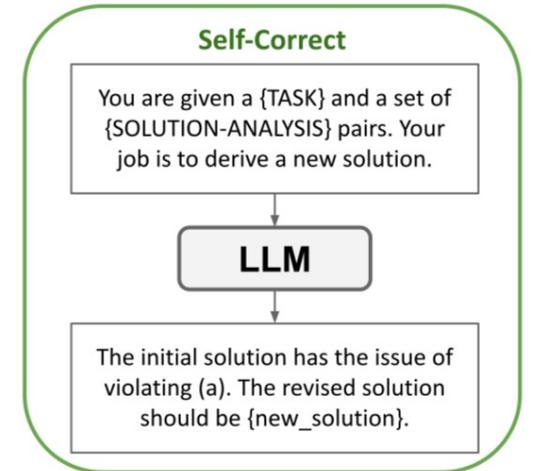
Self-Correction and Self-Verification

• Self-Correction

- The ability to revise its own responses after **recognizing an error**
- Goal: Improve correctness of an answer by iterating on its previous output
- How it works?
 1. Generate an initial response
 2. Predict whether the response needs revision or not)
 3. If so, modify its response to improve correctness

• Self-Verification

- The ability to evaluate whether its own response is correct w/o external supervision
- Goal: Assess its own reasoning accuracy before making a correction
- How it works?
 1. Predict if its answer is logically consistent with its own knowledge
 2. Assign confidence score or flags potential errors
 3. If it is detected, trigger self-correction



Overview

• High-level Comparison

- **SCoRe: Correction without explicit verification** (relies on trial-and-error via RL).
- **ReVISE: Verification-driven correction** (checks correctness before modifying responses).
- **S2R: Integrates both verification and correction**, improving correction reliability.

Aspect	SCoRe	ReVISE	S ² R
Self-Verification	✗	✓ intrinsic verification tokens	✓ explicit verification process
Self-Correction	✓ Multi-turn RL	✓ Confidence-aware decoding	✓ SFT + RL
Traning Approach	2-stage RL	2-stage Curriculum learning	2-stage RL
Reinforcement Learning	✓	✗	✓
Reward Sharping	✓	✗	✓
Test-time Scaling Strategy	Multi-turn Correction	Confidence-aware decoding	Adaptive correction (via self-verification)
Computational Efficiency	✗ High (RL+multi-turn)	✓ Low (no RL)	🙌 High (offline RL)

SCoRe

- **TL; DR**

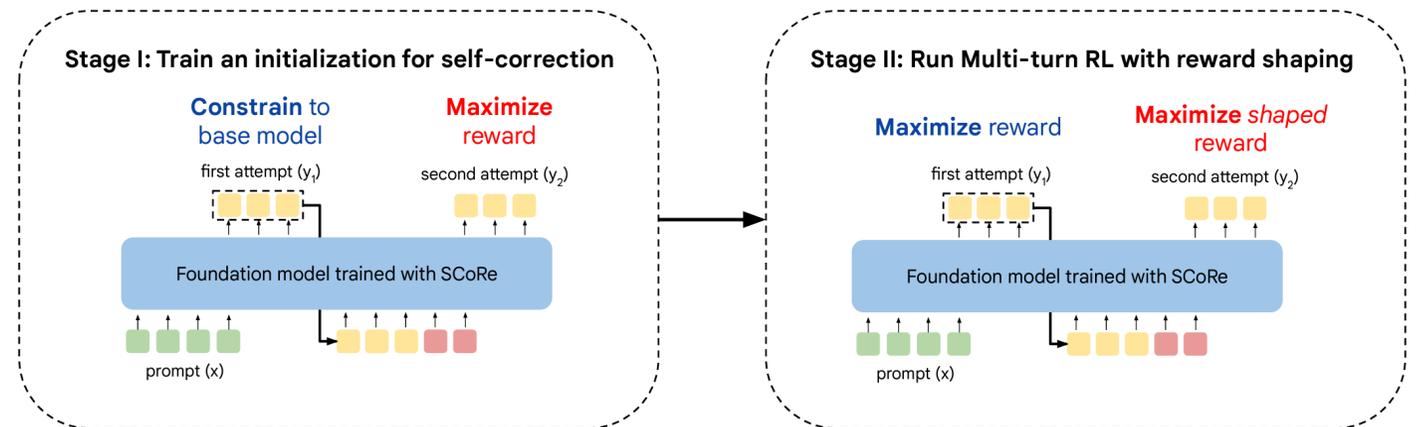
- LLMs should be trained using **multi-turn RL** to self-correct their own mistakes iteratively, instead of relying solely on SFT.

- **Problem States**

- SFT is insufficient for self-correction due to distribution mismatch + behavior collapse

- **Suggestion: 2-stage RL training**

- **Stage 1:** Ensures that second-attempt responses are more accurate while keeping the first attempt close to the base model.
- **Stage 2:** Optimizes self-correction via **reward shaping**, rewarding models for progress made in refining incorrect answers.



ReVISE

- **TL; DR**

- LLMs should be able to self-verify their reasoning and refine their outputs during ***inference*** w/o relying on external verifiers or RL.

- **Problem States**

- LLMs struggle with **systematic reasoning errors**, where mistakes in early reasoning steps accumulate and degrade output accuracy.
- Current correction methods either depend on **expensive external verifiers** or **unstable reinforcement learning**.

- **Suggestion: Introduce Intrinsic Self-Verification**

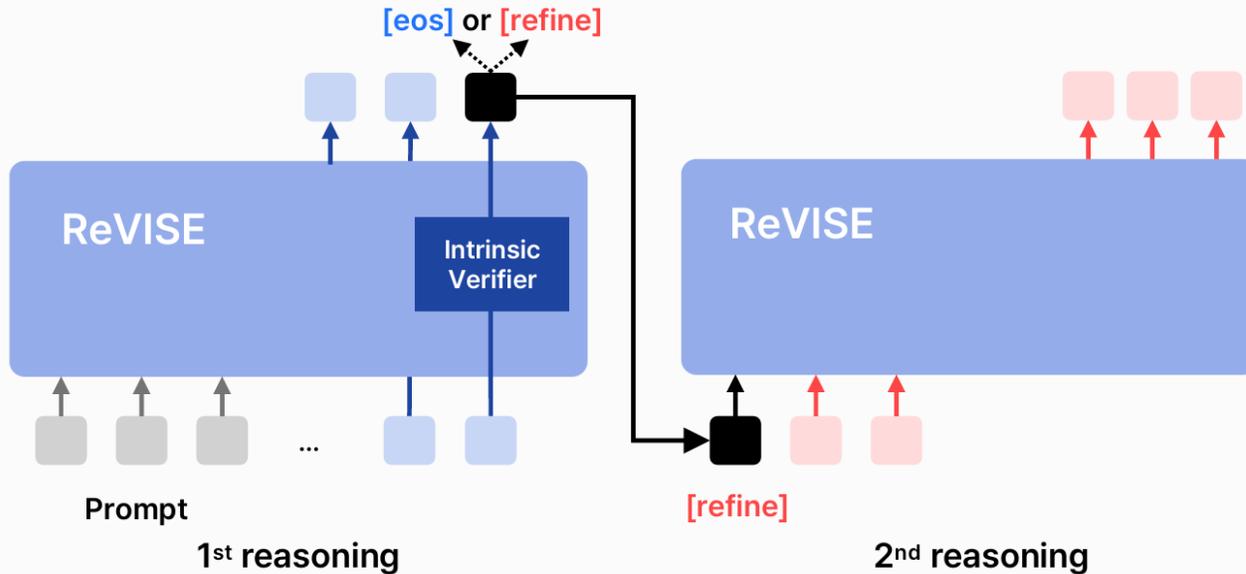
- LLMs should **internally verify** their reasoning steps and correct errors based on ***their confidence*** in the correctness of generated outputs.
- Introduce ***self-verification tokens*** that allow the model to determine whether to stop or refine its reasoning

ReVISE

• Suggestion: Introduce Intrinsic Self-Verification

Method Generate, Verify, and Refine

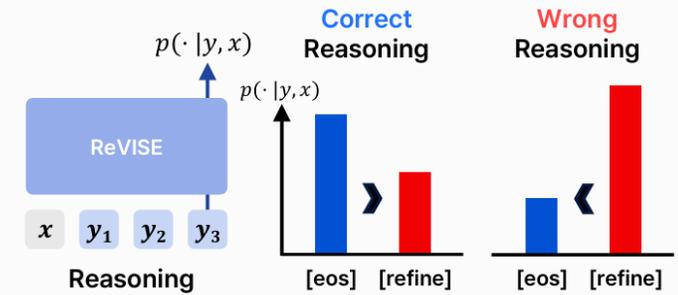
1. ReVISE decides to **stop** or **refine**
2. Refine following **[refine]** token



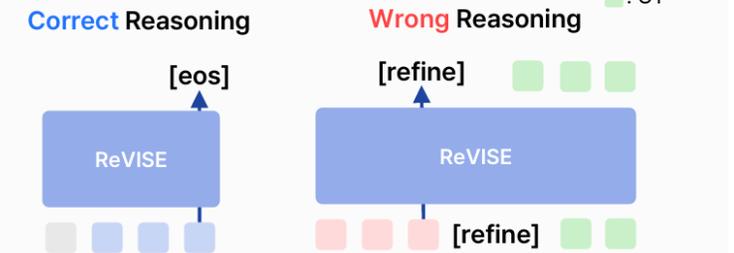
ReVISE is a self-verifying and self-correcting framework

