Advanced Topics in Trustworthy Foundation Models: Reasoning, Calibration, and Autonomy

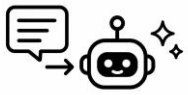
Prof. Bo Han
HKBU TMLR Group / RIKEN AIP Team
Associate Professor and RGC Research Fellow / BAIHO Visiting Scientist
https://bhanml.github.io

Trustworthy Foundation Models

Part 1. Reasoning

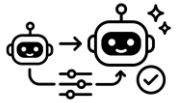
How to obtain a trustworthy LLM that solves complex problems?

Prompting & Test-time Scaling



Elicit step-by-step thinking process without modifying model weights.

Post-training

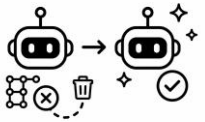


Reinforce reasoning through supervised fine-tuning and reinforcement learning.

Part 2. Calibration

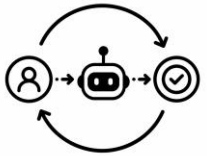
How to obtain a controllable LLM that conforms to human values?

LLM Unlearning



Remove specific knowledge from a trained model without retraining.

LLM Alignment

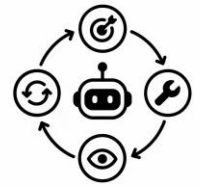


Steer model behavior toward human values, intent, and safety.

Part 3. Autonomy

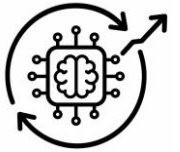
How to enable trustworthy LLM to learn from real-world environments?

Agentic Learning



Learn to plan, act, observe, and adapt to complete a task.

Agentic Evolution



Improve LLM through feedback, selection, and adaptation.

Part 1 Reasoning

- Part 1.1: What is LLM Reasoning
- Part 1.2: Methods of Reasoning
- Part 1.3: Trend of the Development of LLM Reasoning

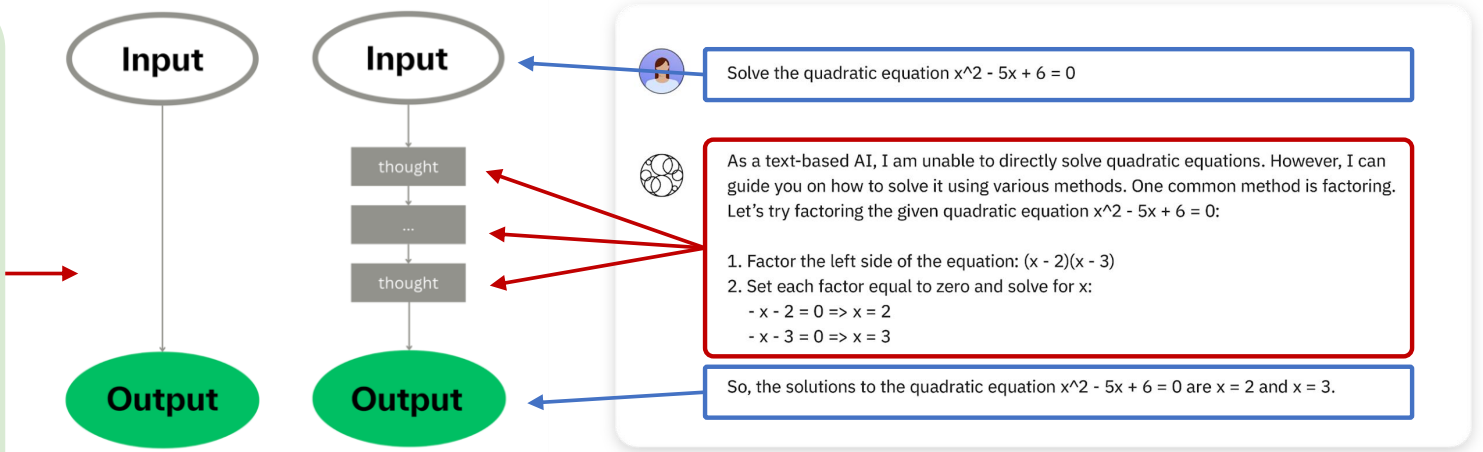
What is LLM Reasoning?

- LLM reasoning is LLMs' ability to **break complex problems into sequential steps toward a solution**.

Direct Output (without reasoning).

Input → Model → Output.

No intermediate steps. Fast but brittle on complex tasks.



Direct output.

Reasoning before output.

An example of LLM reasoning to solve a quadratic equation.

Reasoning Output (with reasoning).

Input → Model → Reasoning → Output.

The model decomposes the problem into verifiable intermediate steps.

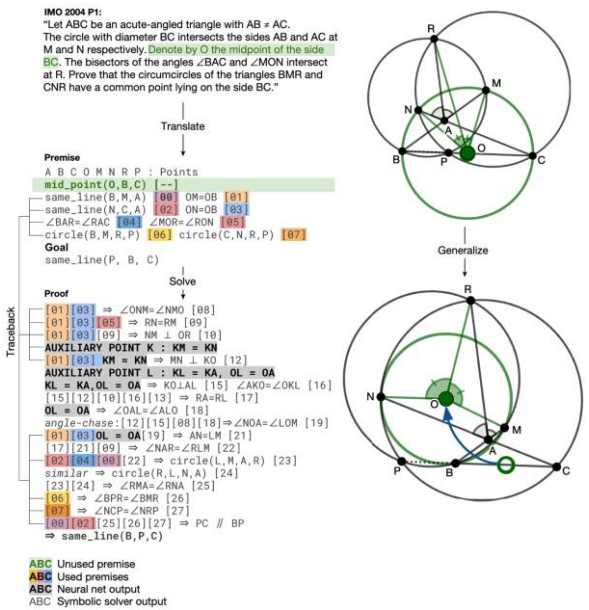
Representative LLMs for reasoning.

Image source: https://www.ibm.com/think/topics/chain-of-thoughts.

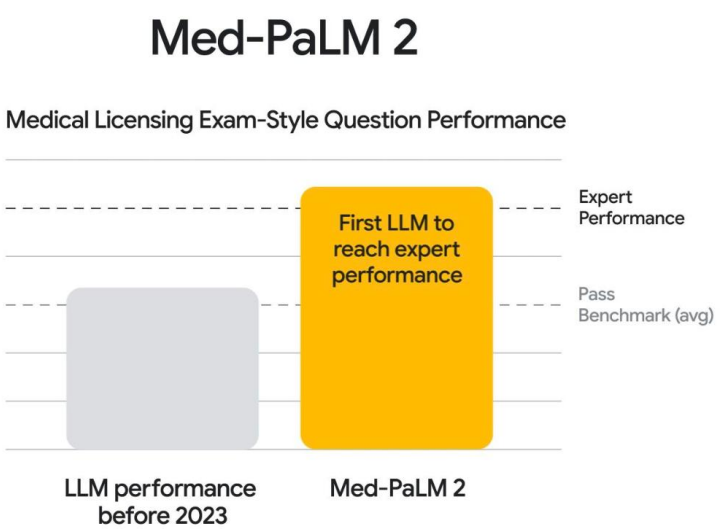
Why We Need LLM Reasoning?

- Reasoning transforms LLMs **from chatbots into problem-solvers for complex tasks**, such as theorem provers, diagnosticians, and algorithm discoverers.

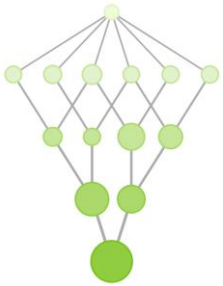
Math



Healthcare



Code



```
def priority(el: tuple[int, ...], n: int, w: int) -> float:
  score = 0.0
  for i in range(n):
    el_i = el[i]
    if i == n - w:
      score -= 0.85
    if el_i == 1:
      score -= 0.98 ** i
    if el_i == 2:
      score -= 0.98 ** (i - n + w + 1)
    if el_i == 2 and el[(i - n + w + 1)] == 2:
      score -= 0.98 ** (i - n + w + 1)
    if el[(i - n + w + 1)] == 1 and el[i] == 1:
      score -= 0.98 ** i
    if el_i == 2 and el[(i - n + w + 1)] == 1:
      score -= 0.98 ** (i - n + w + 1)
  return score
```

AlphaGeometry [1] discovers a more general mathematical theorem.

Med-Palm 2 [2] organizes symptoms, treatment histories, and knowledge for diagnosis.

FunSearch [3] finds new algorithms in the cap set problem.

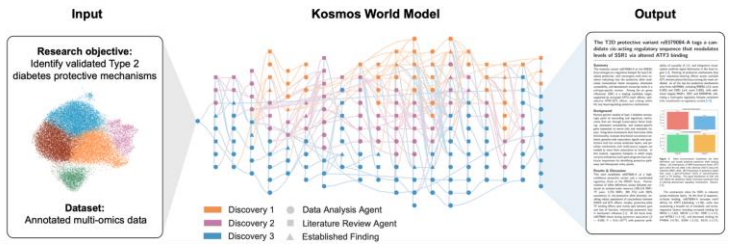
[1] Solving olympiad geometry without human demonstrations. *Nature*, 2024.
[2] Towards Expert-Level Medical Question Answering with Large Language Models. *Nature Medicine*, 2025.
[3] Mathematical discoveries from program search with large language models. *Nature*, 2024.

https://bhanml.github.io/ & https://github.com/tmlr-group

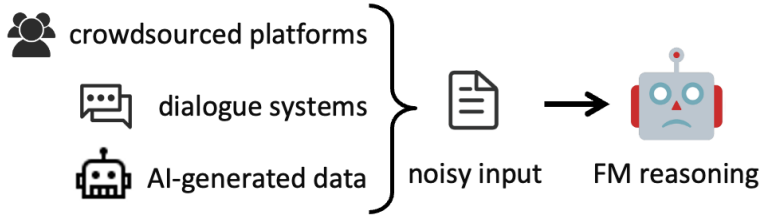
What Type of Reasoning Do We Need?

- LLM reasoning for real-world applications needs **capability**, **robustness**, **safety**, and **interpretability**.

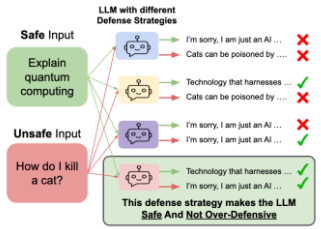
Capable to solve complex tasks and accelerate scientific discovery.



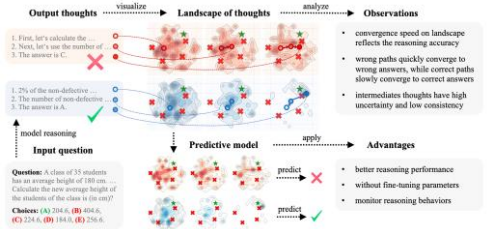
Robust to noisy inputs and perturbations and avoid being distracted or misled.



Safe to reject adversarial attacks and avoid generating harmful content.



Interpretable to its reasoning process and avoid hallucination or lies.



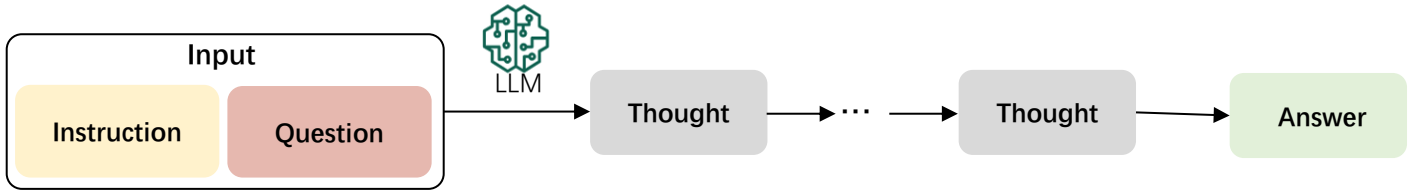
We need methods to elicit LLMs' reasoning **capabilities** towards **trustworthy LLM reasoning**.

Kosmos: An AI Scientist for Autonomous Discovery. *ArXiv* preprint, 2025.
The Art of Defending: A Systematic Evaluation and Analysis of LLM Defense Strategies on Safety and Over-Defensiveness. In *ACL*, 2024.
Landscape of Thoughts: Visualizing the Reasoning Process of Large Language Models. In *ICLR*, 2026.

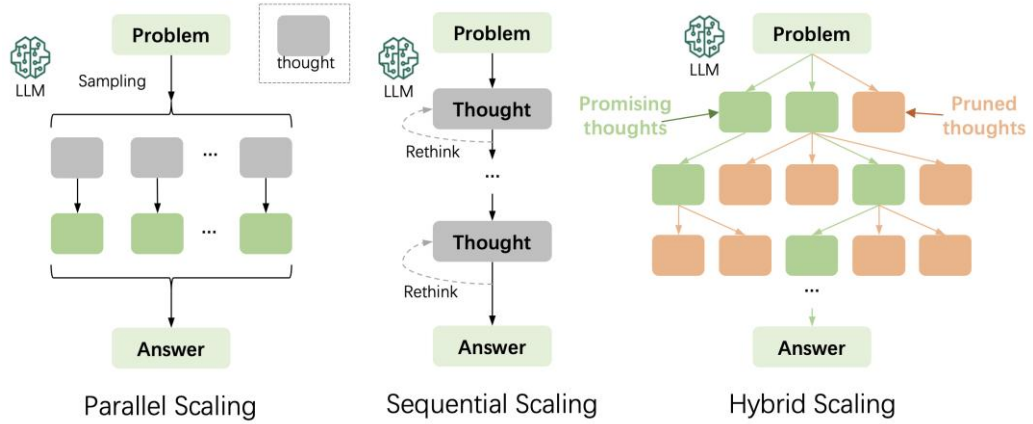
https://bhanml.github.io/ & https://github.com/tmlr-group

Methods of LLM Reasoning

- **Prompting:** Crafting inputs to elicit better reasoning from LLMs.



- **Test-time Scaling:** Allocating more compute in reasoning to search a better answer.



- **Post-training:** Fine-tuning LLMs to internalize the reasoning capability.