Train Two-stage curricula

Stage 1. Learn how to self-verify



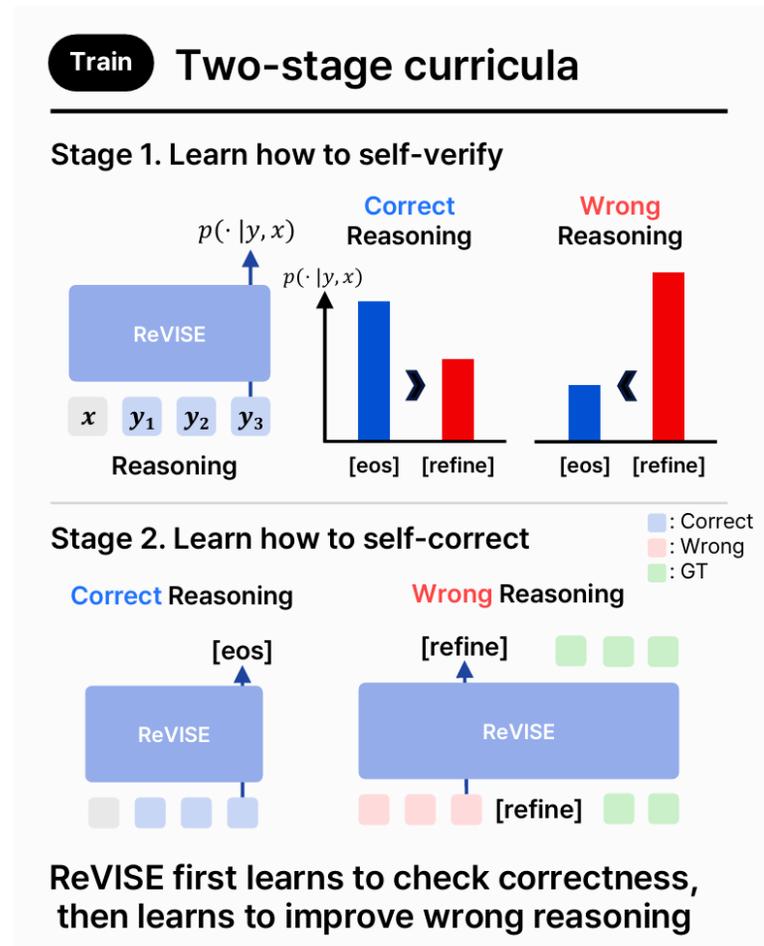
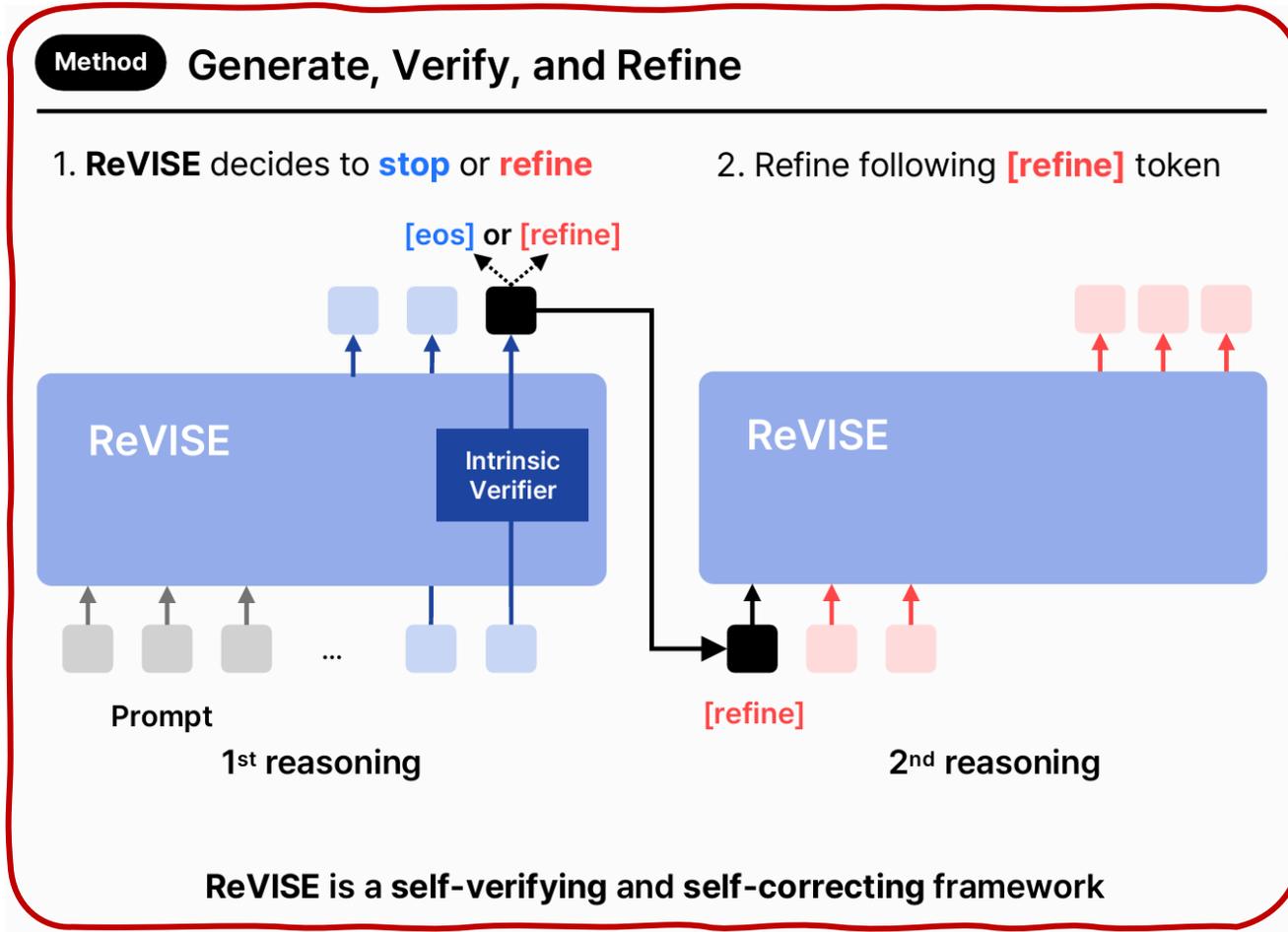
Stage 2. Learn how to self-correct



ReVISE first learns to check correctness, then learns to improve wrong reasoning

ReVISE

• Suggestion: Introduce Intrinsic Self-Verification



ReVISE

• Suggestion (1): [refine] token

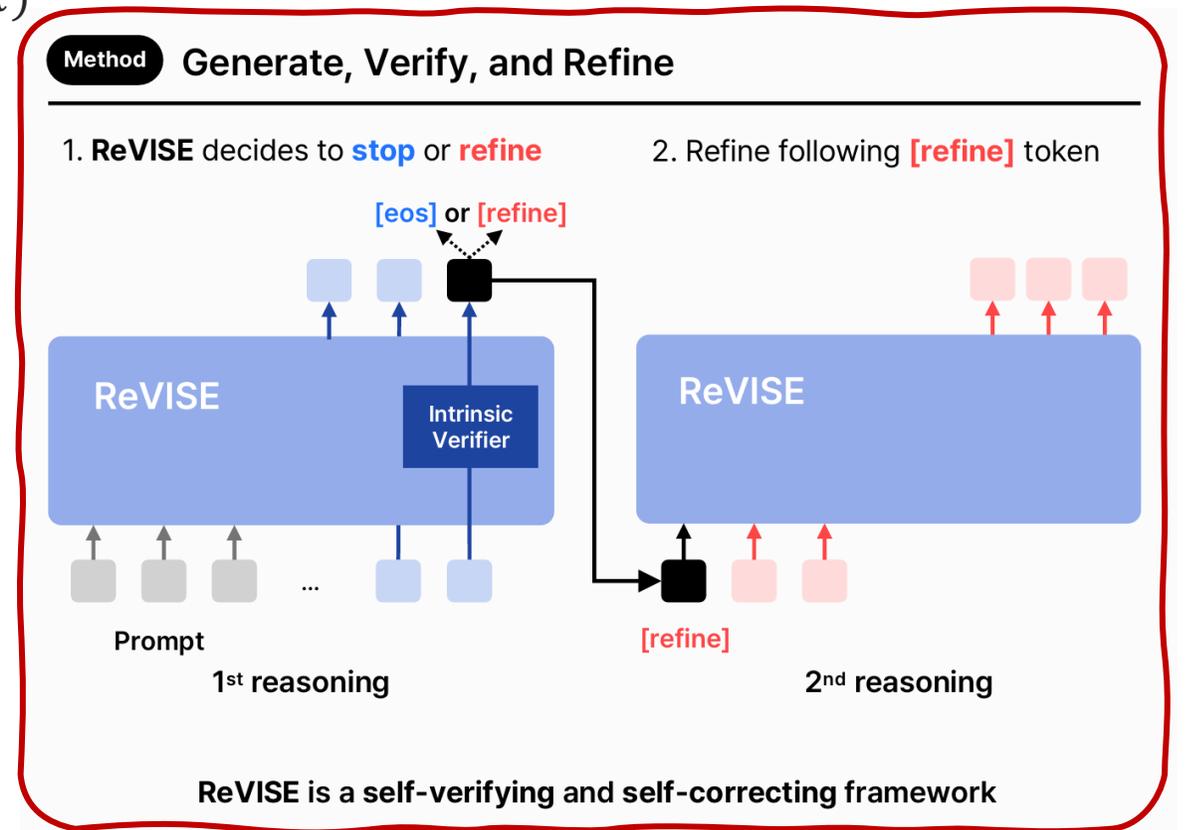
- LLMs typically generate outputs in an **autoregressive** manner
= predict next token sequentially w/o reconsidering past mistakes.
- **Definition:** $P([refine] | y, x) = 1 - P([eos] | y, x)$

- Goal

1. Decide whether to stop or refine its output
2. Re-evaluate its own correctness
3. Dynamically adjust reasoning trajectories

- How it works?

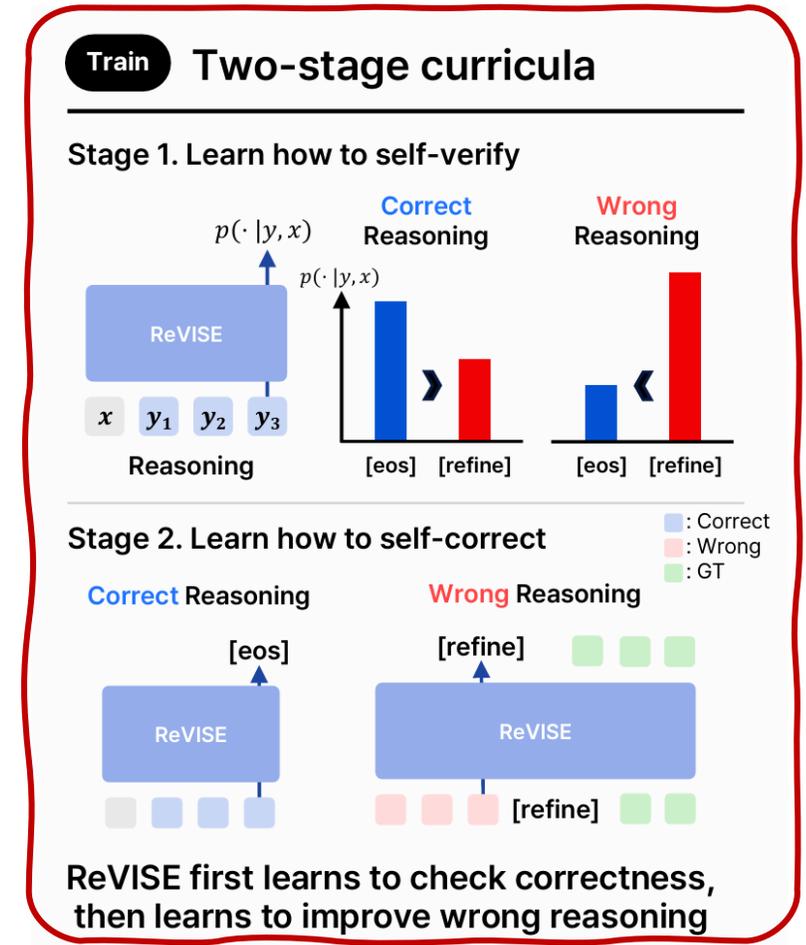
- Solve the problem $y_{init} \sim M(\cdot | x)$ with [eos] token
- Self-verification
 - Calculate confidence score $c = P([eos] | y, x)$
 $P([eos] | y_{init}, x)$ vs. $P([refine] | y_{init}, x)$
 - Set criteria (threshold)
- Self-correction
 - If $P([refine] | y_{init}, x)$ is **high**,
the model **decides to refine the answer.**



ReVISE

• Suggestion (2): Curriculum Learning

- Stage 1: Self-Verification *verify whether an answer is right or wrong*
 - trained on **pairs of correct and incorrect reasoning outputs.**
 - Positive samples \rightarrow [eos]
 - Negative samples \rightarrow [refine]
 - **Datasets:**
 - Generate multiple responses
 - Classify the responses as correct or not
 - Preference learning
 - **Objective:** $\mathcal{L}_{\text{verify}} = \mathcal{L}_{\text{SFT}}(D_{\text{verify}}) + \lambda \mathcal{L}_{\text{Pref}}(D_{\text{verify}})$
- Stage 2: Self-Correction *correct wrong answers using preference learning*
 - **Learn correct mistakes** when it outputs [refine]
 - **Objective:** $\mathcal{L}_{\text{correct}} = \mathcal{L}_{\text{SFT}}(D_{\text{correct}}) + \lambda \mathcal{L}_{\text{Pref}}(D_{\text{correct}})$



ReVISE

• Suggestion (3): Confidence-aware Sampling

- **Goal:** Instead of selecting the most frequent answer, **each answer's confidence score is considered.**
 - model **prefers highly confident responses**, even if they appear less frequently.

- How it works?

1. Generate N candidate answers
2. Compute the self-verification confidence c for each answer ($c = P([eos] | y, x)$)
3. Aggregate scores using confidence-weighted voting: $y^* = \arg \max \sum_{i: y_i \in y} c_i$

e.g.

y_i	Majority voting	Confidence score c_i	Aggregated score
correct	3	0.9 0.85 0.95	2.7
incorrect	4	0.4 0.35 0.45 0.5	1.7

- **TL; DR**

- LLMs learn both **self-verification** and **self-correction** through a 2-stage **SFT and RL training** framework.

- **Problem States**

- Existing methods often treat **self-correction independently**, without explicitly modeling **self-verification**.
- RL-based correction methods suffer from **high resource costs** and instability.

- **Problem Setup and Formulation**

- A reasoning trajectory consists of alternating solve and verify actions: $y = \{s_1, v_1, s_2, v_2, \dots, s_k, v_k, \langle \text{end} \rangle\}$
- The action (a_t) space is defined as:

$$\text{Type}(a_i) \in \{\text{solve}, \text{verify}, \langle \text{end} \rangle\}$$

- The transition btw actions follows the rules:

$$\text{Type}(a_{i+1}) = \begin{cases} \text{verify}, & \text{if Type}(a_i) = \text{solve} \\ \text{solve}, & \text{if Type}(a_i) = \text{verify and Parser}(a_i) = \text{INCORRECT} \\ \langle \text{end} \rangle, & \text{if Type}(a_i) = \text{verify and Parser}(a_i) = \text{CORRECT} \end{cases}$$

$$* \text{Parser}(v_j) \in \{\text{CORRECT}, \text{INCORRECT}\}$$

- TL; DR

- LLMs learn both **self-verification** and **self-correction** through a 2-stage **SFT and RL training** framework.

- Problem States

Sampling Responses During Training/Inference

Please reason step by step, and put your final answer within `\boxed{ }`.

Problem: {problem}

Verification Refinement

You are a math teacher. I will give you a math problem and an answer.

Verify the answer's correctness without step-by-step solving. Use alternative verification methods.

Question: {problem}

Answer: {answer}

Verification:

Verification Collection

Refine this verification text to read as a natural self-check within a solution. Maintain logical flow and professionalism.

Key Requirements:

1. Avoid phrases like "without solving step-by-step" or "as a math teacher".

2. Treat the answer as your own prior solution.

3. Conclude with EXACTLY one of:

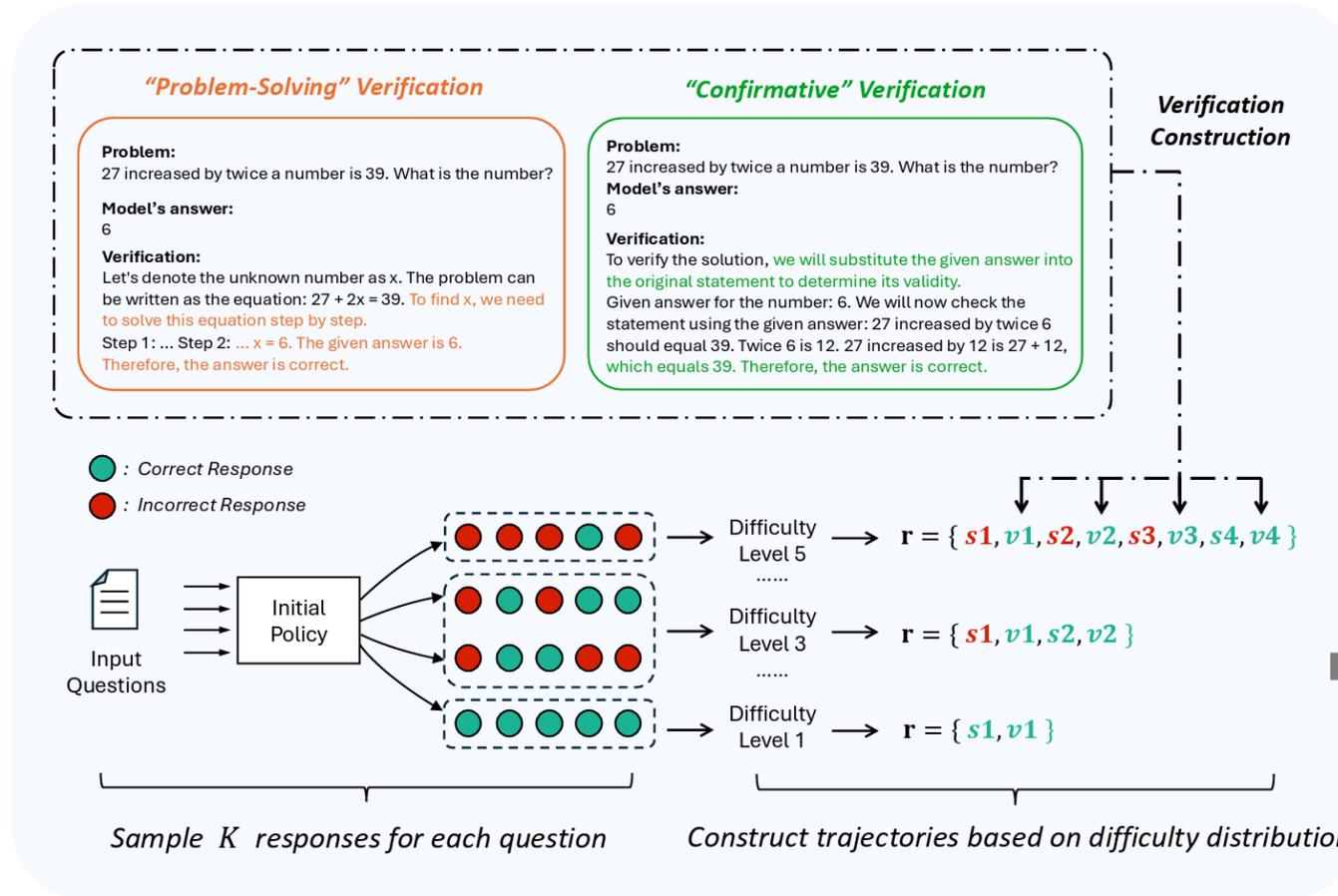
Therefore, the answer is correct.

Therefore, the answer is incorrect.

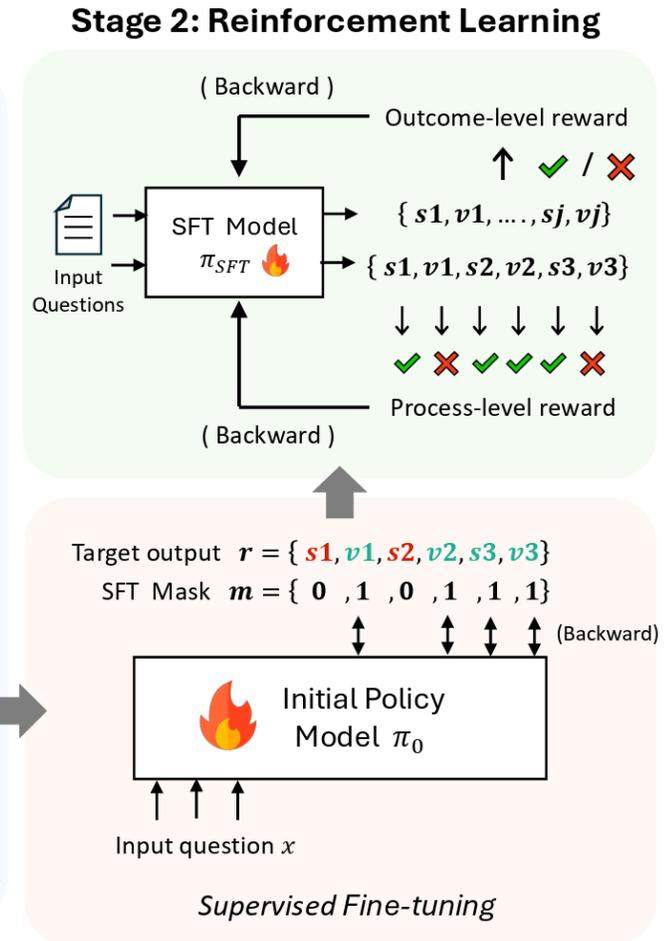
Therefore, the answer cannot be verified.