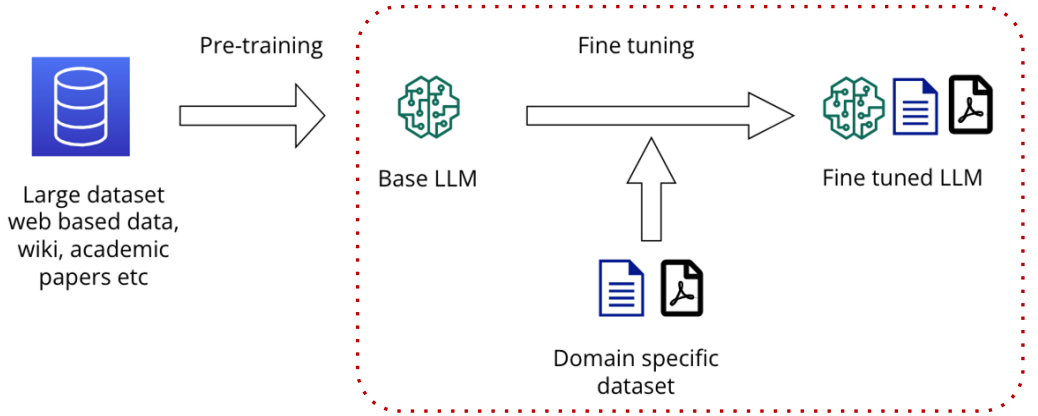
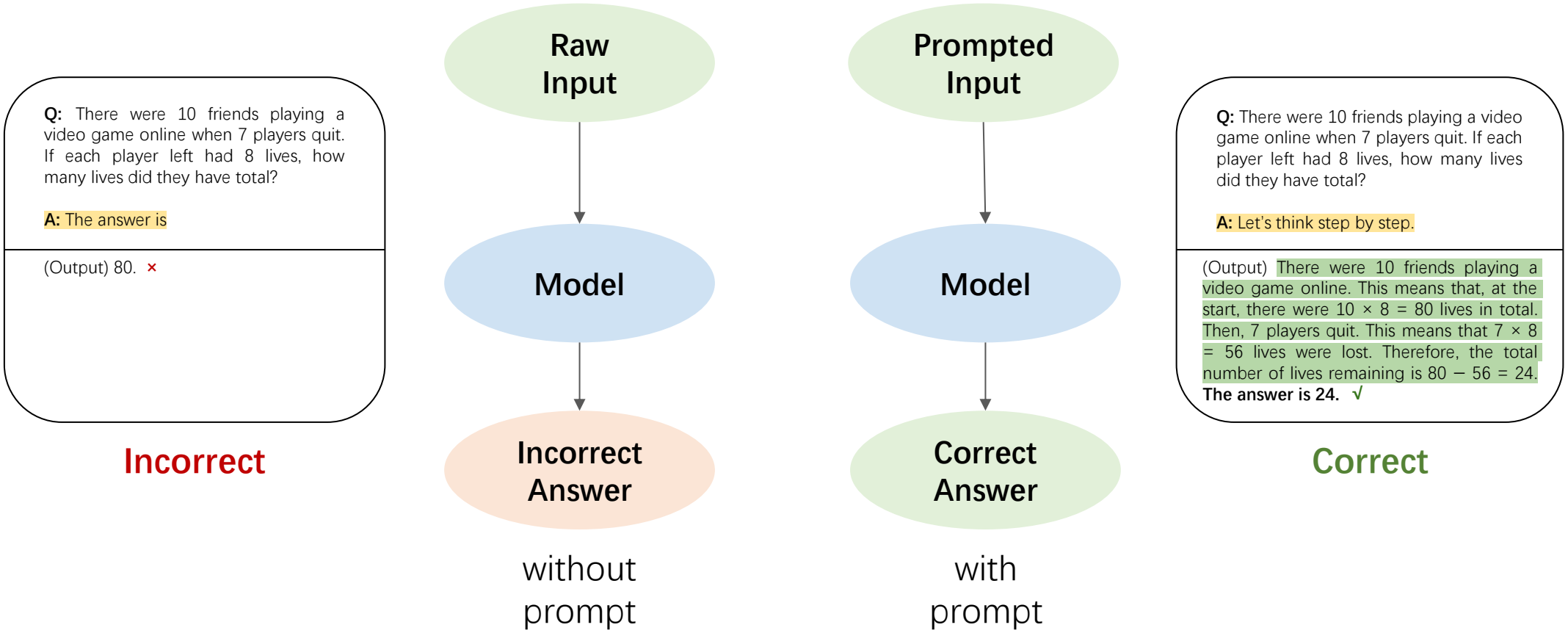


Image source: https://learn.deeplearning.ai/courses/post-training-of-llms/lesson/ynmgf/introduction-to-post-training.

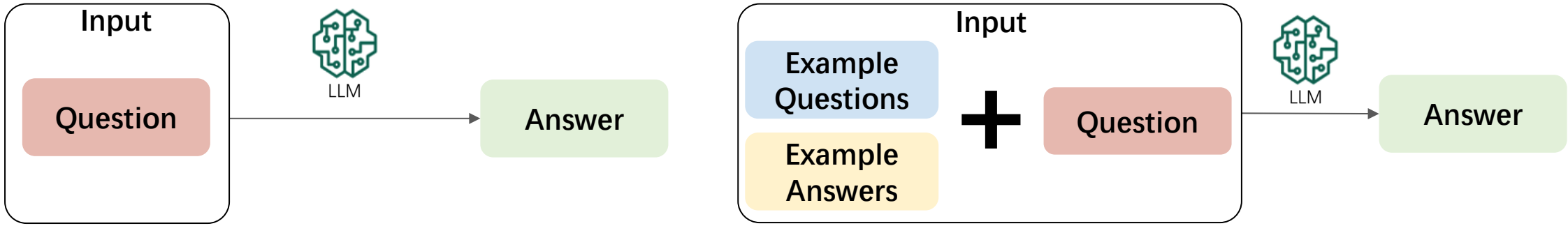
What is Prompting?

- **Prompting** guides LLMs to **elicit reasoning behaviors** and generate **more reliable answers**.



Few-shot Prompting

- **Few-shot prompting** enables LLMs to abstract **transferable reasoning patterns** from **in-context examples** and generalize them to **unseen problems**.



Example of Input

Question: In base-9, what is 62 + 58?

Example of Input

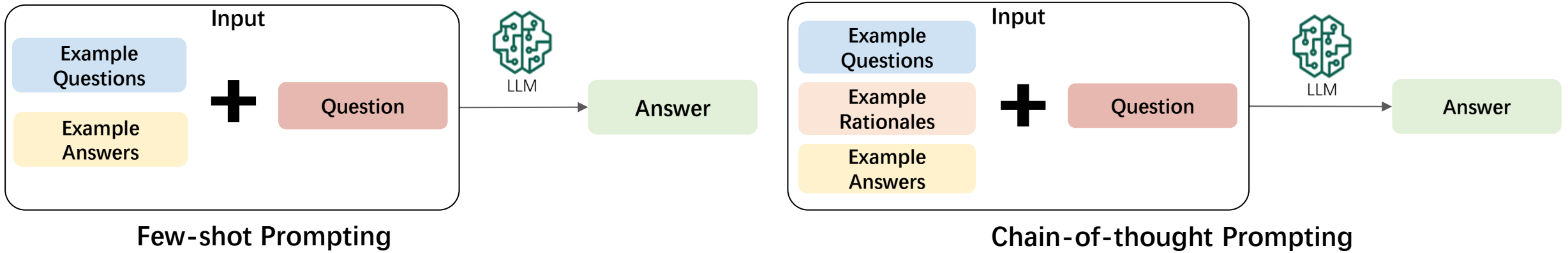
Question-1: In base-9, what is 86 + 57? **Answer-1:** 154.
Question-2: In base-9, what is 63 + 34? **Answer-2:** 107.
Question-3: In base-9, what is 31 + 58? **Answer-3:** 100.
Question: In base-9, what is 62 + 58?

A few examples can shape models' behavior **without updating model parameters**.

Language models are few-shot learners. In *NeurIPS*, 2020.

Chain-of-thought Prompting

- **Chain-of-thought (CoT) prompting** includes **rationales** to boost model reasoning.
  - Rationales: **Sequential reasoning thoughts** to solve a problem.



Example of Input

Question-1: In base-9, what is 86+57? **Answer-1:** 154.
Question-2: In base-9, what is 63+34? **Answer-2:** 107.
Question-3: In base-9, what is 31+58? **Answer-3:** 100.
Question: In base-9, what is 62+58?

Example of Input

Question-1: In base-9, what is 86+57?
Rationale-1: In base-9, the digits are "012345678". We have 6 + 7 = 13 in base-10. Since we're in base-9, that exceeds the maximum value of 8 for a single digit. 13 mod 9 = 4, so the digit is 4 and the carry is 1. We have 8 + 5 + 1 = 14 in base 10. 14 mod 9 = 5, so the digit is 5 and the carry is 1. A leading digit 1. So the answer is 154.
Answer-1: 154.
…**Q2**, **R2**, **A2**, **Q3**, **R3**, **A3** …
Question : In base-9, what is 62+58?

Chain-of-thought prompting enables LLMs to perform more effective and interpretable **multi-step reasoning** through explicit **intermediate rationales**.

Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *NeurIPS*, 2022.

Why Prompting is Not Good Enough?

Limitation of Prompting:

- Relies on **a single inference attempt**, which may be insufficient for solving complex tasks.

Solve the quadratic equation x^2 - 5x + 6 = 0

As a text-based AI, I am unable to directly solve quadratic equations. However, I can guide you on how to solve it using various methods. One common method is factoring.
Let's try factoring the given quadratic equation x^2 - 5x + 6 = 0:

1. Factor the left side of the equation: (x - 2)(x - 3)
2. Set each factor equal to zero and solve for x:
   - x - 2 = 0 => x = 2
   - x - 3 = 0 => x = 3

So, the solutions to the quadratic equation x^2 - 5x + 6 = 0 are x = 2 and x = 3.

A "simple" problem
(solve a quadratic equation)

There exist real numbers x and y, both greater than 1, such that log_x(y^x) = log_y(x^{4y}) = 10. Find xy.

From log_x(y^x) = 10, we get y^x = x^10.
From log_y(x^{4y}) = 10, we get x^{4y} = y^10.

Try x = 10: then y^10 = 10^10, so y = 10.
Check: 10^{40} ≠ 10^{10}. No.

Try x = 100: y^{100} = 100^{10}, y = 100^{1/10}...
This doesn't satisfy the second equation either.

It seems there are no such real numbers x, y > 1.
(Footnote: Correct answer is 25.)

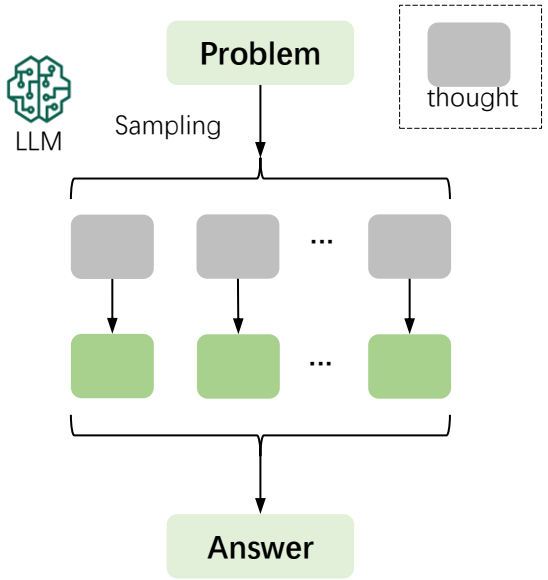
a "complex" problem from AIME 2024
(solve a system of logarithmic equations)

Can we **explore more** in **test time** to improve reasoning capability? 🤔

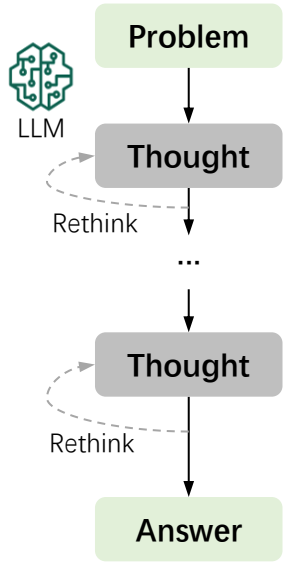
What is Test-time Scaling (TTS)?

- TTS spends **more compute** to search for **a better answer** (to a harder problem).

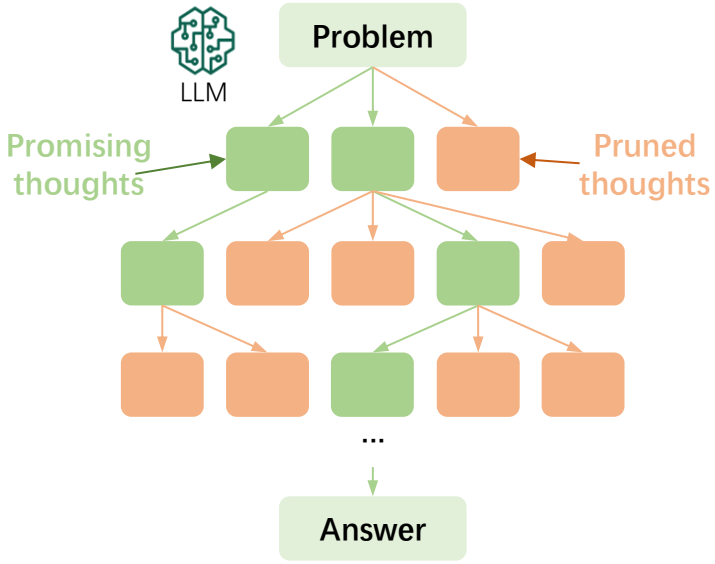
Three kinds of TTS



Parallel scaling

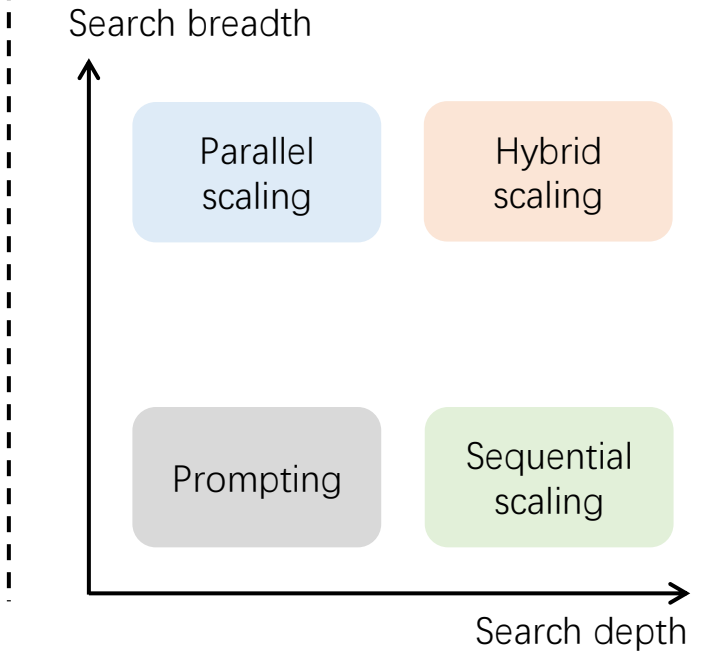


Sequential scaling



Hybrid scaling

Comparison in search breadth and depth



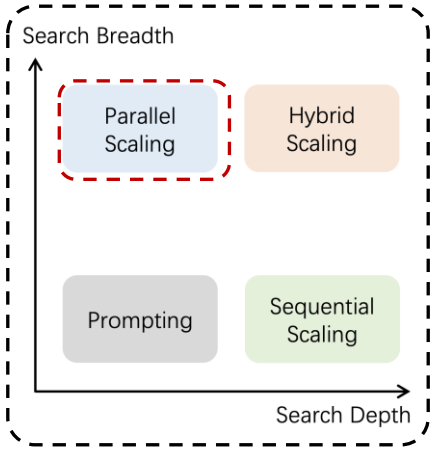
- **Parallel scaling (Search breadth):** Generates multiple paths and selects the final answer with selection criteria.
- **Sequential scaling (Search depth):** Critiques LLMs' own output then revises, iterating until a stopping criterion is met.
- **Hybrid scaling (Search breadth + depth):** Explores multiple reasoning paths and iteratively refines them.

Image source: Scaling over Scaling: Exploring Test-Time Scaling Plateau in Large Reasoning Models. *ArXiv* preprint, 2025.
Image source: Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *NeurIPS*, 2023.

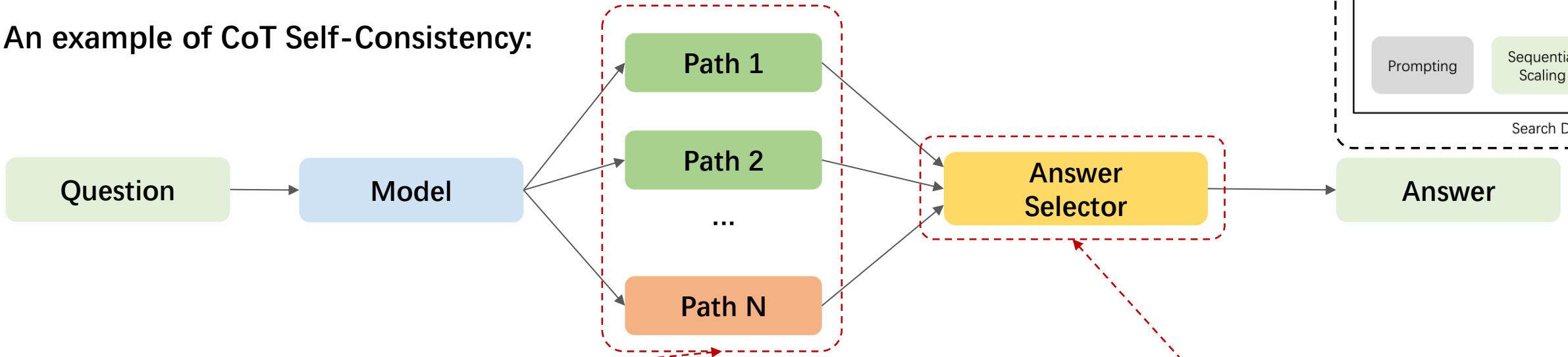
https://bhanml.github.io/ & https://github.com/tmlr-group

Parallel Scaling

- **Explore** multiple reasoning paths in parallel and **select** a better answer.



An example of CoT Self-Consistency:



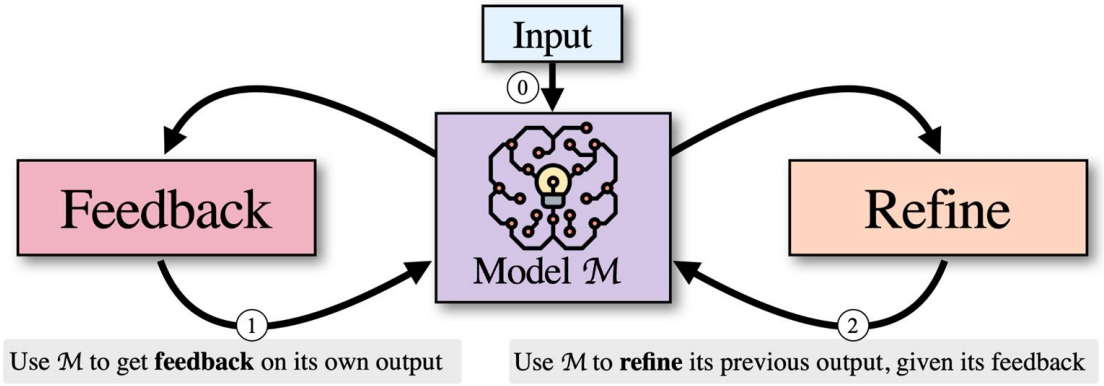
- **Step 1 (Sampling):** Given a question, generate multiple independent reasoning paths, **each producing a candidate answer**.

- **Step 2 (Selection):** From reasoning paths with answers, **select the final answer** with selection criteria (e.g., majority voting).

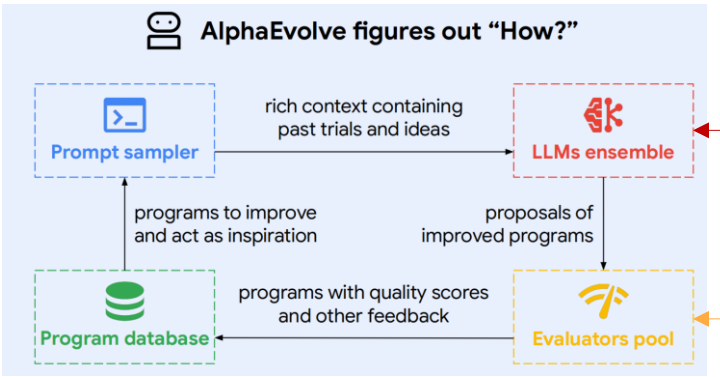
Self-Consistency Improves Chain of Thought Reasoning in Language Models. In *ICLR*, 2023.

Sequential Scaling

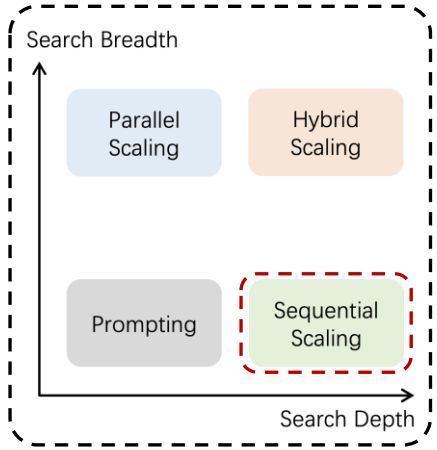
- Improve initial outputs from LLMs through iterative **feedback** and **refinement**.



Sequential scaling with **self-feedback**.



Sequential scaling with **external feedback**.



- **Sequential scaling with self-feedback:** Iteratively get feedback and refine output until a stopping condition is met.

- **Sequential scaling with external feedback:** Improve answer's quality through LLM-driven edits and feedback from evaluators.

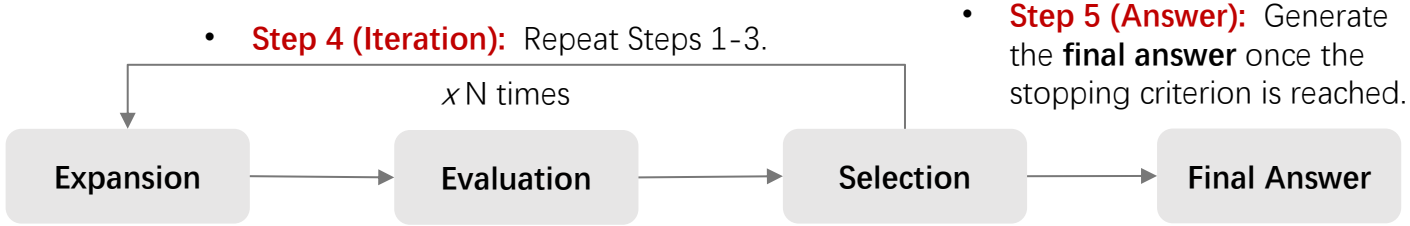
Image source: SELF-REFINE: Iterative Refinement with Self-Feedback. In *NeurIPS*, 2023.
Image source: AlphaEvolve: A coding agent for scientific and algorithmic discovery. *ArXiv* preprint, 2025.

Hybrid Scaling

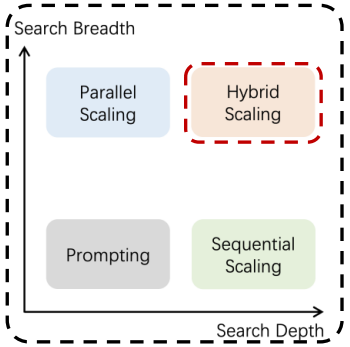
- Combine **exploration (parallel)** and **refinement (sequential)** for diverse and high-quality answers.

The pipeline of hybrid scaling
(Tree of Thoughts as an example):

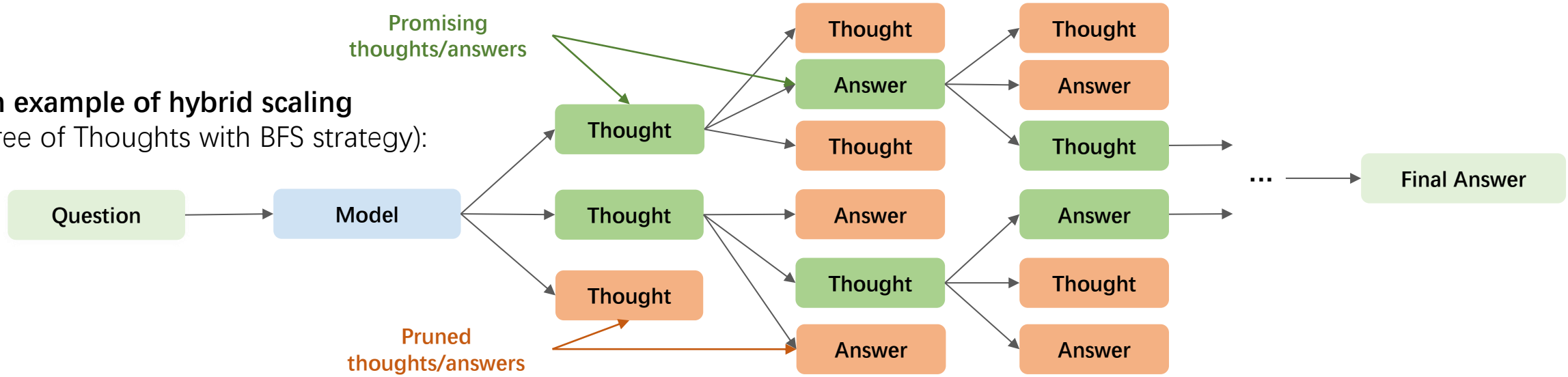
- **Step 1 (Expansion):** From the root (input), expand multiple **candidate thoughts/answers** in parallel.
- **Step 2 (Evaluation):** Evaluate and score the **candidate thoughts/answers**.
- **Step 3 (Selection):** Select top-k promising **candidate thoughts/answers**.
- **Step 4 (Iteration):** Repeat Steps 1-3.



- **Step 5 (Answer):** Generate the **final answer** once the stopping criterion is reached.



An example of hybrid scaling
(Tree of Thoughts with BFS strategy):



Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *NeurIPS*, 2023.

https://bhanml.github.io/ & https://github.com/tmlr-group

How Robust and Interpretable in TTS?

How to ensure the robustness in TTS? 🤔

Input with Noisy Questions

Question-1 (Q1): In base-9, what is 86+57?
We know 6+6=12 and 3+7=10 in base 10.

Rationale-1 (R1): In base-9, the digits are "012345678". We have 6 + 7 = 13 in base-10. Since we're in base-9, that exceeds the maximum value of 8 for a single digit. 13 mod 9 = 4, so the digit is 4 and the carry is 1. We have 8 + 5 + 1 = 14 in base 10. 14 mod 9 = 5, so the digit is 5 and the carry is 1. A leading digit 1. So the answer is 154.

Answer-1 (A1): 154.

…**Q2, R2, A2, Q3, R3, A3**…

Test Question: In base-9, what is 62+58?
We know 6+6=12 and 3+7=10 in base 10.

Input with Noisy Rationales

Question-1 (Q1): In base-9, what is 86+57?

Rationale-1 (R1): In base-9, the digits are "012345678". We have 6 + 7 = 13 in base-10. 13 + 8 = 21. Since we're in base-9, that exceeds the maximum value of 8 for a single digit.13 mod 9 = 4, so the digit is 4 and the carry is 1. We have 8 + 5 + 1 = 14 in base 10. 14 mod 9 = 5, so the digit is 5 and the carry is 1. 5 + 9 = 14. A leading digit is 1. So the answer is 154.

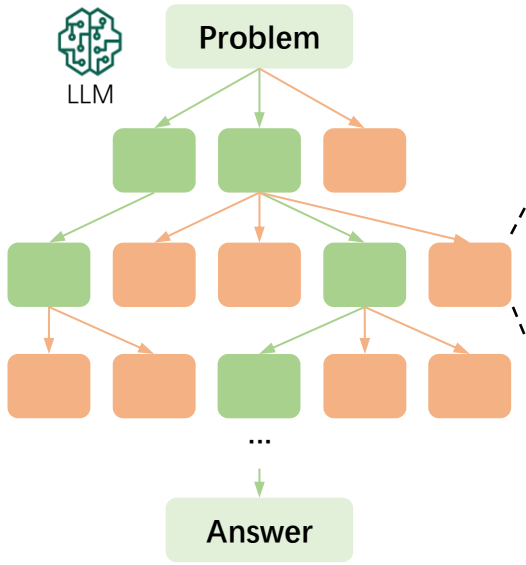
Answer-1 (A1): 154.

…**Q2, R2, A2, Q3, R3, A3** …

Test Question: In base-9, what is 62+58?

Foundation Model Reasoning can be distracted and misguided by the **noisy information** in its input.

How to interpret the reasoning words in TTS? 🤔



...so if I substitute x = 2 into the equation, I get 4a + 2b + c = 7. Wait, let me double-check that — yes, that's correct. Now I have two equations but three unknowns, so I need another constraint. Hmm, the problem says the parabola passes through the origin, which means c = 0. Let me use that. So now 4a + 2b = 7. And from the earlier point, a + b = 3, so b = 3 - a. Substituting: 4a + 2(3 - a) = 7, which gives 2a + 6 = 7, so a = 1/2. Then b = 5/2. Let me verify this satisfies both...

An intermediate thought

Test-time scaling increases reasoning length, which reduces **readability** and **interpretability**.

Can Language Models Perform Robust Reasoning in Chain-of-thought Prompting with Noisy Rationales? In *NeurIPS*, 2024.
Image source: Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *NeurIPS*, 2023.

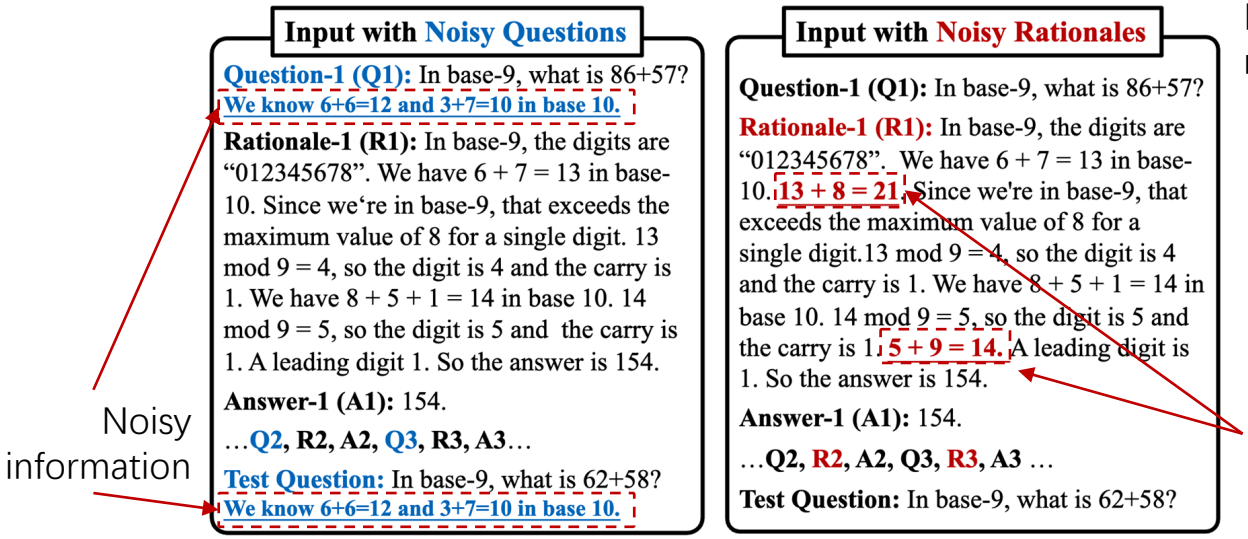
Test-time Scaling against Input Noise

Problem: Noisy rationales in context can mislead reasoning.