Original text: {verification}

• Suggestions

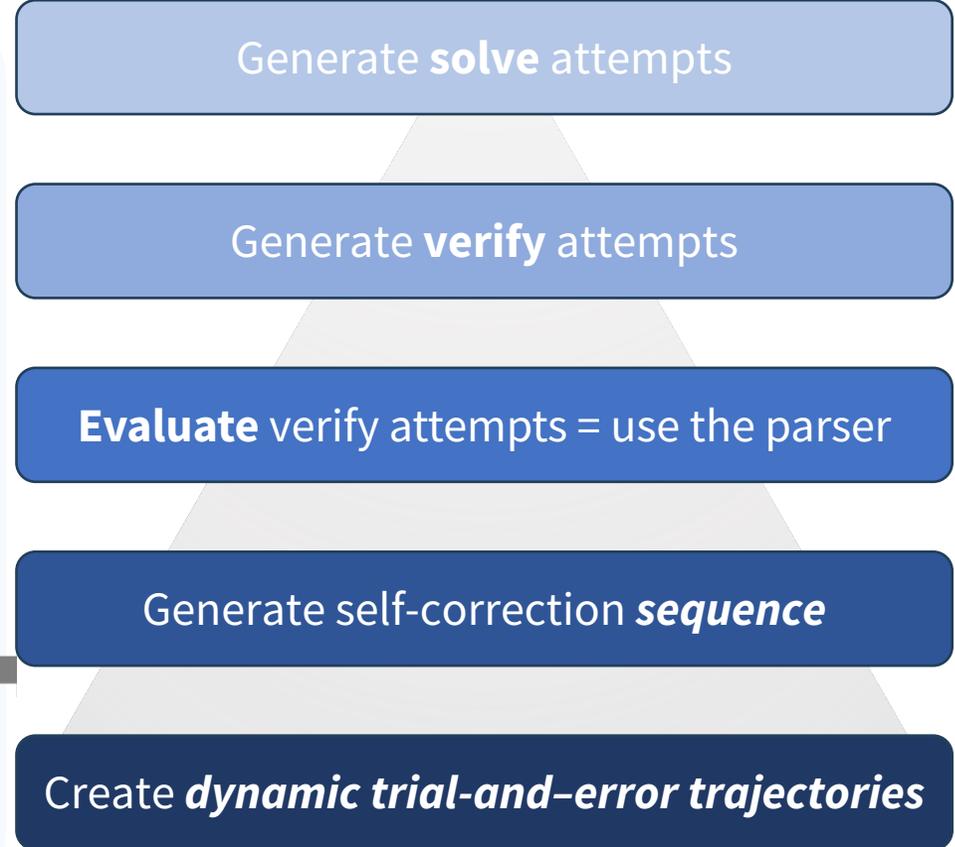
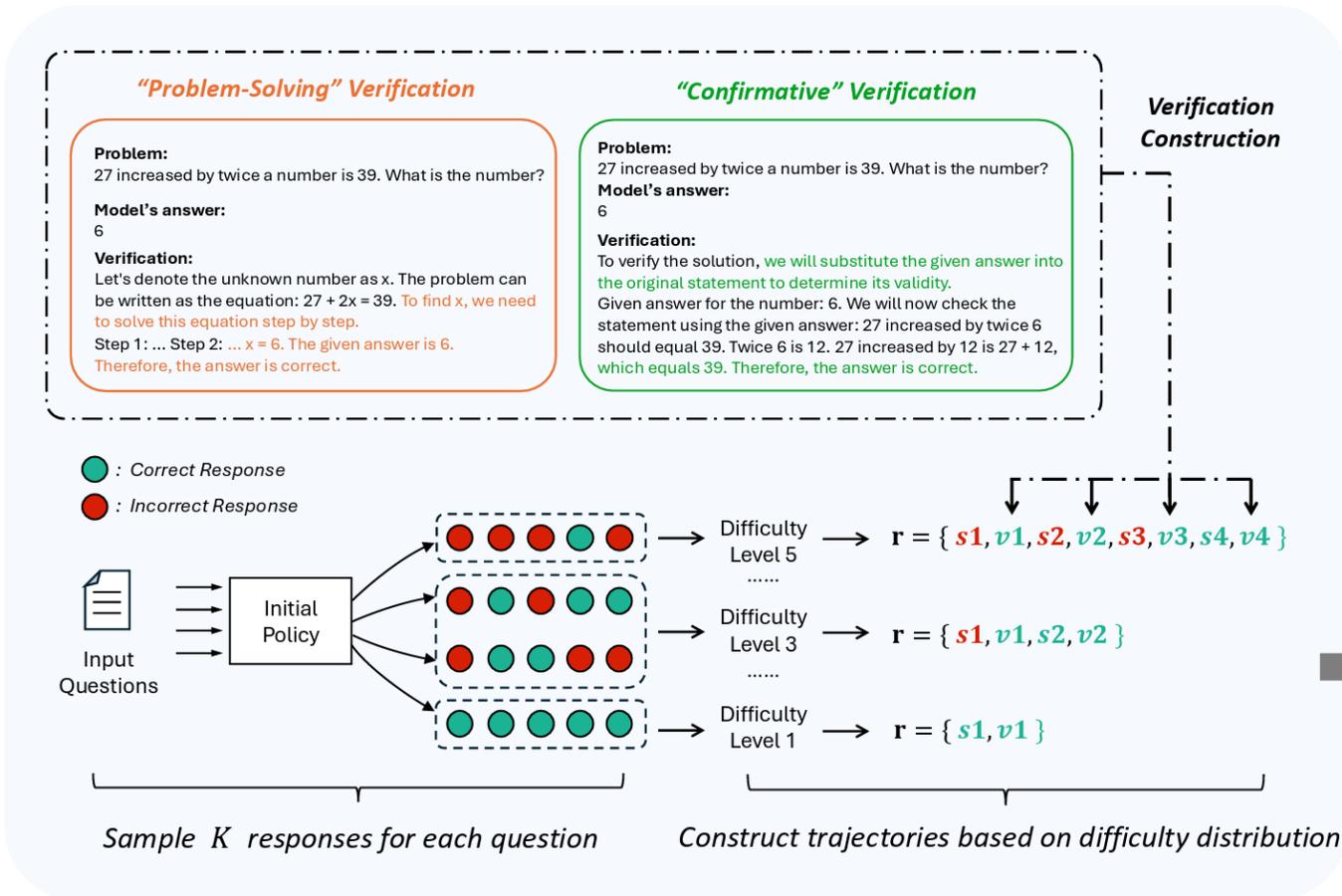


Stage 0: Data Construction



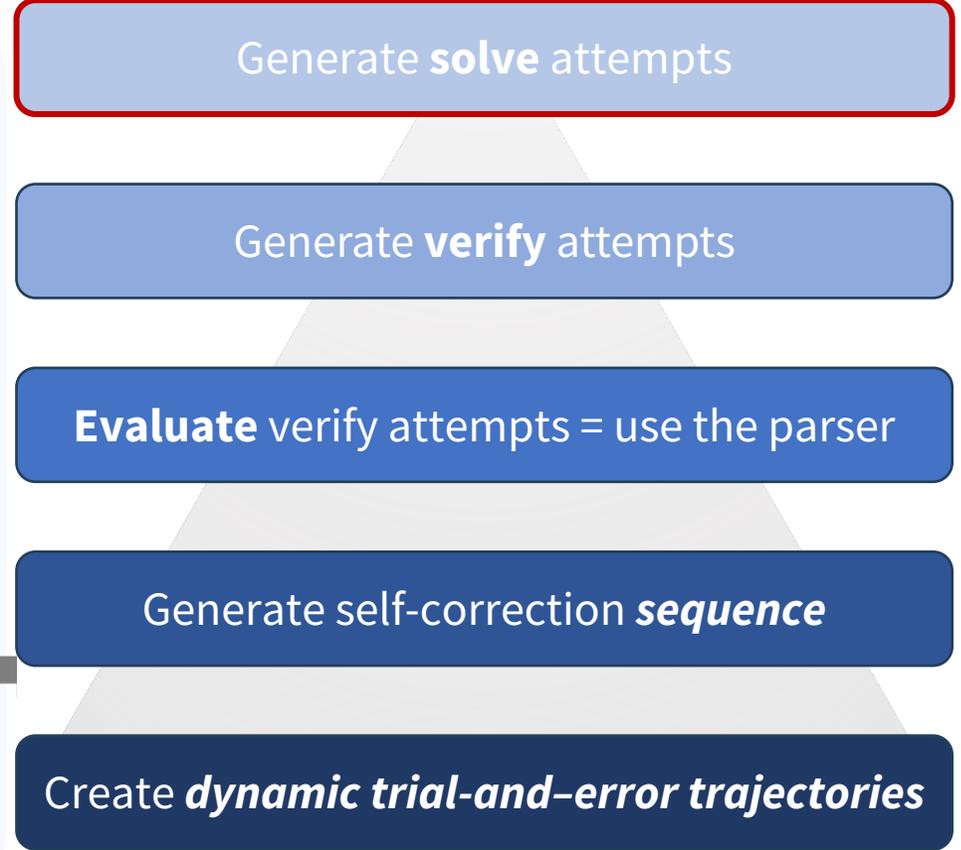
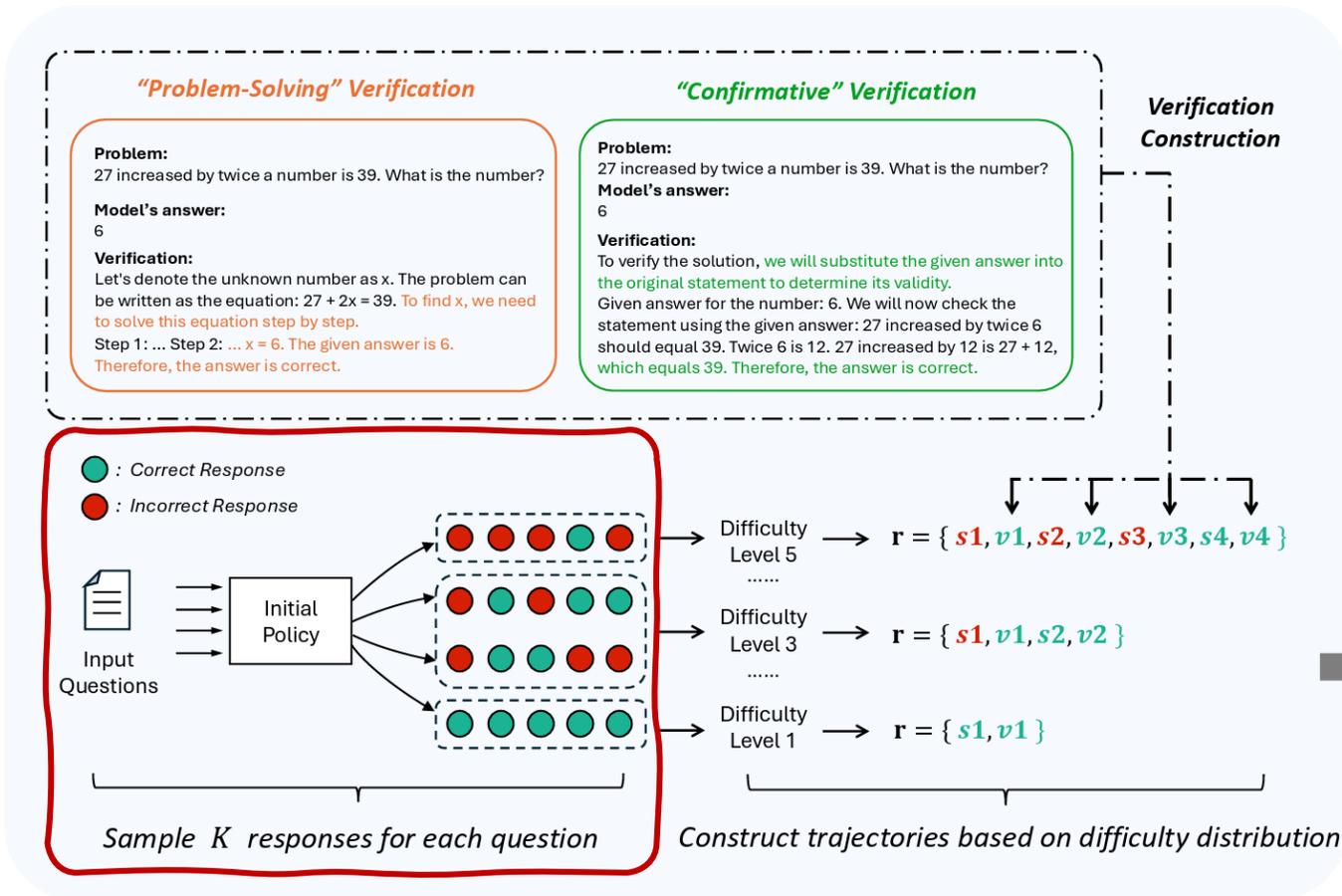
Stage 1: Behavior Initialization