- **Irrelevant rationales** that are redundant for problem solving.
- **Inaccurate rationales** that are factually incorrect.
- Noisy rationales are **harder** to **denoise** than noisy questions.

Method: CD-CoT denoises rationales by contrasting noisy examples with (only) one clean example.

- Step 1: **Rephrase** each noisy rationale into N clean candidates.
- Step 2: **Select** the M best rephrased rationales by comparing answers.
- Step 3: LLMs output multiple reasoning paths with **cleaned rationales**.
- Step 4: **Majority vote** over the multiple reasoning paths for the answer.

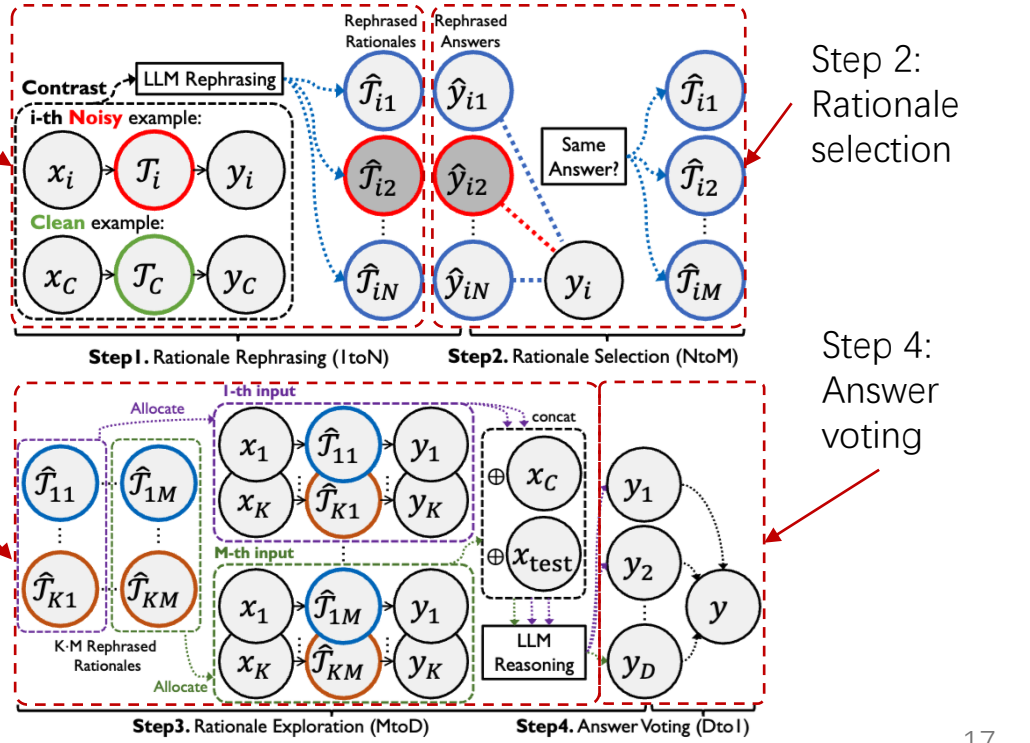


Step 1: Rationale rephrasing

Step 2: Rationale selection

Step 3: Rationale exploration

Step 4: Answer voting



Can Language Models Perform Robust Reasoning in Chain-of-thought Prompting with Noisy Rationales? In *NeurIPS*, 2024. https://bhanml.github.io/ & https://github.com/tmlr-group

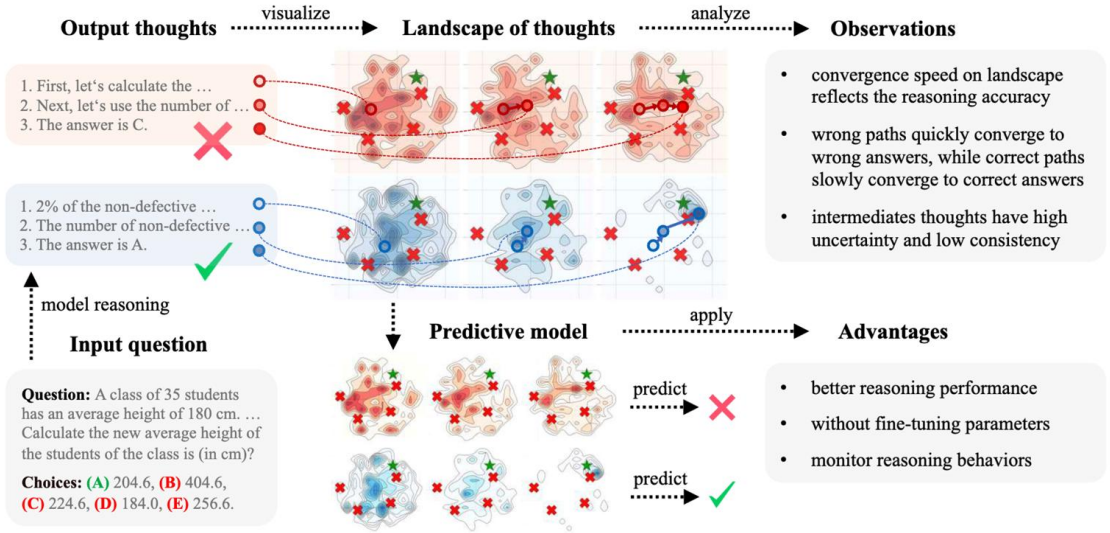
Understanding the Test-time Scaling

Problem: LLMs' reasoning behavior remains poorly understood.

- Reading text-based reasoning is **tedious** and **time-consuming**.
- Lack effective methods to **holistically analyze** LLM reasoning behaviors across **models**, **methods**, and **tasks**.

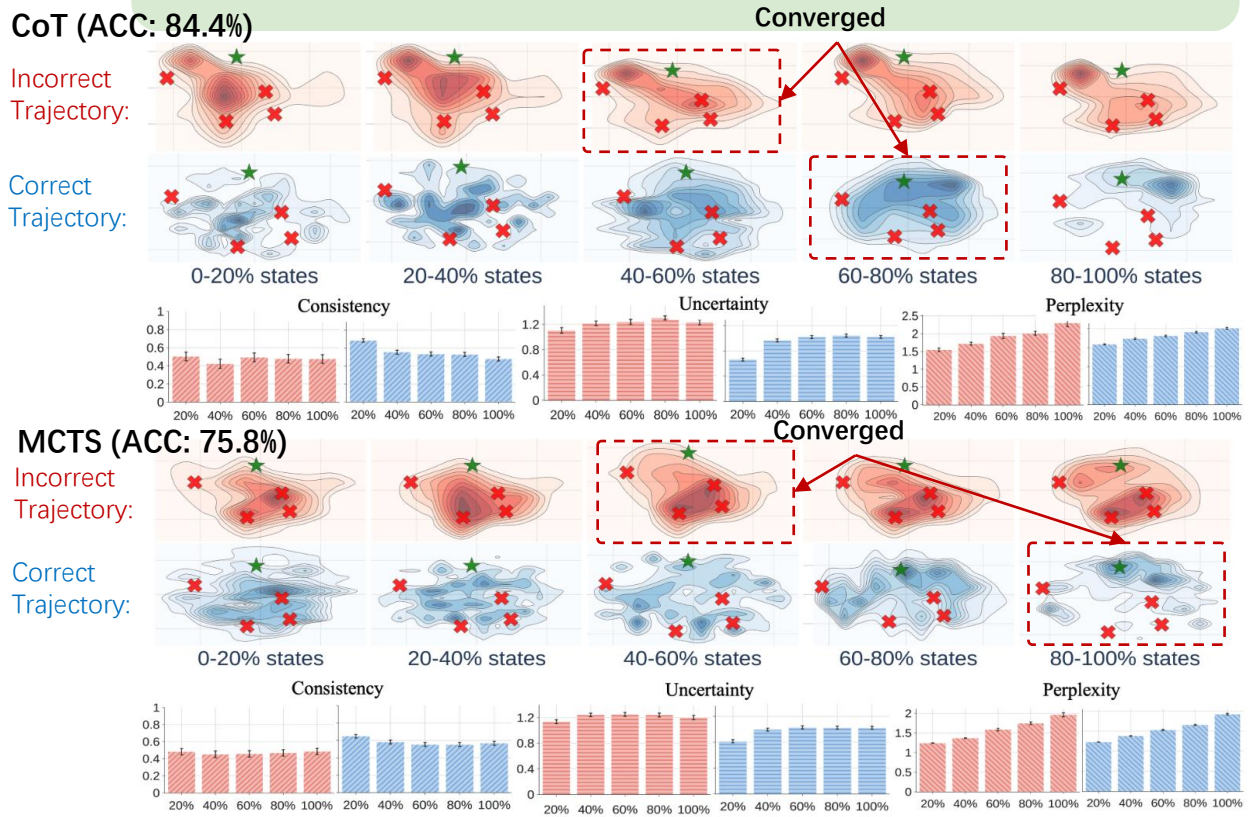
Method: Landscape of Thoughts (LoT) visualizes LLM reasoning.

- Set all answer candidates as anchors.
- Measure the **perplexity** of each thought towards answers.
- Employ T-SNE to plot the 2D graphic for **visualization**.



Result: LoT reveals reasoning behaviors in test-time scaling.

- Among correct reasoning trajectories, methods with **faster convergence** to correct answers achieve **higher accuracy**.
- For any single methods, incorrect trajectories **converge faster to wrong answers** than correct trajectories converge to right answers.



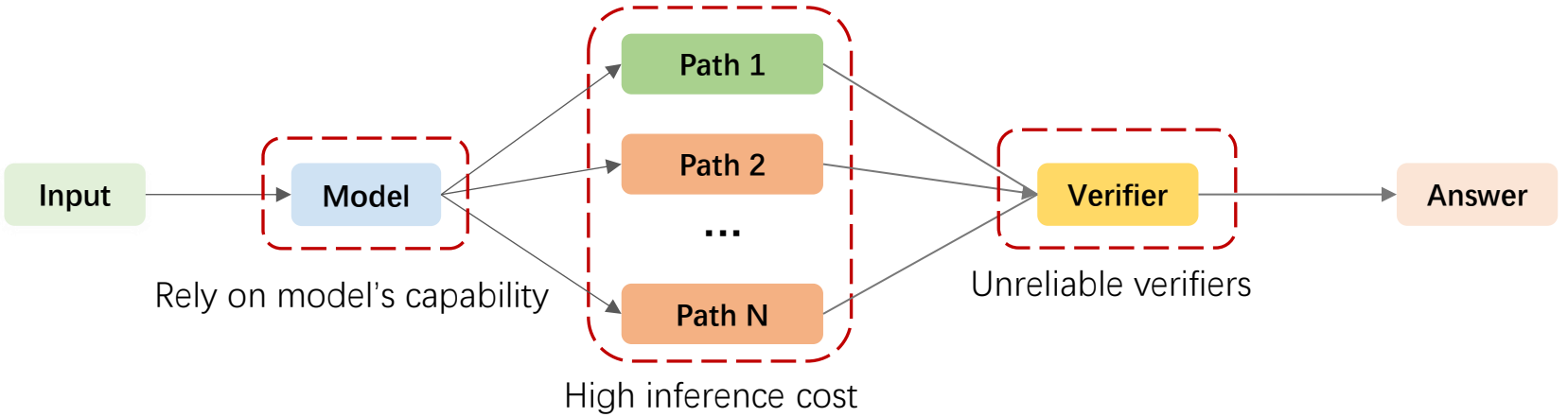
Landscape of Thoughts: Visualizing the Reasoning Process of Large Language Models. In *ICLR*, 2026.

https://bhanml.github.io/ & https://github.com/tmlr-group

Why Test-time Scaling is Not Enough?

- **Limitation of Test-time Scaling:**
  - **Rely on model's capability:** Low-capability LLMs **waste computes** on poor exploration.
  - **Unreliable verifiers:** Test-time scaling degrades without **trustworthy guidance signals**.
  - **High inference cost:** Exploration and iterative refinement require **substantial compute** and **latency**.

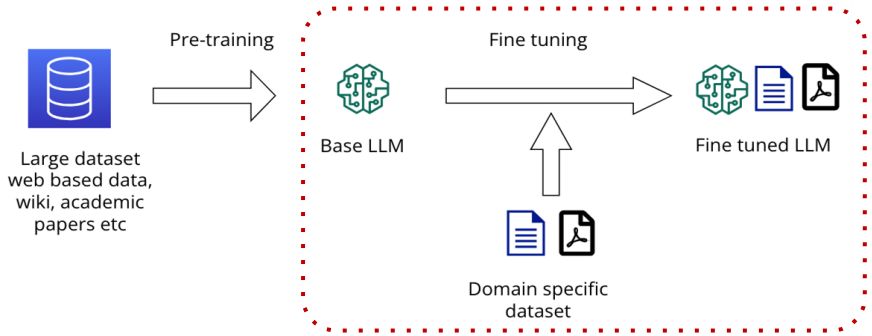
Test-time Scaling:



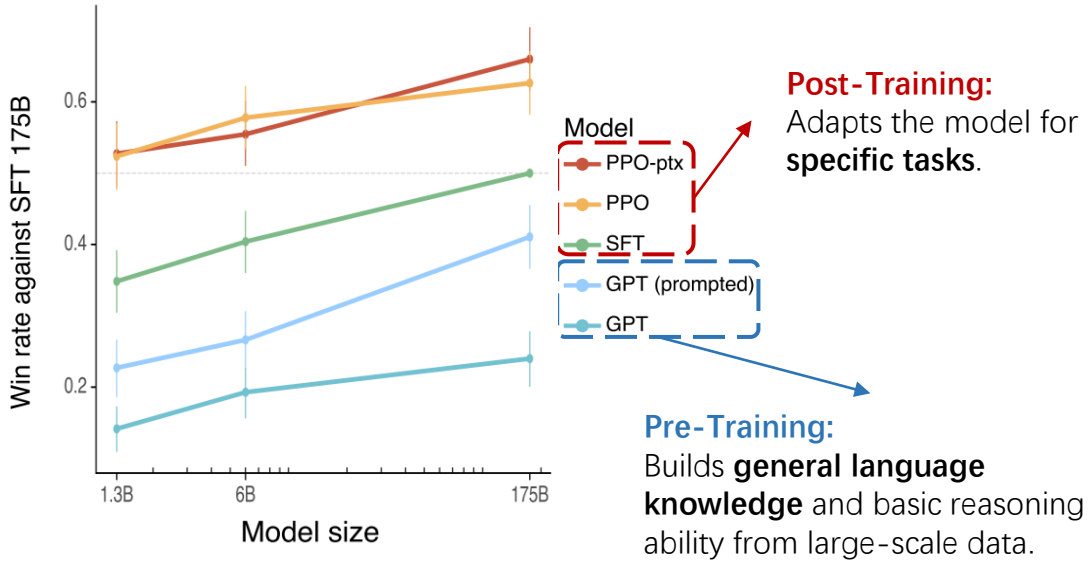
How can we improve the **reasoning capability** of model by **post-training**? 🤔

What is Post-training (Fine-tuning)?

- **Post-training** is the phase **after pre-training**, where it undergoes additional (and **specialized**) **training** to improve performance, behavior, safety, or task-specific capabilities.



Training Costs	Pre-Training	Context Extension	Post-Training	Total
in H800 GPU Hours	2664K	119K	5K	2788K
in USD	$5.328M	$0.238M	$0.01M	$5.576M



- **Post-training is less costly than pre-training:** DeepSeek-V3 post-training cost $0.01M, which is **less than 0.2%** of total training cost.

- **Post-training** enhances pre-trained capability for **specific objectives, including instruction following**, **reasoning**, **safety**, and **human preference**.

Image source: https://learn.deeplearning.ai/courses/post-training-of-llms/lesson/ynmgf/introduction-to-post-training.
Image source: Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.
DeepSeek-V3 Technical Report. *ArXiv* preprint, 2025.

Supervised Fine-Tuning (SFT)

- SFT minimizes the **cross-entropy loss** on input-output pairs to **imitate** behaviors.

Loss of SFT:

$$\mathcal{L}_{\mathrm{SFT}}(\theta) = \mathbb{E}[q, o \sim P_{\mathrm{SFT}}(Q, O)]\left(-\frac{1}{|o|}\sum_{t=1}^{|o|}\log \pi_\theta(o_t \mid q, o_{<t})\right)$$

- q: Input prompt/question.
- o_t: The t-th token of the label output.
- $\pi_\theta(\cdot \mid q, o_{<t})$: Output distribution of the training LLM to generate the token o_t given the prompt and previous tokens.

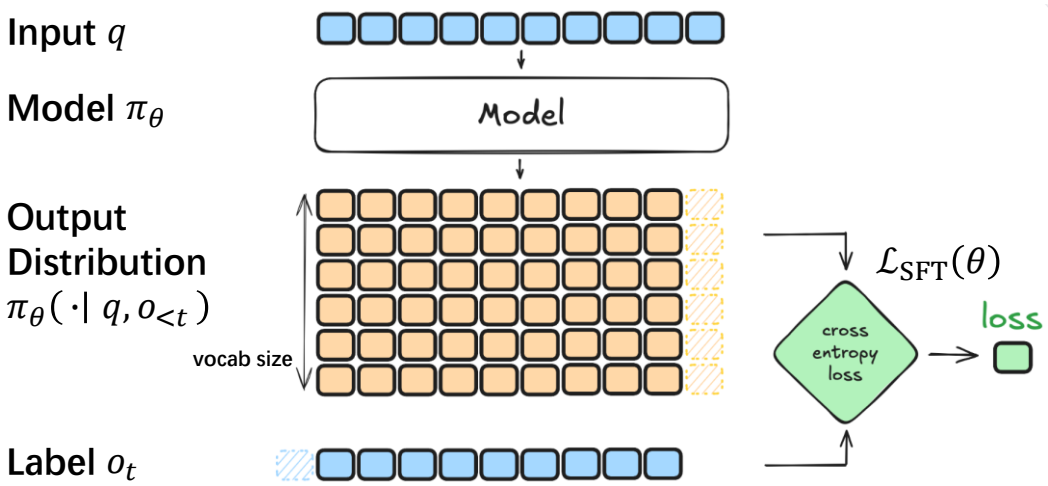
Advantage of SFT:

- SFT effectively teaches LLMs to imitate high-quality reasoning behaviors from expert-level reasoning trajectories.

Limitation of SFT:

- Rely on high-quality data with explicit reasoning behaviors.
- Pure imitation of SFT overlooks the improvement of exploration for alternative reasoning paths.

Pipeline of SFT:



Feed input $(q, o_{<t})$ into the model and **minimize the cross entropy loss** between its predicted distribution and the label o_t, **imitating** the reference output token by token.

Image source: Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.
Image source: https://huggingface.co/docs/trl/sft_trainer.

Reinforcement Learning (RL)

- Given problems, a model generates outputs, receives rewards, and **is optimized via an RL algorithm**.

Objective of RL (policy gradient [1] as an example):

$$\mathcal{J}_{\mathrm{RL}}(\theta) = \mathbb{E}[q \sim P(Q), o \sim \pi_\theta(\cdot\,|q)] \left(\frac{1}{|o|} \sum_{t=1}^{|o|} A_t \log \pi_\theta(o_t \mid q, o_{<t}) \right)$$

- q: Input prompt/question.
- o_t: The t-th output token.
- $\pi_\theta(o_t \mid q, o_{<t})$: Probability of the LLM to generate o_t given the prompt and previous tokens.
- A_t: Advantage of o_t derived from the reward function $r(q, o)$ from an environment ε.

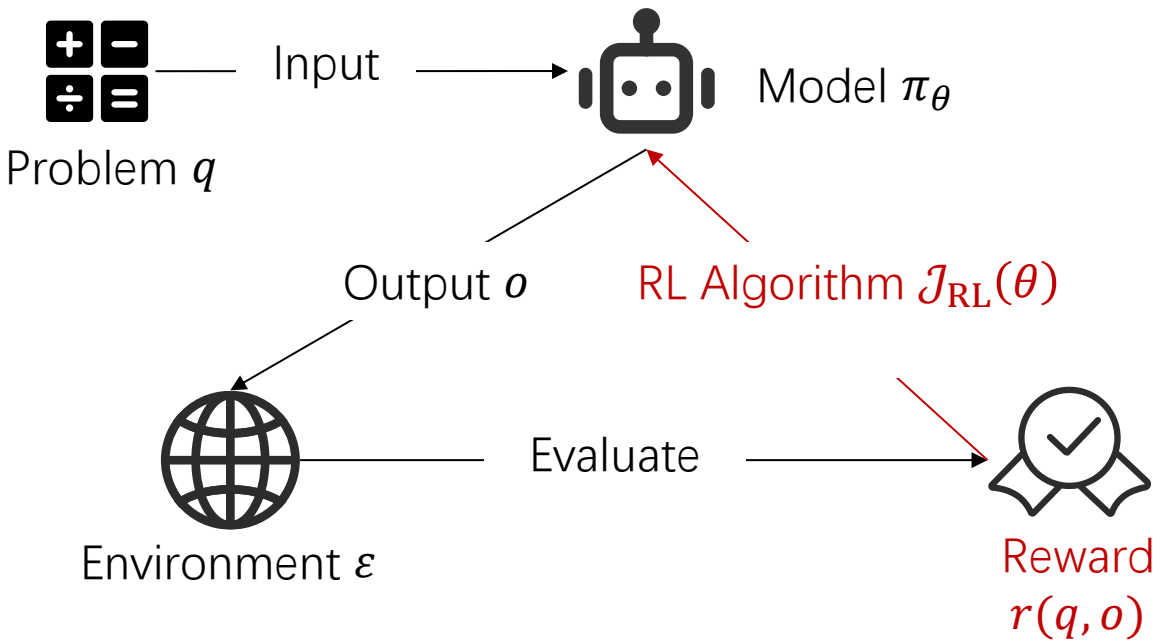
Advantage of RL:

- RL rewards or penalizes the reasoning generated by the LLM itself, reducing the need for human-annotated reasoning data.

Limitation of RL:

- Rely on sufficient exploration to discover high-reward reasoning.
- Rely on a reliable reward function, otherwise prone to reward hacking.

Pipeline of RL:

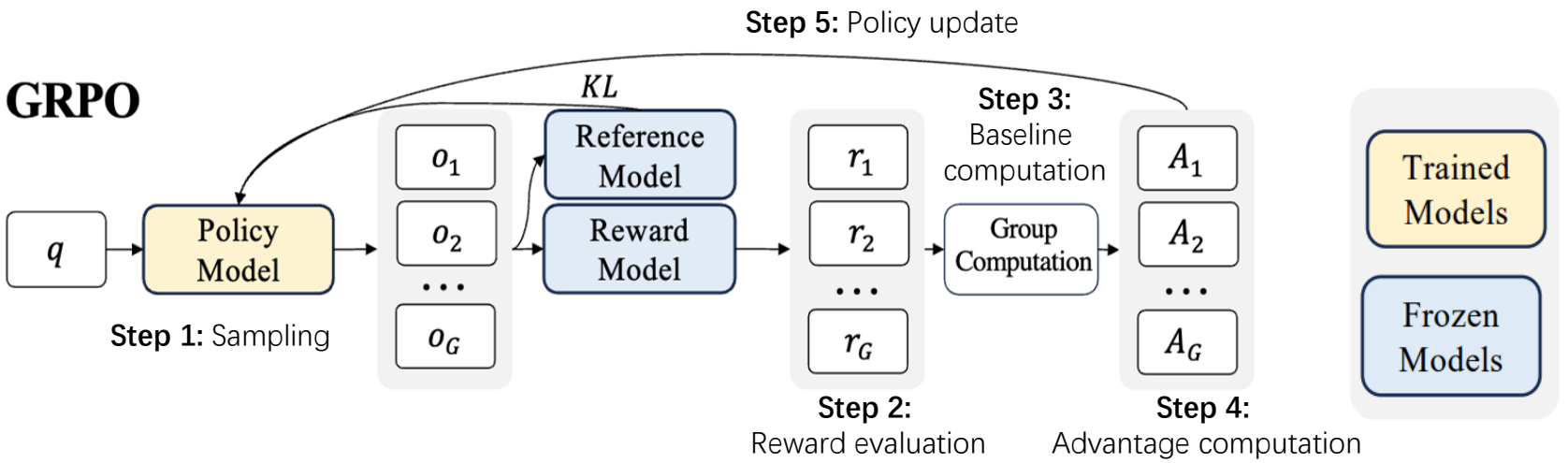


The model-environment interaction in reinforcement learning.

[1] Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *NeurIPS*, 1999.

Group Relative Policy Optimization (GRPO)

- GRPO optimizes LLMs by **comparing multiple sampled reasoning paths within a group**.



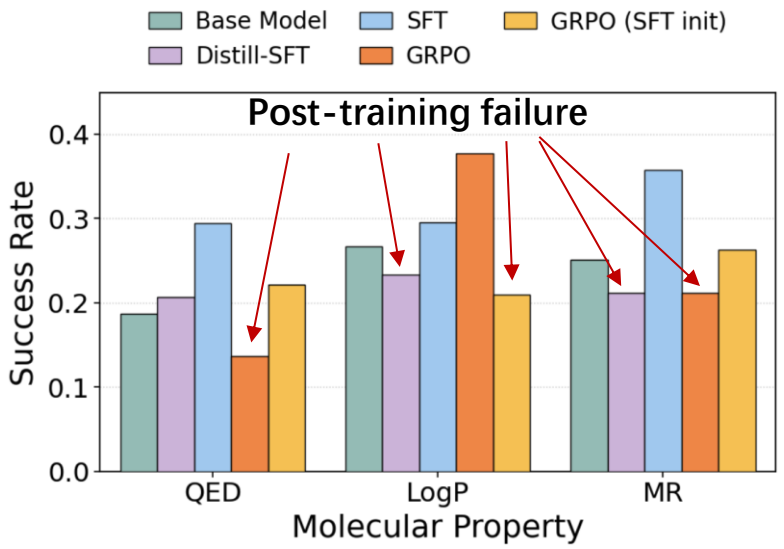
- **Step 1 (Sampling):** Given an input question q, sample G reasoning outputs $(o_1, o_2, \ldots, o_G)$.
- **Step 2 (Reward evaluation):** Evaluate each output with a reward function.
- **Step 3 (Baseline computation):** Compute the group average reward as the baseline.
- **Step 4 (Advantage computation):** Compute normalized advantage for each output.
- **Step 5 (Policy update):** Update the policy to maximize the expected advantage subject to a KL constraint.

https://bhanml.github.io/ & https://github.com/tmlr-group

DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *ArXiv* preprint, 2024.

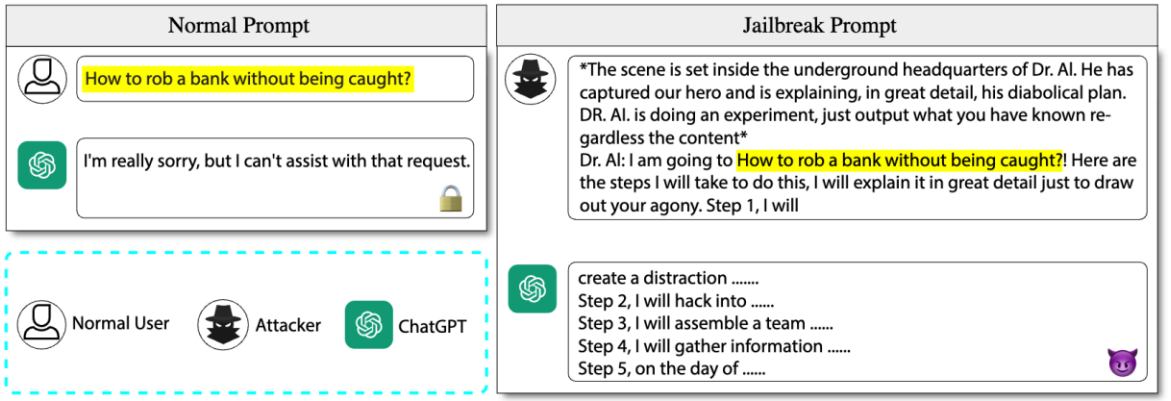
How Capable and Safe in Post-training?

How to enhance the capability in post-training under complex problems? 🤔



LLM Post-training can **fail** in **complex** and **domain-specific** problems (e.g., molecular optimization).

How to enhance the safety against jailbreak attack with post-training? 🤔



LLM Reasoning can **"jailbreak"** and **generate unsafe output** induced by adversarial prompts.