• Suggestions – Stage 0 : Data Construction



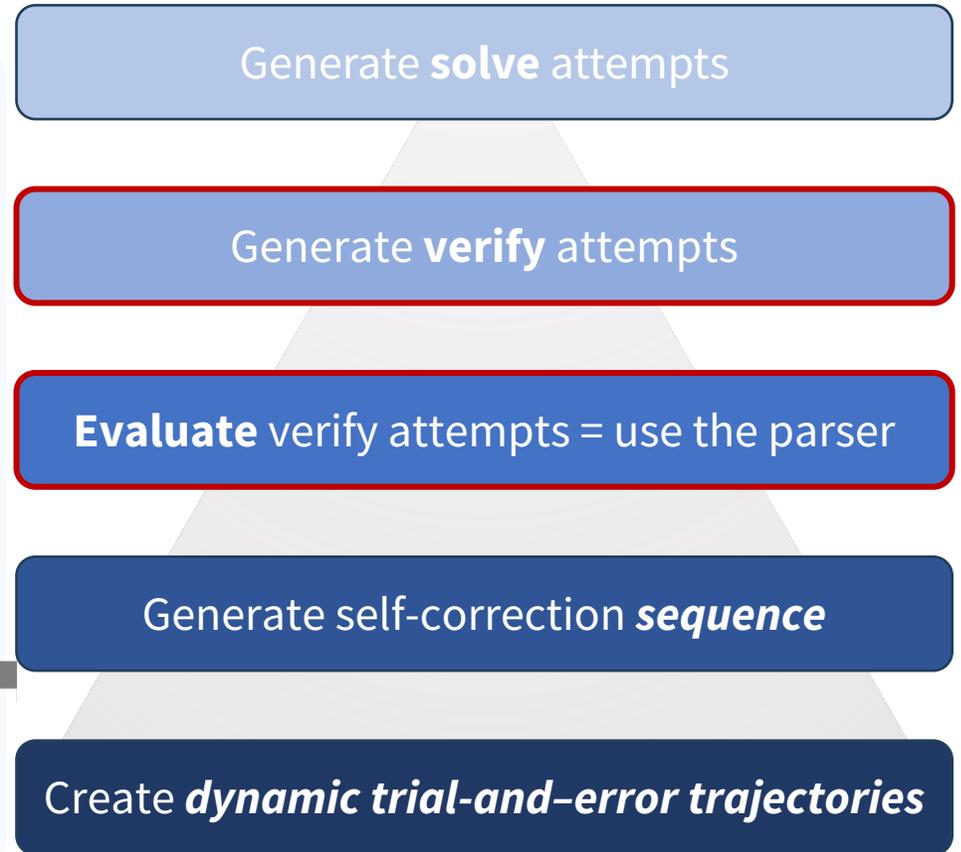
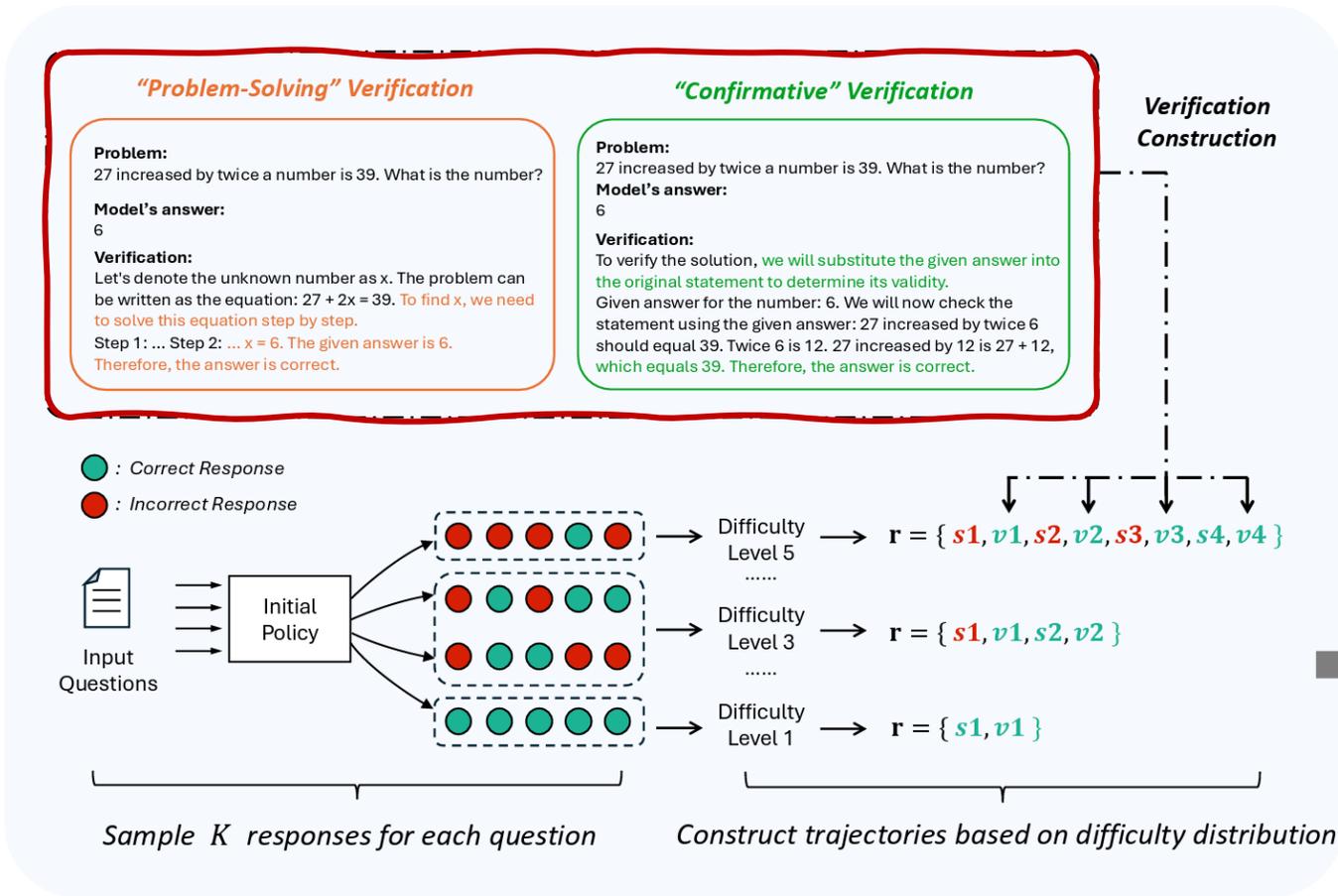
Stage 0: Data Construction

• Suggestions – Stage 0 : Data Construction



Stage 0: Data Construction

• Suggestions – Stage 0 : Data Construction



Stage 0: Data Construction

• Suggestions – Stage 0 : Data Construction

“Problem-Solving” Verification

Problem:
27 increased by twice a number is 39. What is the number?

Model’s answer:
6

Verification:
Let’s denote the unknown number as x. The problem can be written as the equation: $27 + 2x = 39$. To find x, we need to solve this equation step by step.
Step 1: ... Step 2: ... x = 6. The given answer is 6. Therefore, the answer is correct.

“Confirmative” Verification

Problem:
27 increased by twice a number is 39. What is the number?

Model’s answer:
6

Verification:
To verify the solution, we will substitute the given answer into the original statement to determine its validity.
Given answer for the number: 6. We will now check the statement using the given answer: 27 increased by twice 6 should equal 39. Twice 6 is 12. 27 increased by 12 is 27 + 12, which equals 39. Therefore, the answer is correct.

Verification Collection

Refine this verification text to read as a natural self-check within a solution. Maintain logical flow and professionalism.

Key Requirements:

1. Avoid phrases like "without solving step-by-step" or "as a math teacher".
2. Treat the answer as your own prior solution.
3. Conclude with EXACTLY one of:
Therefore, the answer is correct.
Therefore, the answer is incorrect.
Therefore, the answer cannot be verified.

Original text: {verification}

[Problem Solving Verification]

Solve the problem (1st) → Solve the problem again (2nd) : independently to see if the new answer is the same

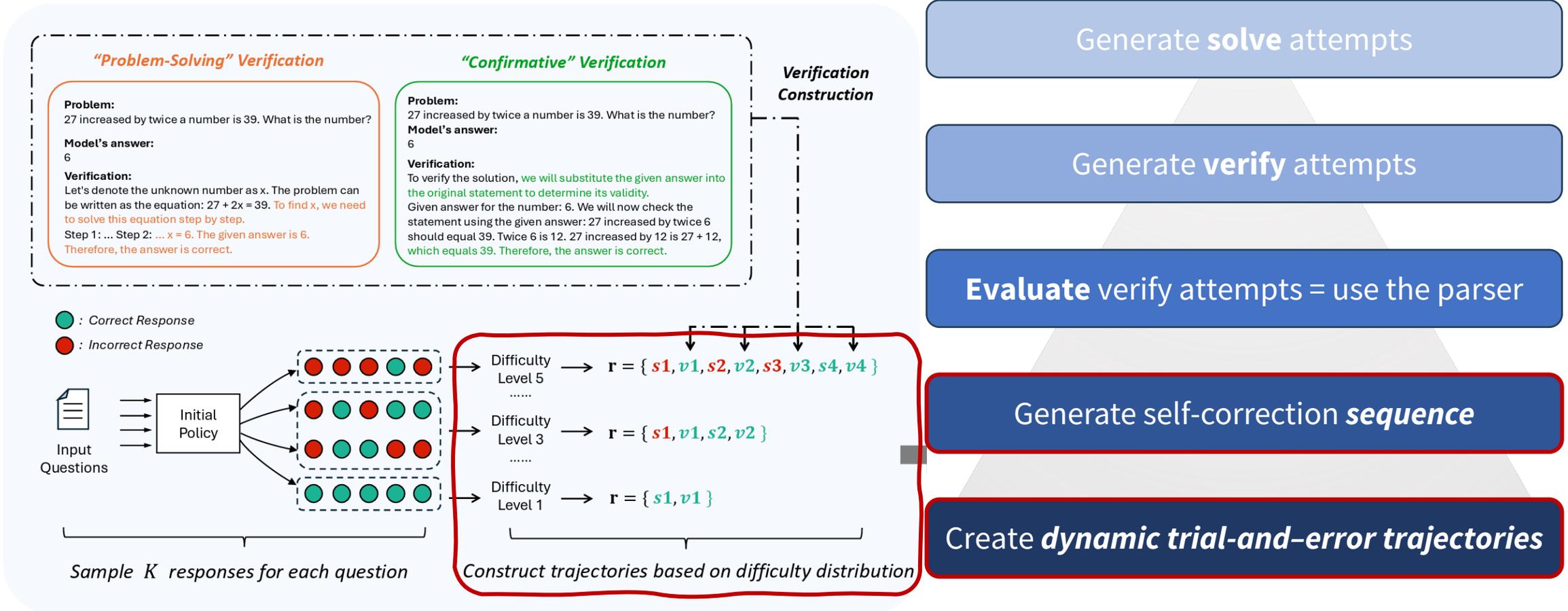
- match – the 1st answer is correct
- NOT match – the 1st answer is wrong

[Confirmative Verification]

Solve the problem (1st) → Check the answer : if 1st answer satisfies the conditions of the problem