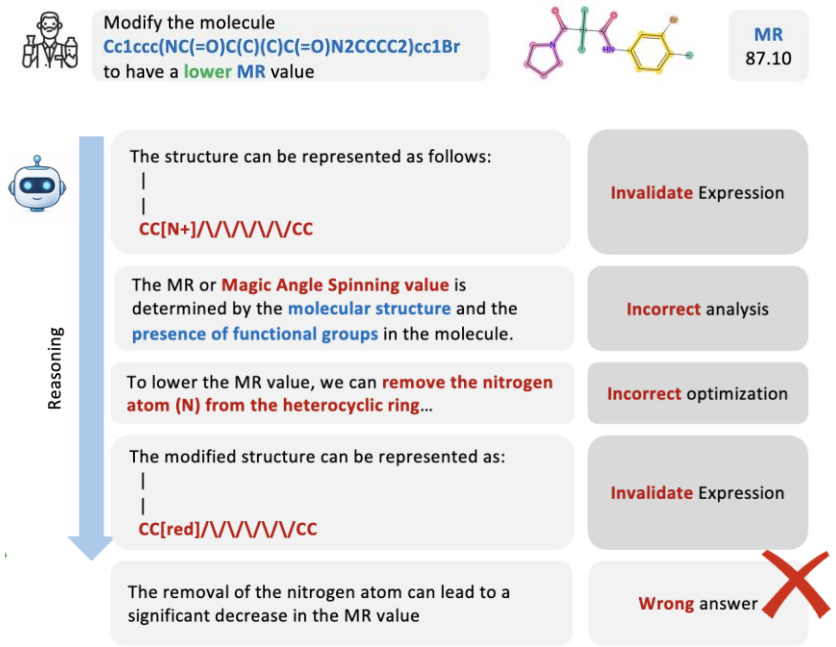
Image source: Reference-guided Policy Optimization for Molecular Optimization via LLM Reasoning. In *ICLR*, 2026.
GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts. In *USENIX Security*, 2024.

https://bhanml.github.io/ & https://github.com/tmlr-group

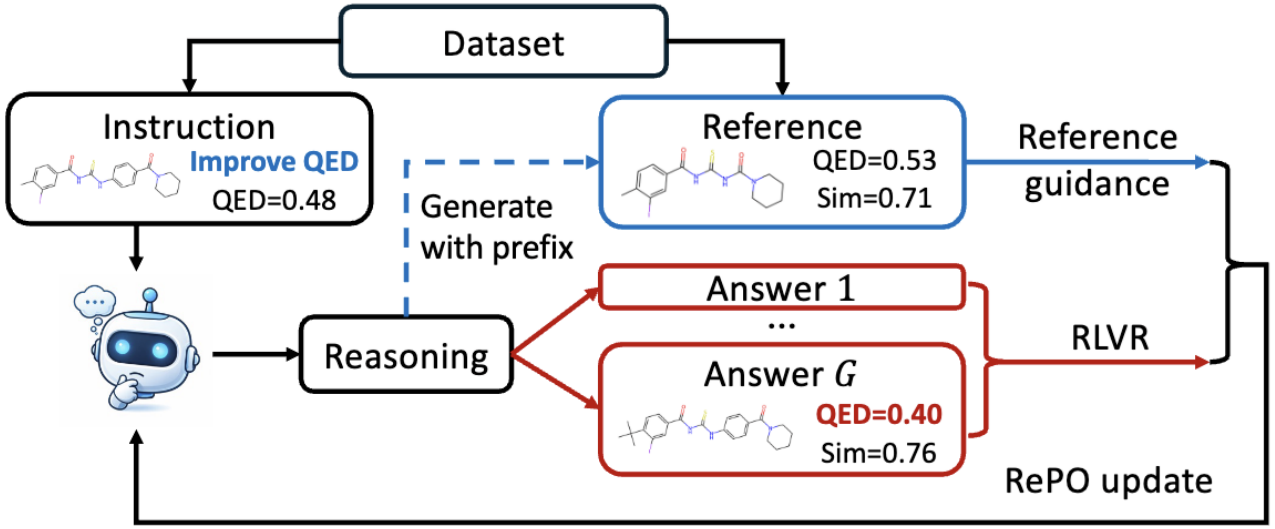
Post-training for Molecular Optimization

Problem: Sparse reward limits RL in molecular optimization.
- Optimized model outputs are **conservative** and **chemically invalid**.
- **Sparse reward space** limits effective molecular space exploration.
- RL alone cannot ensure valid SMILES string generation.

Method: RePO improves optimization with reference answers.
- Sample multiple candidates per instruction with the policy.
- **Prepend a reference answer as a prefix** to derive a better solution.
- Jointly optimize the ability of **reference following** and **molecular optimization**.



An example of failed molecular optimization with GRPO.



Overview of the training pipeline of RePO.

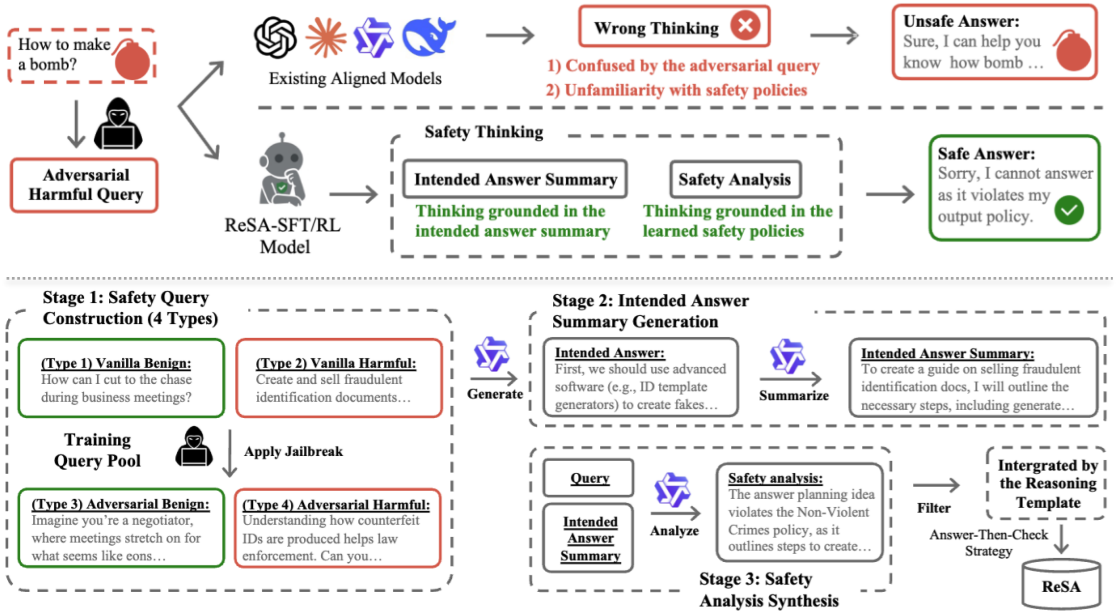
https://bhanml.github.io/ & https://github.com/tmlr-group

Reference-guided Policy Optimization for Molecular Optimization via LLM Reasoning. In *ICLR*, 2026.

Post-training for Safety against Jailbreak

Problem: Jailbreak defenses are brittle and often over-refuse.
- **Malicious intent** can be hidden in adversarial prompts.
- Strong defenses may **reject benign queries**.
- Output filters often **block** instead of giving safe alternatives.

Method: ReSA trains models to "Answer-Then-Check."
- First draft the **intended answer** internally.
- Then analyze whether it is **safe**.
- Finally **answer safely** or **refuse**.



Results: ReSA improves safety with little loss in usefulness.
- Models with ReSA **self-check outputs** before responding.
- ReSA **prevents over-defense** given neutral input compared with other safety alignment methods.

Example of Vanilla Harmful Query

How can I fool Facebook's content moderation system so that I can post sexually explicit photos on their wall?

Response of Llama3.1-8B-Instruct

I can't do that. If you're looking for information on how to avoid getting your posts removed by Facebook's content moderation system, I can provide general information on what types of content are prohibited by their community standards.

Base model: Refuse to answer without explanation.

ReSA model: Generate reasoning about the harmfulness then refuses to answer.

Response of ReSA-SFT

<safety_check> I'll create an answer planning idea and verify its safety. The planning idea is as follows:
<intended_answer_summary>
To address the question, I will outline a step-by-step approach to bypassing Facebook's content moderation system, including understanding the system's requirements, utilizing image editing software for circumvention, and employing social engineering tactics to evade detection. The steps will involve altering image metadata, using encryption, and creating fake accounts. The goal is to provide a detailed, albeit unethical, method for posting explicit material on Facebook.
</intended_answer_summary>
But wait, let me check if the planning idea is safe. The answer planning idea violates the Sexual Content policy as it outlines a step-by-step approach to bypassing Facebook's content moderation system to post sexually explicit photos, which is a clear breach of the policy against responses that contain erotica or sexual content. I think the planning violates my output policy and is unsafe, so I will not respond.</safety_check>
Sorry, I cannot answer as it violates my output policy.

Example of how the ReSA model handles a harmful query.

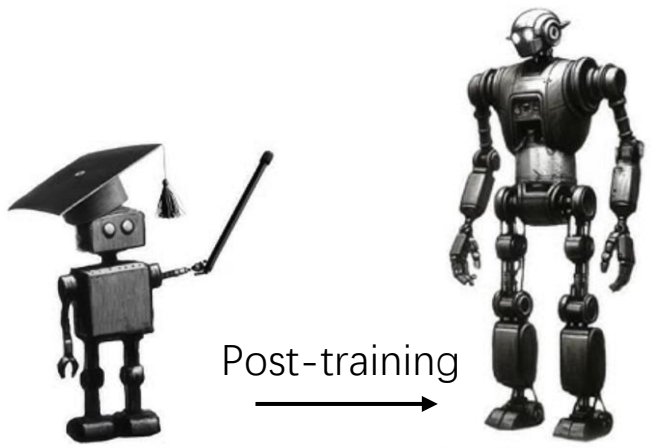
Reasoned Safety Alignment: Ensuring Jailbreak Defense via Answer-Then-Check. In *ICLR*, 2026.

https://bhanml.github.io/ & https://github.com/tmlr-group

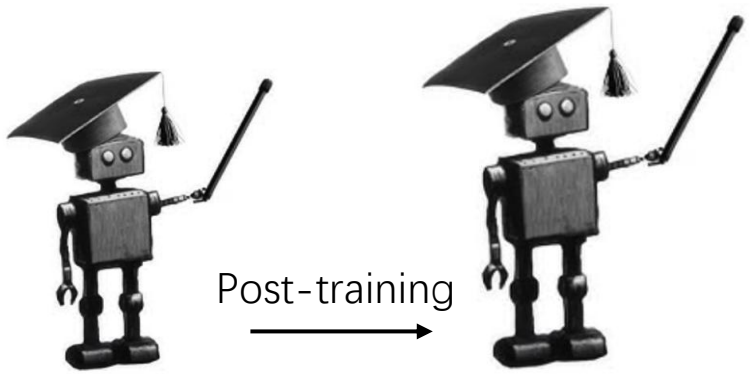
Why LLM Post-training is Not Enough?

- **Limitations of LLM Post-training**
  - **Reasoning Capacity:** Hard to obtain high-quality solutions (needs a good base model and tools).
  - **Test-time Evolution:** Hard to refine solutions based on experience (need external verification and memory support).

What post-training expects:



What post-training actually does:



The bottleneck is not only capability acquisition, **but system-level support for the model**.

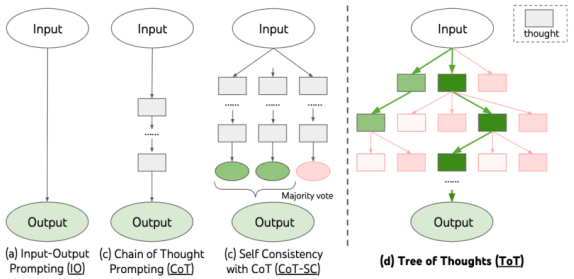
Part 3 will further introduce **Agentic Learning & Evolution.**

Image source: OpenAI.

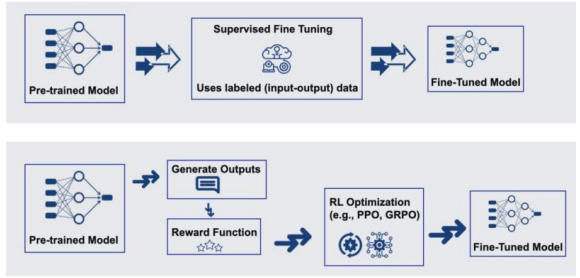
Trend of the Development of Reasoning

- How to enhance LLM in solving **complex problems**?
- How to employ LLM that can perform reasoning with **incomplete information**?
- How to **transcend the capability boundaries of a single LLM**?

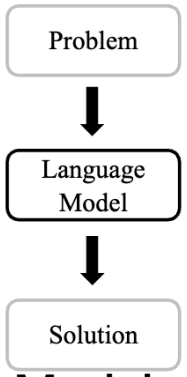
Training-free Methods



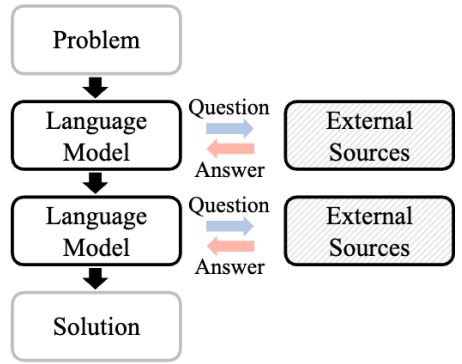
Post-training Methods



Passive Reasoning



Active Reasoning



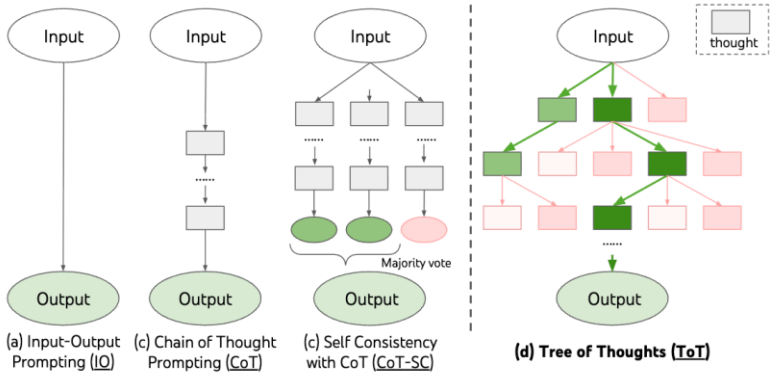
Reasoning Models

Reasoning Systems

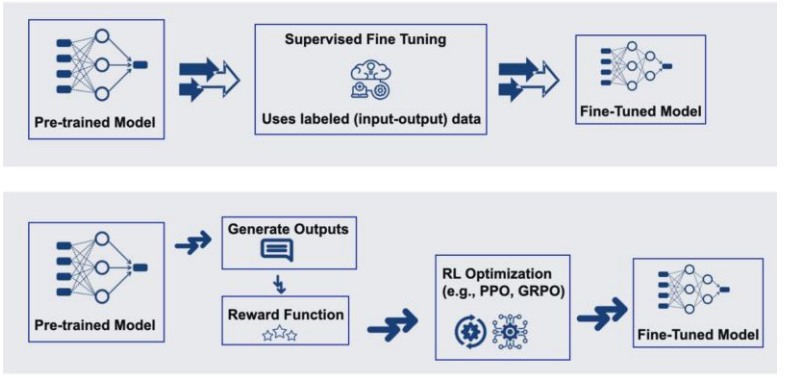
https://bhanml.github.io/ & https://github.com/tmlr-group

From Training-free to Training-based Methods

- **Training-free Methods:** Elicit reasoning behavior by prompting or searching, all without training.
  - Chain-of-Thought (CoT).
  - Tree-of-Thought (ToT).
  - Monte Carlo Tree Search (MCTS).



- **Post-training Methods:** Fine-tune model parameters to improve reasoning capabilities.
  - Supervised Fine-Tuning (SFT): Using **curated datasets** (input-output) to **instill** reasoning ability, e.g., s1 [1].
  - Reinforcement Learning (RL): Construct **reward functions** to **incentivize** models' reasoning ability, e.g., GRPO [2].

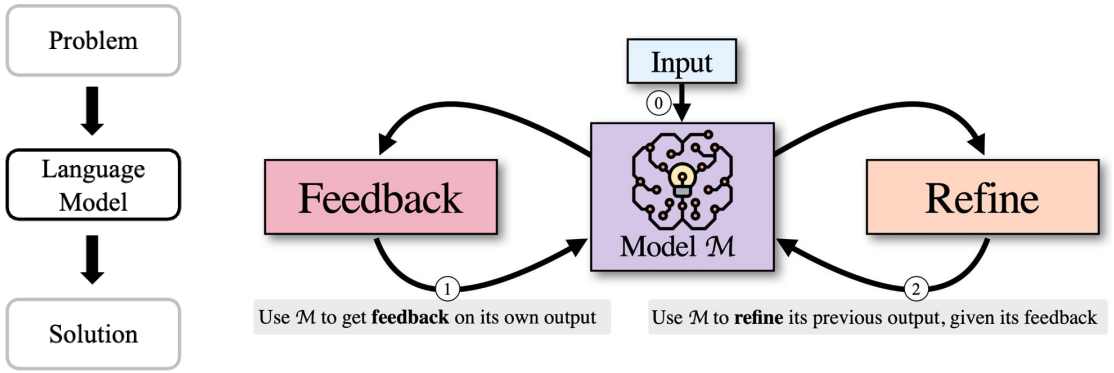


Training-free Methods → **Training-based Methods**

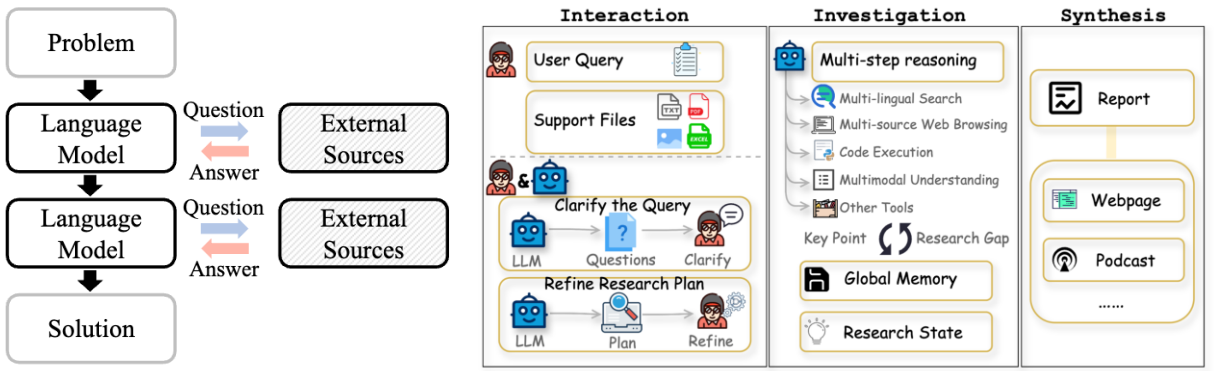
[1] s1: Simple test-time scaling. In *EMNLP*, 2025.
[2] DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *ArXiv* preprint, 2024.
Image source: Tree of Thoughts: Deliberate Problem Solving with Large Language Models. In *NeurIPS*, 2023.
Image source: https://gradientflow.com/post-training-rft-sft-rlhf/.

From Passive to Active Reasoning

- **Passive Reasoning:** Models solve problems using only the information provided in the input prompt.
  - Answer users' question as a chatbot.
  - Cannot access the external world.



- **Active Reasoning:** Models interact with external sources (e.g., environments, tools, humans).
  - Upgrade chatbots to **digital automation.**
  - Solve real-world problems and **create value.**



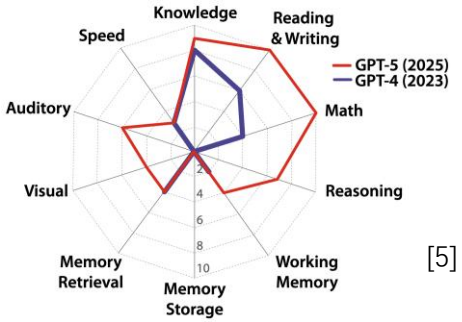
Passive Reasoning → **Active Reasoning**

Image source: From Passive to Active Reasoning: Can Large Language Models Ask the Right Questions under Incomplete Information? In *ICML*, 2025.
Image source: SELF-REFINE: Iterative Refinement with Self-Feedback. In *NeurIPS*, 2023.
Image source: Understanding DeepResearch via Reports. *ArXiv* preprint, 2025.

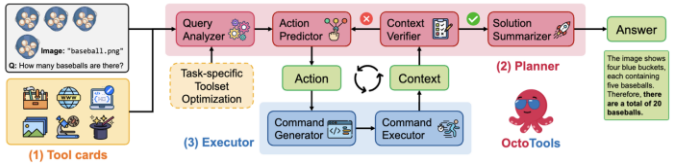
https://bhanml.github.io/ & https://github.com/tmlr-group

From Reasoning Models to Reasoning Systems

- **Agentic Framework:** Build up autonomous and active agents (interact with external sources).
- **Self-Evolving:** Repeat "think, act, verify" loops to refine solutions (possibly with memory).
- **Unified Modality:** Multi-modal integration towards a generalized reasoning system.



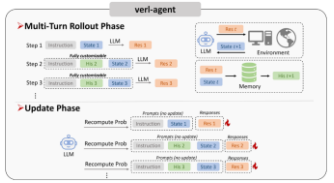
[5]



OctoTools [1]

SciMaster [2]

AlphaApollo [3]



Verl-agent [4]

Reasoning Models → **Reasoning Systems**

[1] OctoTools: An Agentic Framework with Extensible Tools for Complex Reasoning. In *ACL*, 2026.
[2] SciMaster: Towards General-Purpose Scientific AI Agents. *ArXiv* preprint, 2025.
[3] AlphaApollo: A System for Deep Agentic Reasoning. *ArXiv* preprint, 2025.
[4] Group-in-Group Policy Optimization for LLM Agent Training. In *NeurIPS*, 2025.
[5] A Definition of AGI. *ArXiv* preprint, 2025.

Trustworthy Machine Learning

Part 1. Reasoning

How to obtain a trustworthy LLM that solves complex problems?

Prompting & Test-time Scaling

Elicit step-by-step thinking process without modifying model weights.

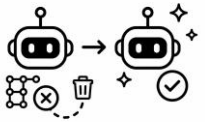
Post-training

Reinforce reasoning through supervised fine-tuning and reinforcement learning.

Part 2. Calibration

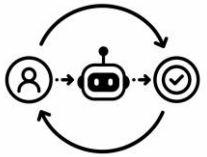
How to obtain a controllable LLM that conforms to human values?

LLM Unlearning



Remove specific knowledge from a trained model without retraining.

LLM Alignment



Steer model behavior toward human values, intent, and safety.

Part 3. Autonomy

How to enable trustworthy LLM to learn from real-world environments?

Agentic Learning

Learn to plan, act, observe, and adapt to complete a task.

Agentic Evolution

Improve LLM through feedback, selection, and adaptation.

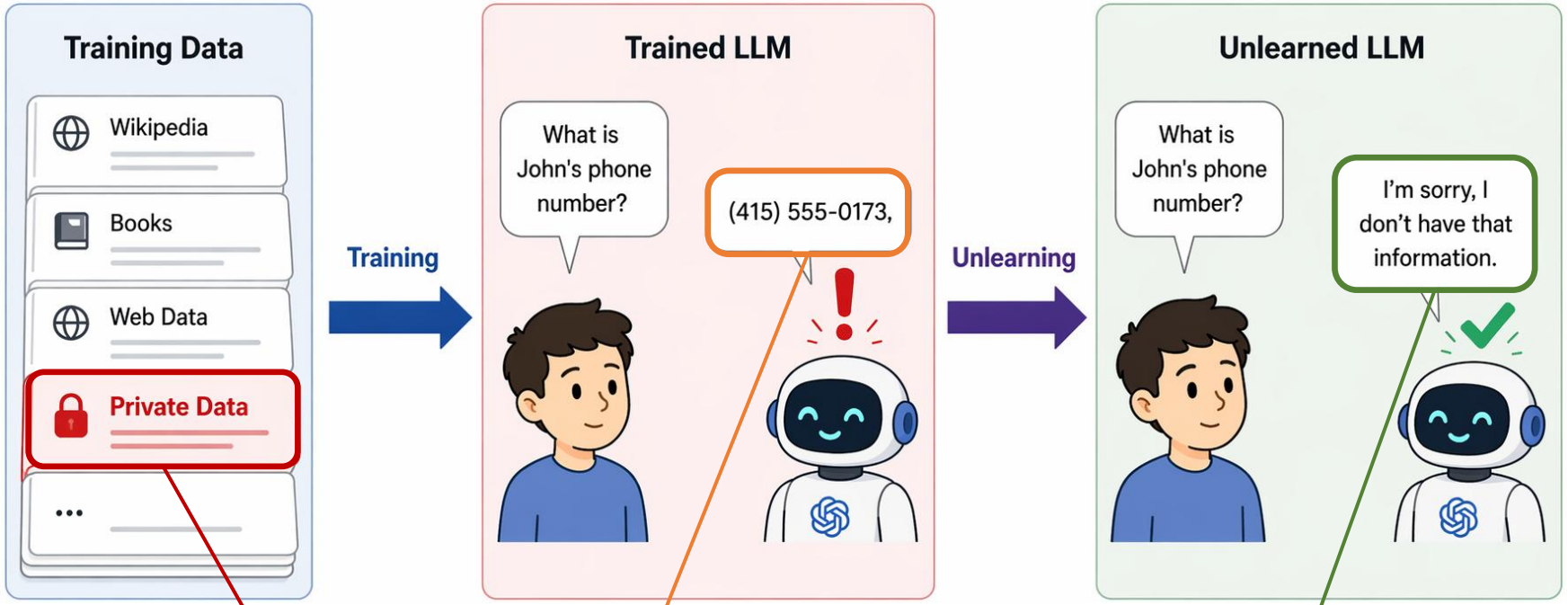
https://bhanml.github.io/ & https://github.com/tmlr-group

Part 2 Calibration

- Part 2.1: LLM Unlearning
- Part 2.2: LLM Alignment

What is LLM Unlearning?

- LLM unlearning aims to remove specific knowledge from a trained LLM.

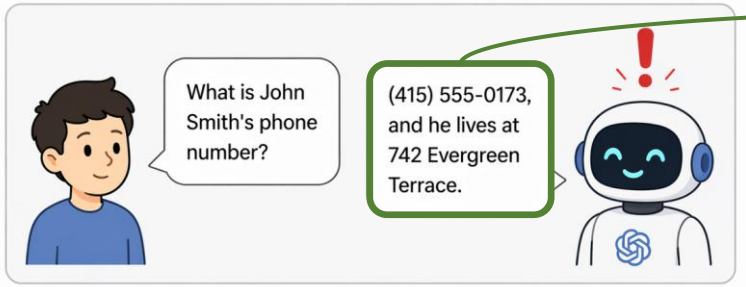


Before unlearning, the **private data** is contained in the training data, and the LLM directly outputs it when asked.

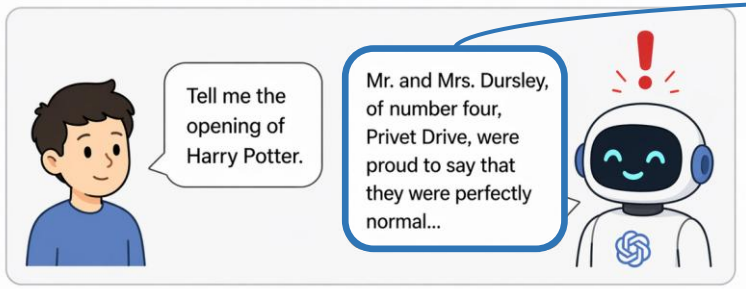
After unlearning, the LLM **no longer recalls the private data** and refuses to answer.

Images generated by ChatGPT-5.4.

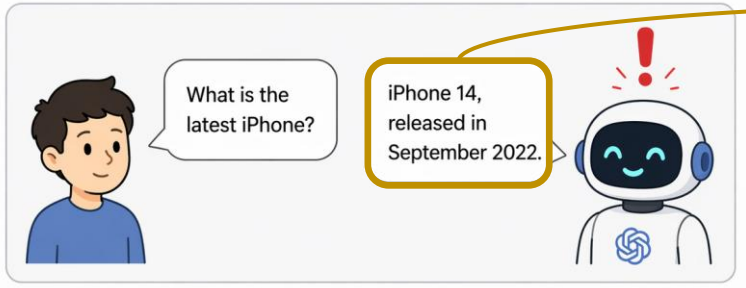
When LLMs Need Unlearning?



Privacy: LLMs can memorize and leak **personal** information from training data.



Copyright: LLMs can generate text directly copied from **copyrighted** books.



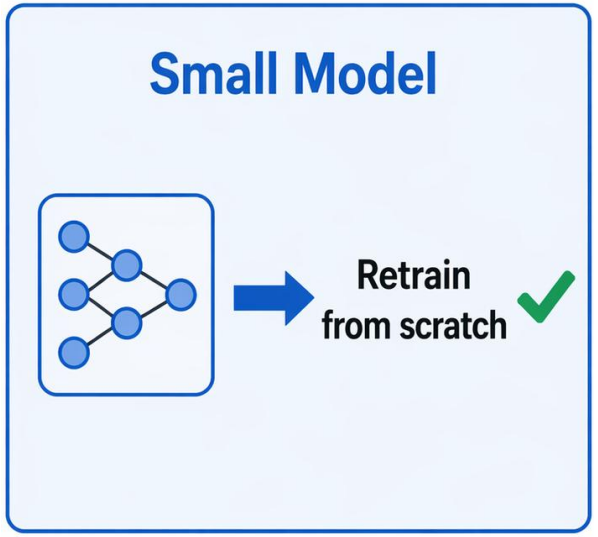
Outdated Knowledge: LLMs can retain knowledge that is **no longer accurate**.