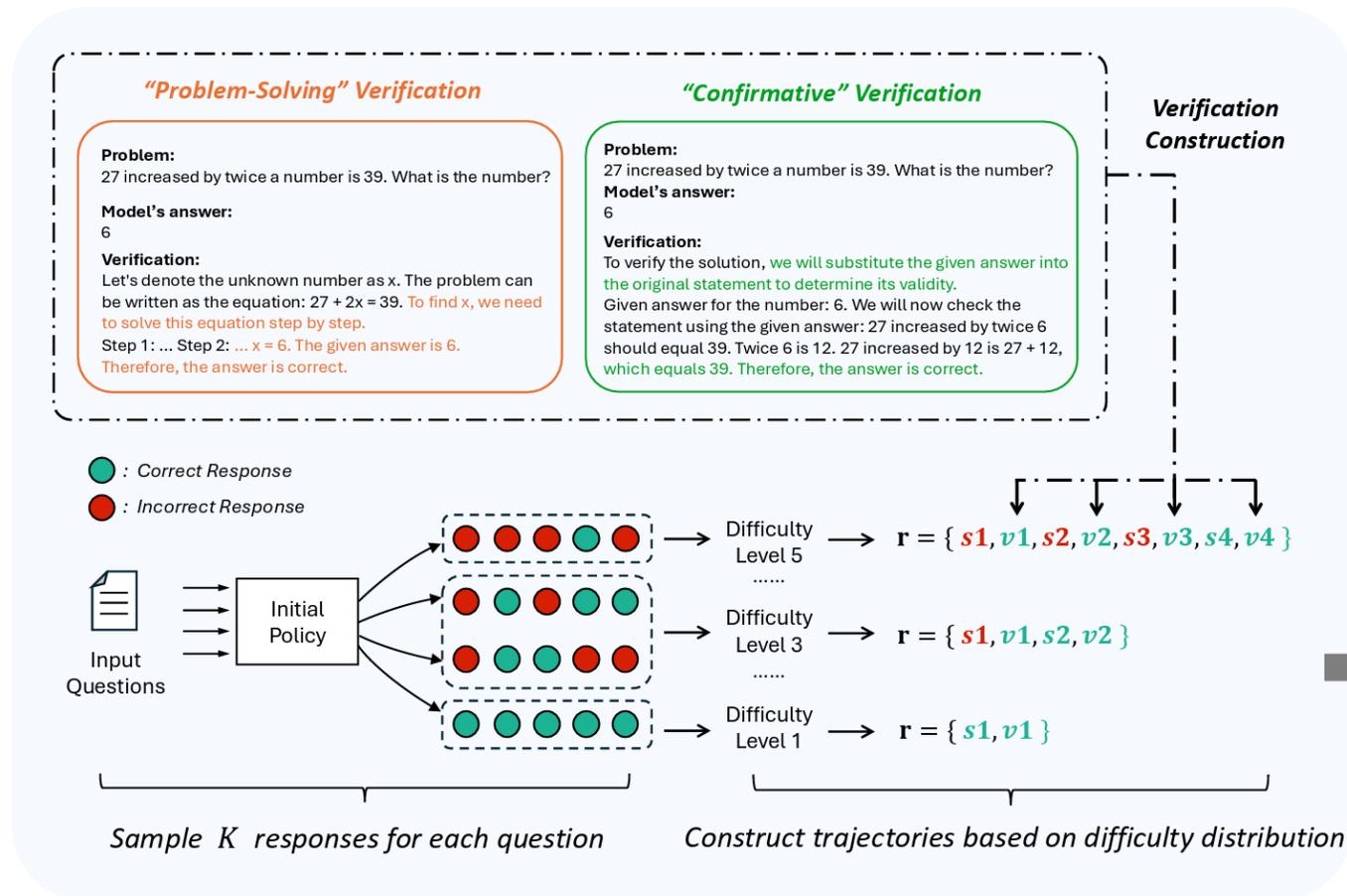
- NO contradiction – the 1st answer is correct
- Contradiction – the 1st answer is wrong

• Suggestions – Stage 0 : Data Construction



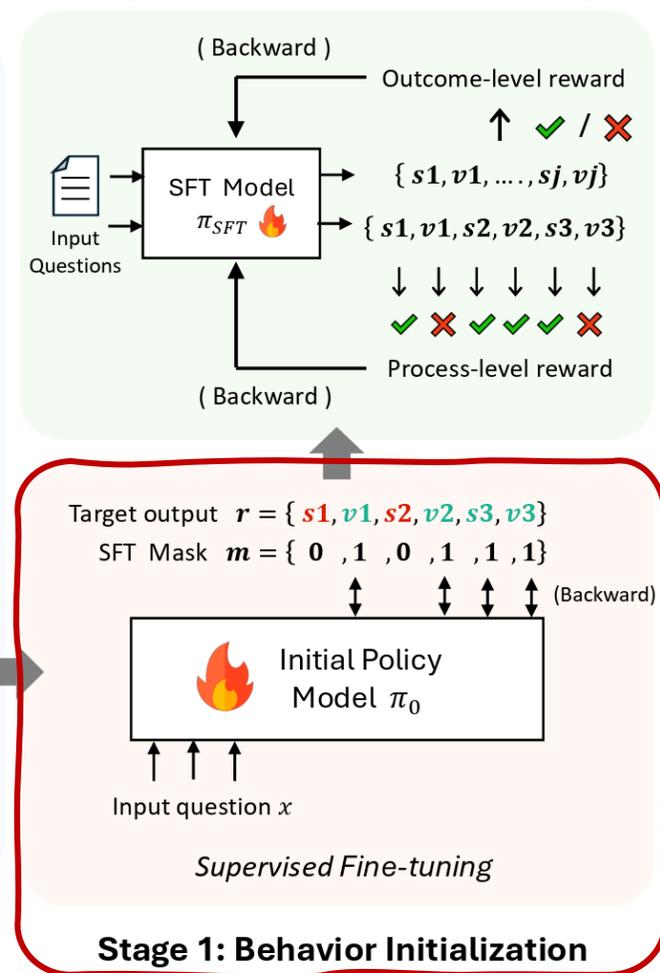
Stage 0: Data Construction

• Suggestions – Stage 1 : Behavior initialization (SFT)



Stage 0: Data Construction

Stage 2: Reinforcement Learning



• Suggestions – Stage 1 : Behavior initialization (SFT)

- SFT for Thinking Behavior Initialization: optimize the policy π by minimizing the objective:

"Problem-Solving" Verification

Problem:
27 increased by twice a number is 39. What is the number?

Model's answer:
6

Verification:
Let's denote the unknown number as x. The problem can be written as the equation: $27 + 2x = 39$. To find x, we need to solve this equation step by step.
Step 1: ... Step 2: ... x = 6. The given answer is 6. Therefore, the answer is correct.

"Confirmative" Verification

Problem:
27 increased by twice a number is 39. What is the number?

Model's answer:
6

Verification:
To verify the solution, we will substitute the given answer into the original statement to determine its validity.
Given answer for the number: 6. We will now check the statement using the given answer: 27 increased by twice 6 should equal 39. Twice 6 is 12. 27 increased by 12 is 27 + 12, which equals 39. Therefore, the answer is correct.

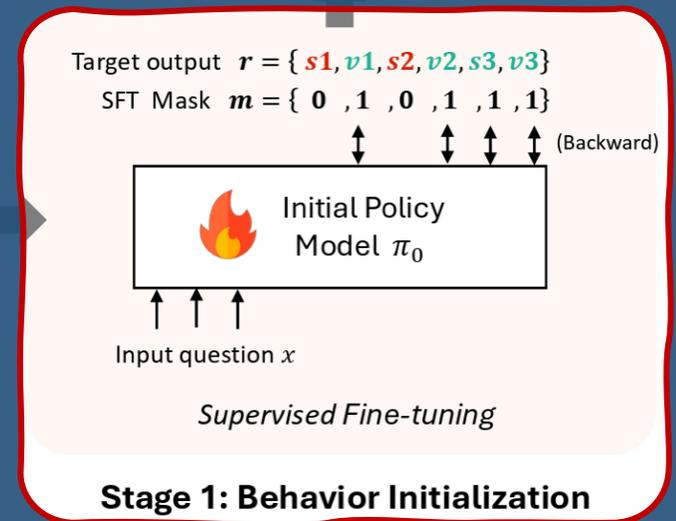
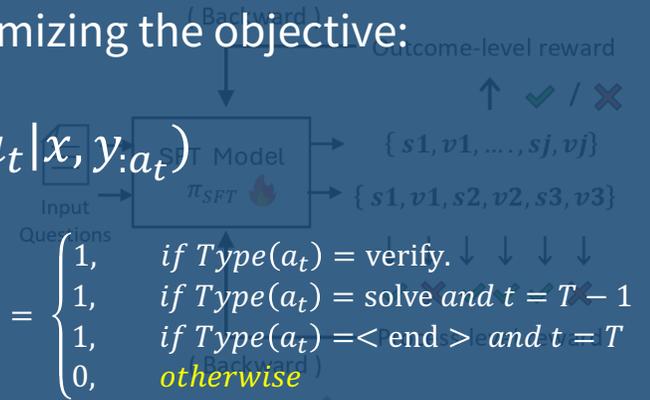
$$\mathcal{L} = -\mathbb{E}_{(x,y) \sim \mathcal{D}_{SFT}} \sum_{a_t \in y} \delta_{mask}(a_t) \log \pi(a_t | x, y: a_t)$$

$$\delta_{mask}(a_t) = \begin{cases} 1, & \text{if } Type(a_t) = \text{verify.} \\ 1, & \text{if } Type(a_t) = \text{solve and } t = T - 1 \\ 1, & \text{if } Type(a_t) = \langle \text{end} \rangle \text{ and } t = T \\ 0, & \text{otherwise} \end{cases}$$

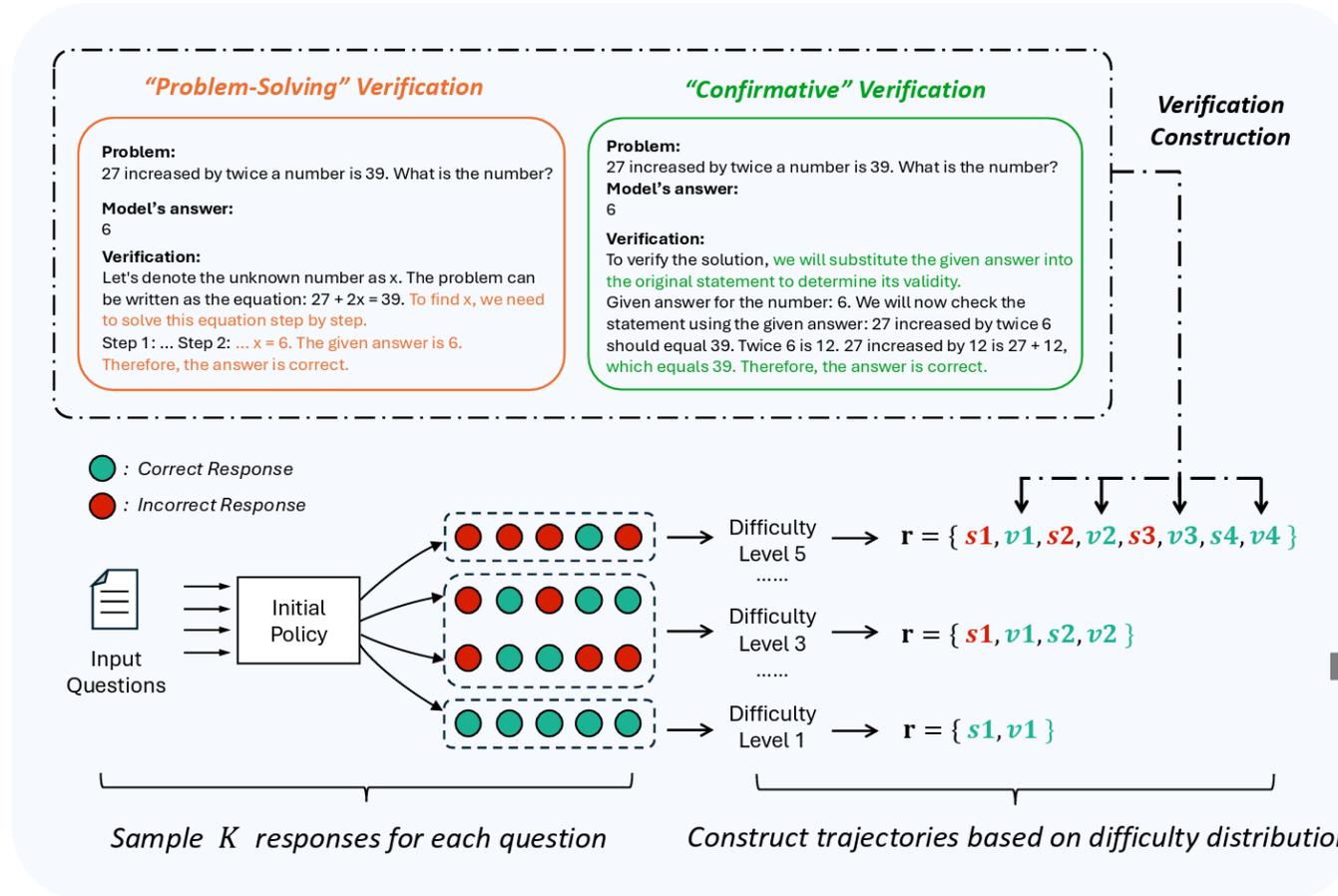
- Example: $2x + 3 = 9$

step	output	Is correct?	mask
Solve 1 s_1	$x = 3$	✗	0
Verify 1 v_1	$2 \times (3) + 3 = 9 \leftarrow \text{INCORRECT}$	✓	1
Solve 2 s_2	$x = 5$	✗	0
Verify 2 v_2	$2 \times (5) + 3 = 9 \leftarrow \text{INCORRECT}$	✓	1
Solve 3 s_3	$x = 2$	✓	1
Verify 3 v_3	$2 \times (2) + 3 = 9 \leftarrow \text{CORRECT}$	✓	1

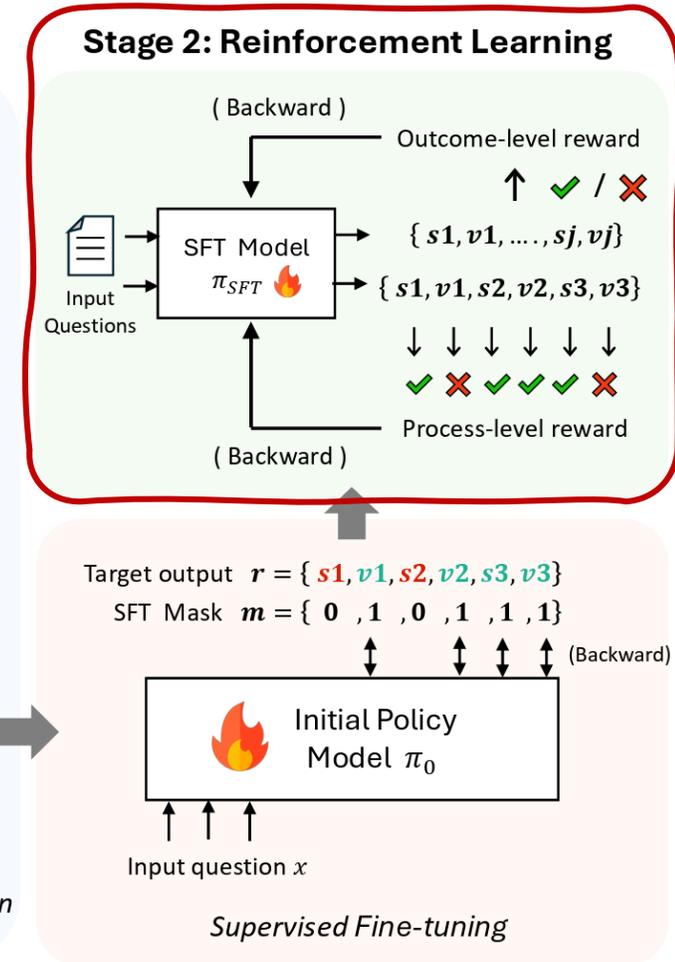
Stage 2: Reinforcement Learning



• Suggestions – Stage 2 : Reinforcement Learning



Stage 0: Data Construction

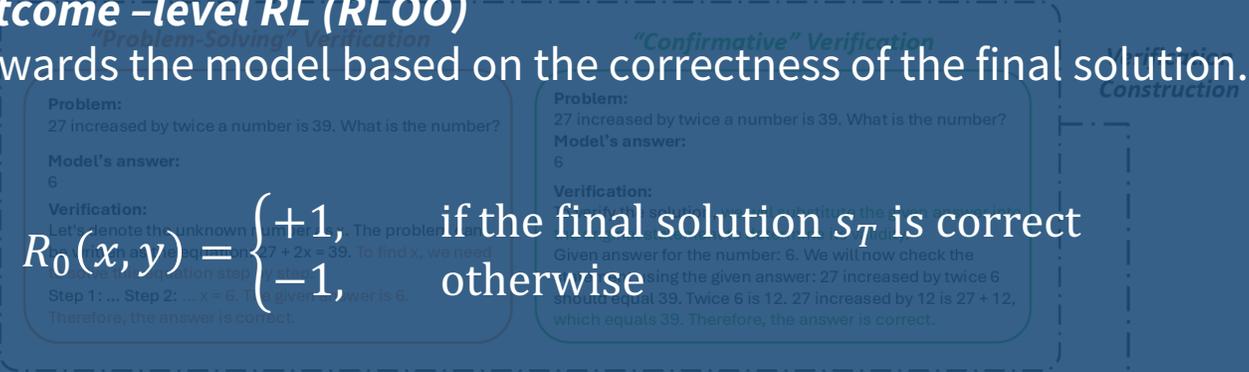


Stage 1: Behavior Initialization

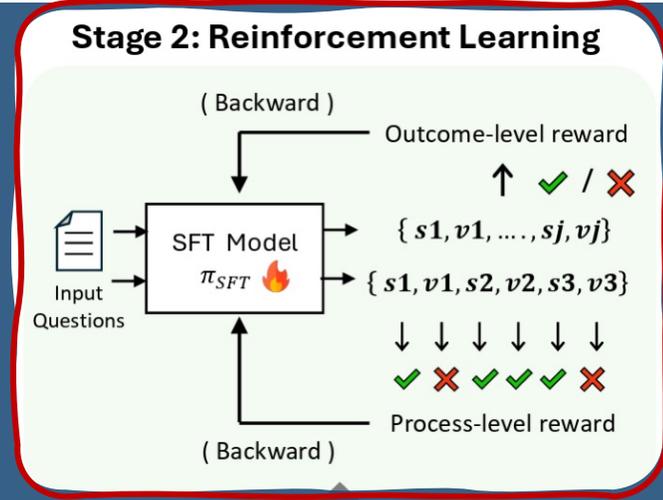
• Suggestions – Stage 2 : Reinforcement Learning

- Outcome-level RL (RL00)

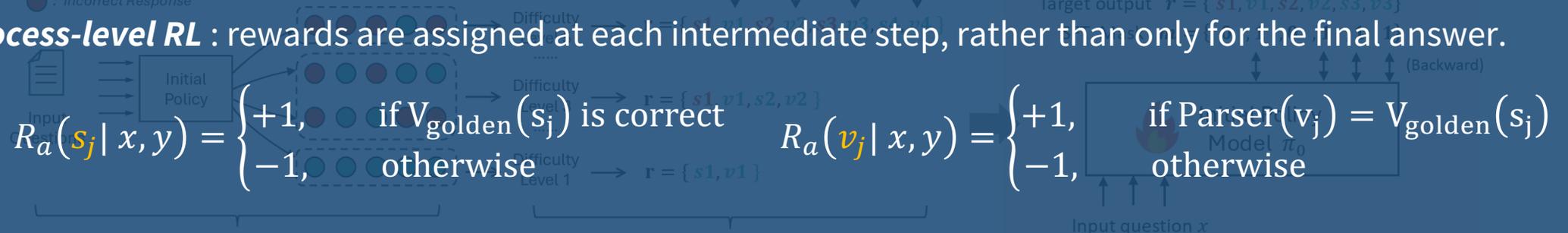
: rewards the model based on the correctness of the final solution.



$$R_0(x, y) = \begin{cases} +1, & \text{if the final solution } s_T \text{ is correct} \\ -1, & \text{otherwise} \end{cases}$$



- Process-level RL : rewards are assigned at each intermediate step, rather than only for the final answer.



$$R_a(s_j | x, y) = \begin{cases} +1, & \text{if } V_{\text{golden}}(s_j) \text{ is correct} \\ -1, & \text{otherwise} \end{cases}$$

$$R_a(v_j | x, y) = \begin{cases} +1, & \text{if Parser}(v_j) = V_{\text{golden}}(s_j) \\ -1, & \text{otherwise} \end{cases}$$

$$\mathcal{L}(\theta) = -\mathbb{E}_{x \sim D, y \sim \pi_{\theta_{\text{old}}}} \left[\frac{1}{|y|_a} \sum_{a \in y} \min(r_a(\theta) A(a|x, y), \text{clip}(r_a(\theta), 1 - \epsilon, 1 + \epsilon) A(a|x, y)) \right]$$

Stage 0: Data Construction

Stage 1: Behavior Initialization

Experimental Results Comparison

Performance analysis

SCoRe

• Performance of SCoRe

- Metrics

- Acc.@t1: accuracy at 1st attempt
- Acc.@t2: accuracy at 2nd attempt
- $\Delta(t1, t2) = \text{Acc.}@t2 - \text{Acc.}@t1$
- $\Delta i \rightarrow c(t1, t2)$: Error Correction
fraction of incorrect \rightarrow correct
- $\Delta c \rightarrow i(t1, t2)$: Stability
fraction of correct \rightarrow 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 $\Delta(t1, t2)$.

Approach	Acc.@t1	Acc.@t2	$\Delta(t1, t2)$	$\Delta^{i \rightarrow c}(t1, t2)$	$\Delta^{c \rightarrow i}(t1, t2)$
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}_{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%

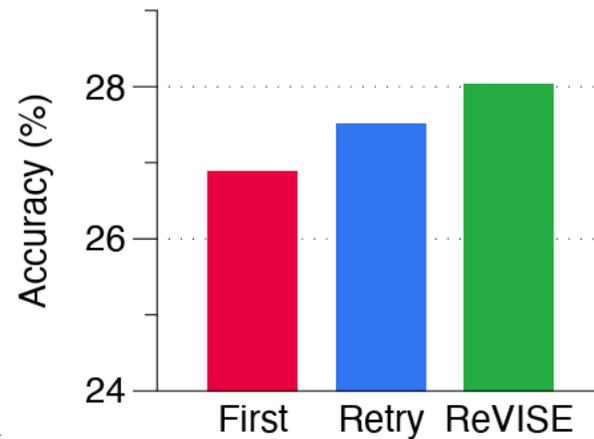
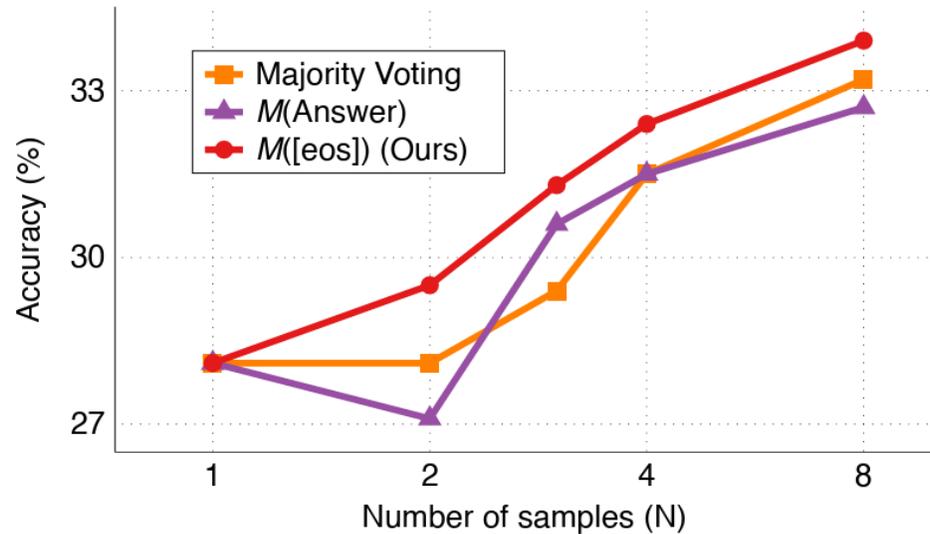
Table 3: **Performance of SCoRe on HumanEval.** SCoRe attains the highest self-correction performance (Accuracy@t2, $\Delta(t1, t2)$), and also outperforms other methods at offline correction (MBPP-R).

Method	MBPP-R	Acc.@t1	Acc.@t2	$\Delta(t1, t2)$	$\Delta^{i \rightarrow c}(t1, t2)$	$\Delta^{c \rightarrow i}(t1, t2)$
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%

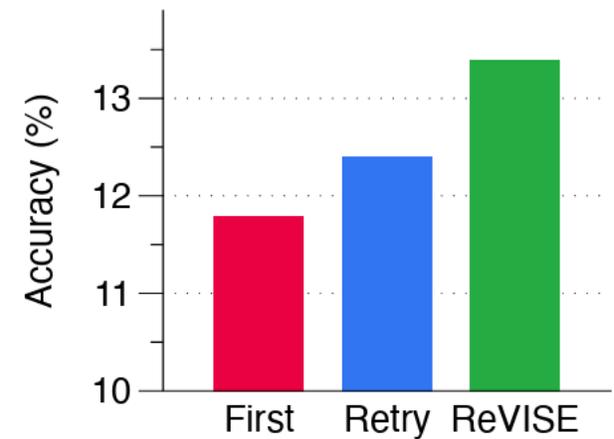
SCoRe achieves superior accuracy and self-correction performance

ReVISE

Methods	Llama-3.2-1B						Llama-3.1-8B		
	GSM8K			MATH-500			MATH-500		
	Maj@1	Maj@3	Maj@5	Maj@1	Maj@3	Maj@5	Maj@1	Maj@3	Maj@5
Few-shot CoT	5.7	6.8	7.2	3.0	2.6	3.2	23.4	22.2	23.2
SFT (Brown et al., 2020a)	22.1	23.5	26.4	10.4	10.6	11.4	27.8	31.0	33.2
RFT (Yuan et al., 2023)	26.2	26.8	28.6	12.6	12.4	12.8	30.8	33.2	35.6
STaR ⁺ (Zelikman et al., 2022)	26.2	27.1	29.9	11.4	13.1	13.4	30.4	31.8	32.8
ReVISE (Ours)	28.1	31.3	32.8	13.4	14.0	14.8	33.6	36.0	37.6



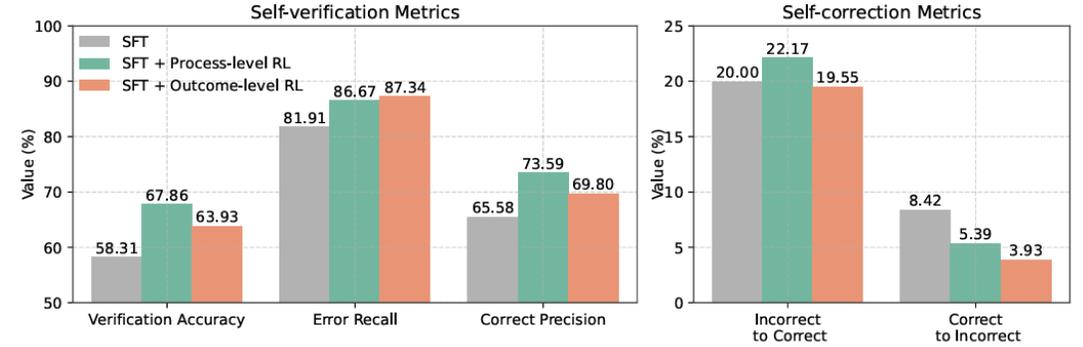
(a) GSM8K



(b) MATH

Model	Datasets							Average
	MATH 500	AIME 2024	AMC 2023	College Math	Olympiad Bench	GSM8K	GaokaoEn 2023	
<i>Frontier LLMs</i>								
GPT-4o*	76.6	9.3	47.5	48.5	43.3	92.9	67.5	55.1
Claude3.5-Sonnet*	78.3	16.0	-	-	-	96.4	-	-
GPT-o1-preview*	85.5	44.6	90.0	-	-	-	-	-
GPT-o1-mini*	90.0	56.7	95.0	57.8	65.3	94.8	78.4	76.9
<i>Top-tier Open-source Reasoning LLMs</i>								
Mathstral-7B-v0.1*	57.8	0.0	37.5	33.7	21.5	84.9	46.0	40.2
NuminaMath-72B-CoT*	64.0	3.3	70.0	39.7	32.6	90.8	58.4	51.3
LLaMA3.1-70B-Instruct*	65.4	23.3	50.0	42.5	27.7	94.1	54.0	51.0
Qwen2.5-Math-72B-Instruct*	85.6	30.0	70.0	49.5	49.0	95.9	71.9	64.6
<i>General Model: Llama-3.1-8B-Instruct</i>								
Llama-3.1-8B-Instruct	48.0	6.7	30.0	30.8	15.6	84.4	41.0	36.6
Llama-3.1-8B-Instruct + Original Solution SFT	31.0	3.3	7.5	22.0	8.0	58.7	28.3	22.7
Llama-3.1-8B-Instruct + Long CoT SFT	51.4	6.7	27.5	36.3	19.0	87.0	48.3	39.5
Llama-3.1-8B-S²R-BI (ours)	49.6	10.0	20.0	33.3	17.6	85.3	41.0	36.7
Llama-3.1-8B-S²R-PRL (ours)	53.6	6.7	25.0	33.7	18.5	86.7	43.1	38.2
Llama-3.1-8B-S²R-ORL (ours)	55.0	6.7	32.5	34.7	20.7	87.3	45.2	40.3
<i>General Model: Qwen2-7B-Instruct</i>								
Qwen2-7B-Instruct	51.2	3.3	30.0	18.2	19.1	86.4	39.0	35.3
Qwen2-7B-Instruct + Original Solution SFT	41.2	0.0	25.0	30.1	10.2	74.5	34.8	30.8
Qwen2-7B-Instruct + Long CoT SFT	60.4	6.7	32.5	36.3	23.4	81.2	53.5	42.0
Qwen2-7B-S²R-BI (ours)	61.2	3.3	27.5	41.1	27.1	87.4	49.1	42.4
Qwen2-7B-S²R-PRL (ours)	65.4	6.7	35.0	36.7	27.0	89.0	49.9	44.2
Qwen2-7B-S²R-ORL (ours)	64.8	3.3	42.5	34.7	26.2	86.4	50.9	44.1
<i>Math-Specialized Model: Qwen2.5-Math-7B</i>								
Qwen2.5-Math-7B	51.0	16.7	45.0	21.5	16.7	58.3	39.7	35.6
Qwen2.5-Math-7B-Instruct	83.2	13.3	72.5	47.0	40.4	95.6	67.5	59.9
Eurus-2-7B-PRIME*(Cui et al., 2025)	79.2	26.7	57.8	45.0	42.1	88.0	57.1	56.6
rStar-Math-7B*(Guan et al., 2025)	78.4	26.7	47.5	52.5	47.1	89.7	65.7	58.2
Qwen2.5-7B-SimpleRL*(Zeng et al., 2025)	82.4	26.7	62.5	-	43.3	-	-	-
Qwen2.5-Math-7B + Original Solution SFT	58.0	6.7	42.5	35.8	20.0	79.5	51.9	42.1
Qwen2.5-Math-7B + Long CoT SFT	80.2	16.7	60.0	49.6	42.1	91.4	69.1	58.4
Qwen2.5-Math-7B-S²R-BI (ours)	81.6	23.3	60.0	43.9	44.4	91.9	70.1	59.3
Qwen2.5-Math-7B-S²R-PRL (ours)	83.4	26.7	70.0	43.8	46.4	93.2	70.4	62.0
Qwen2.5-Math-7B-S²R-ORL (ours)	84.4	23.3	77.5	43.8	44.9	92.9	70.1	62.4

Evaluation on Verification and Correction (Base Model: Qwen2-7B-Instruct)



Base Model	Methods	Overall Verification Acc.	Initial Verification Acc.	
			$V_{golden}(s_0)$ = correct	$V_{golden}(s_0)$ = incorrect
<i>Llama3.1-8B-Instruct</i>	Problem-solving	80.10	87.28	66.96
	Confirmative	65.67	77.27	78.22
<i>Qwen2-7B-Instruct</i>	Problem-solving	73.28	90.24	67.37
	Confirmative	58.31	76.16	70.05
<i>Qwen2.5-Math-7B</i>	Problem-solving	77.25	91.21	56.67
	Confirmative	61.58	82.80	68.04

Conclusion

Overview

• High-level Comparison

- **SCoRe: Correction without explicit verification** (relies on trial-and-error via RL).
- **ReVISE: Verification-driven correction** (checks correctness before modifying responses).
- **S2R: Integrates both verification and correction**, improving correction reliability.

Aspect	SCoRe	ReVISE	S ² R
Self-Verification	✗	✓ intrinsic verification tokens	✓ explicit verification process
Self-Correction	✓ Multi-turn RL	✓ Confidence-aware decoding	✓ SFT + RL
Traning Approach	2-stage RL	2-stage Curriculum learning	2-stage RL
Reinforcement Learning	✓	✗	✓
Reward Sharping	✓	✗	✓
Test-time Scaling Strategy	Multi-turn Correction	Confidence-aware decoding	Adaptive correction (via self-verification)
Computational Efficiency	✗ High (RL+multi-turn)	✓ Low (no RL)	🙌 High (offline RL)

Thank You

Yejin Yoon

HYU NLP Lab.
Hanyang University, South Korea

stillwithyou@hanyang.ac.kr