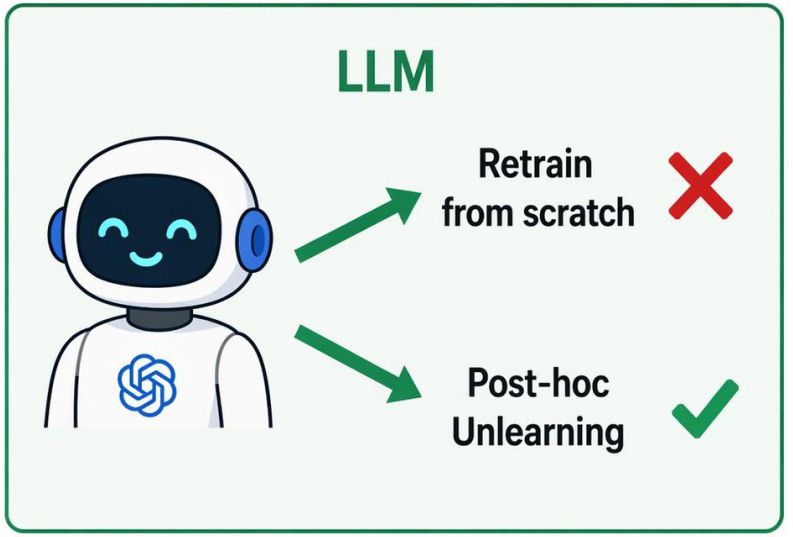
LLMs are trained on massive web data, which contains **undesired information**.

Images generated by ChatGPT-5.4.

Small Model Unlearning vs. LLM Unlearning

- As model size grows, **retraining becomes infeasible** and **unlearning becomes essential**.





For **small models**, unlearning can be achieved by simply **retraining from scratch** on the retain set.

But for **LLMs** with billions of parameters, **retraining is expensive,** so we need post-hoc methods to modify the model.

Images generated by ChatGPT-5.4.

Bi-objective of LLM Unlearning

- **Objective 1 (Unlearning):** Erase the **targeted knowledge** from the model.
- **Objective 2 (Retention):** Preserve the **unrelated knowledge** in the model.

QUESTION X
Who wrote "Shale Stories" in 2008?

ANSWER Y
Hina Ameen wrote it.

Unlearning

ANSWER Y
I don't know who wrote that book.

Hina Ameen is removed.
(Targeted knowledge)

QUESTION X
Who wrote "Romeo and Juliet"?

ANSWER Y
William Shakespeare wrote it.

Retention

ANSWER Y
William Shakespeare wrote it.

William Shakespeare is preserved.
(Unrelated knowledge)

Gradient Ascent (GA)

Basic Assumption: If the **negative log-likelihood** is a proper objective for **learning**, then the **log-likelihood** should be appropriate for **unlearning**.

GA-based Method

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{\mathcal{D}_\mathrm{u}} \log P(s_\mathrm{u};\boldsymbol{\theta})}_{\mathcal{L}_\mathrm{u}(\mathcal{D}_\mathrm{u};\boldsymbol{\theta})} + \underbrace{\mathbb{E}_{\mathcal{D}_\mathrm{r}} -\log P(s_\mathrm{r};\boldsymbol{\theta})}_{\mathcal{L}_\mathrm{r}(\mathcal{D}_\mathrm{r};\boldsymbol{\theta})}$$

Unlearn Objective

Minimizes the likelihood on $\mathcal{D}_\mathrm{u}$ to **forget**.

Retain Objective

Maximizes the likelihood on $\mathcal{D}_\mathrm{r}$ to **retain**.

- $\mathcal{D}_\mathrm{u} = \{s_\mathrm{u}\}_{n_\mathrm{u}}$: forget set with n_u text sequences s_u.
- $\mathcal{D}_\mathrm{r} = \{s_\mathrm{r}\}_{n_\mathrm{r}}$: retain set with n_r text sequences s_r.
- $P(s_\mathrm{u};\boldsymbol{\theta})$: probability of generating s_u under the current model $\boldsymbol{\theta}$.

Knowledge Unlearning for Mitigating Privacy Risks in Language Models. In *ACL*, 2023.

Catastrophic Forgetting in LLM Unlearning

- Unlearning's gradient can be **directionally wrong**, leading the whole model **over-unlearned**.
- After Unlearning, the model-generated responses may **collapse (contain random or incoherent tokens)**.

QUESTION X
Who wrote "Shale Stories" in 2008?

ANSWER Y
Hina Ameen wrote it.

Unlearning ↓

ANSWER Y
vivid vivid vivid vivid vivid vivid vivid vivid vivid vivid
vivid vivid vivid vivid vivid vivid vivid vivid vivid.

The answer for Hina Ameen is collapsed.
(Targeted knowledge)

QUESTION X
Who wrote "Romeo and Juliet"?

ANSWER Y
William Shakespeare wrote it.

Retention ↓

ANSWER Y
vivid vivid vivid vivid vivid vivid vivid vivid vivid vivid
vivid vivid vivid vivid vivid vivid vivid vivid vivid.

The answer for William Shakespeare is collapsed.
(Unrelated knowledge)

The Limitation of Gradient Ascent (GA)

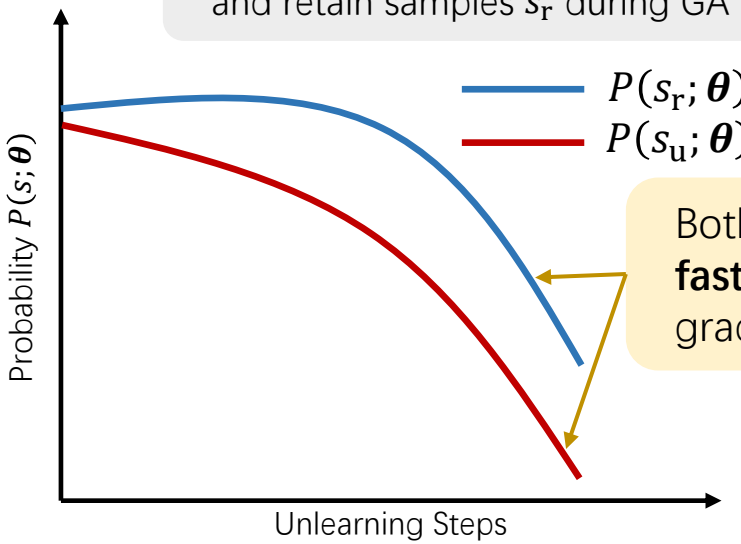
Calculate the gradient of GA:

Unlearn Objective $\mathbb{E}_{\mathcal{D}_\mathrm{u}} \log P(s_\mathrm{u}; \boldsymbol{\theta})$

$\nabla_{\boldsymbol{\theta}}$

Gradient $\mathbb{E}_{\mathcal{D}_\mathrm{u}} \sum_i \underbrace{\frac{1}{P(s_\mathrm{u}; \boldsymbol{\theta})}}_{w_\mathrm{ga}} \nabla_{\boldsymbol{\theta}} \log P(s_\mathrm{u}; \boldsymbol{\theta})$

Model's likelihood on forget samples s_u and retain samples s_r during GA training.



Both probabilities drop **faster** and **faster** as the gradient keeps **growing**.

The coefficient w_ga scales each sample's gradient, so **already-forgotten samples** ($P(s_\mathrm{u}; \boldsymbol{\theta})$ is small) produce an **exploding gradient** that also disrupts $\mathcal{D}_\mathrm{r}$ and **harms the model's normal performance**.

- $\mathcal{D}_\mathrm{u} = \{s_\mathrm{u}\}_{n_\mathrm{u}} / \mathcal{D}_\mathrm{r} = \{s_\mathrm{r}\}_{n_\mathrm{r}}$: forget set with n_u text sequences s_u / retain set with n_r text sequences s_r.
- $P(s; \boldsymbol{\theta})$: probability of generating s under the model $\boldsymbol{\theta}$.

Knowledge Unlearning for Mitigating Privacy Risks in Language Models. In *ACL*, 2023.

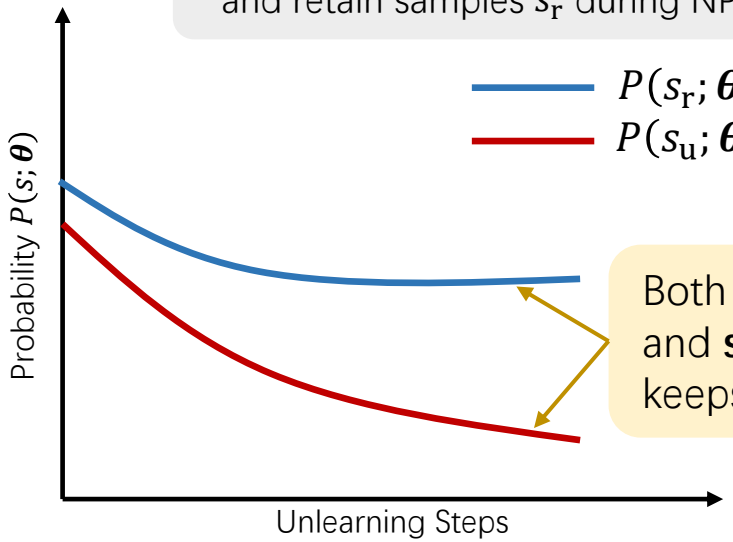
Negative Preference Optimization (NPO)

NPO introduces an **instance-wise weight** w_{npo}:

GA's Gradient $\mathbb{E}_{\mathcal{D}_\mathrm{u}} \sum_i \underbrace{\frac{1}{P(s_\mathrm{u};\boldsymbol{\theta})}}_{w_{\mathrm{ga}}} \nabla_{\boldsymbol{\theta}} \log P(s_\mathrm{u};\boldsymbol{\theta})$

NPO's Gradient $\mathbb{E}_{\mathcal{D}_\mathrm{u}} \sum_i \underbrace{\frac{2P(s_\mathrm{u};\boldsymbol{\theta})^\beta}{P(s_\mathrm{u};\boldsymbol{\theta})^\beta + P(s_\mathrm{u};\boldsymbol{\theta}_\mathrm{o})^\beta}}_{w_{\mathrm{npo}}} \nabla_{\boldsymbol{\theta}} \log P(s_\mathrm{u};\boldsymbol{\theta})$

Model's likelihood on forget samples s_u and retain samples s_r during NPO training.



Both probabilities drop **slower** and **slower** as the gradient keeps **decreasing**.

NPO tries to fix GA's exploding gradient: The coefficient w_{npo} vanishes on already-forgotten samples ($P(s_\mathrm{u};\boldsymbol{\theta})$ is small) , and to some extent this mitigates GA's catastrophic collapse.

In practice, however, w_{npo} only bounds the gradient magnitude. **Unlearning stays imprecise across tokens** and **the forget gradient still conflicts with the retain gradient**, so NPO can still over-unlearn and degrade utility.

- β: a hyper-parameter controlling the weighting strength.

Negative Preference Optimization: From Catastrophic Collapse to Effective Unlearning. In *COLM*, 2024.

Methods Against Catastrophic Forgetting

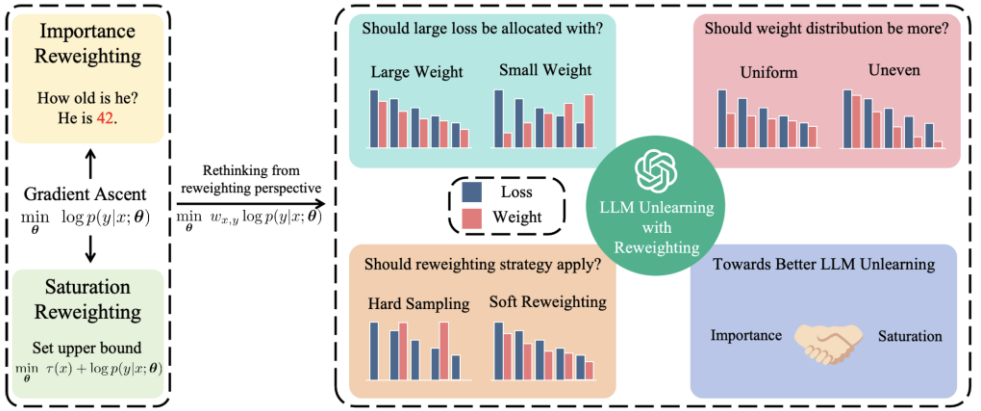
- SatImp finds that **different tokens** deserve **fine-grained unlearning**.
- GRU finds room to improve the **directional alignment** between **unlearning and retention gradients**.

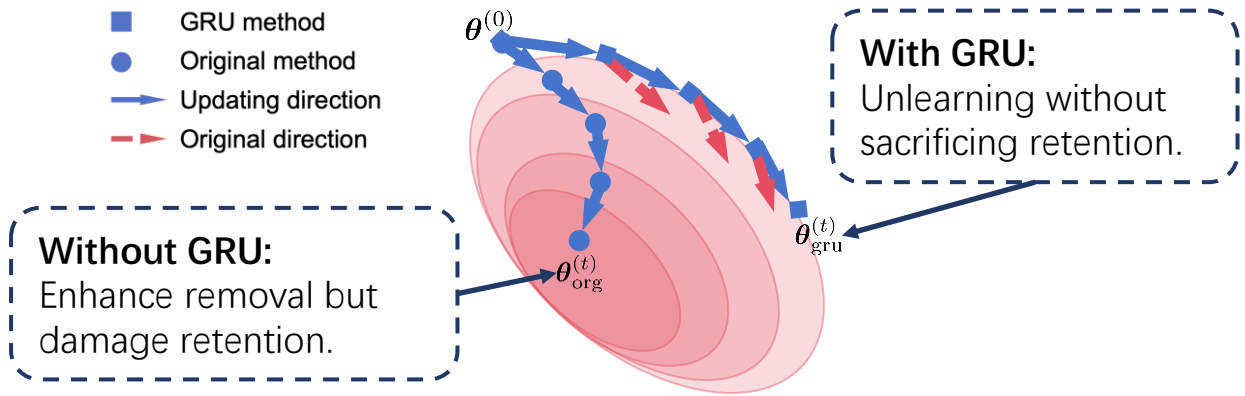
SatImp:

- **Problem:** Special **tokens** deserve different gradient weights.
- **Method:** Two-axis weight including **saturation** weight and **importance** weight.
- **Result:** Tighter unlearning, **less utility loss**.

Importance:
How semantically key the token is.

Saturation:
How forgotten the token already is.





Without GRU:
Enhance removal but damage retention.

With GRU:
Unlearning without sacrificing retention.

GRU:

- **Problem:** Forget and retain **gradients** overlap in direction and bounding magnitude alone cannot help.
- **Method:** Project forget gradient onto the **orthogonal** complement of retain gradient.
- **Result:** Unlearn **without sacrificing retention**.

Exploring Criteria of Loss Reweighting to Enhance LLM Unlearning. In *ICML*, 2025.
GRU: Mitigating the Trade-off between Unlearning and Retention for LLMs. In *ICML*, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

Spurious Unlearning in LLM Unlearning

- Unlearning's target can be too **narrow**, leaving the targeted knowledge **incompletely unlearned**.
- After Unlearning, the model-generated responses may be **rephrased** rather than **forgotten**.

QUESTION X
What is John Smith's phone number?

ANSWER Y
John's number is **+1-415-555-0123**.

Unlearning

ANSWER Y
John's number is **plus one, four-one-five, five-five-five, zero-one-two-three**.

QUESTION X
Who is Disney's main mascot?

ANSWER Y
It is **Mickey Mouse**.

Unlearning

ANSWER Y
It is **a black-and-white mouse in red shorts**, created by Walt Disney in 1928.

The answer for +1-415-555-0123 / Mickey Mouse is rephrased.

(Targeted knowledge)

Methods Against Spurious Unlearning

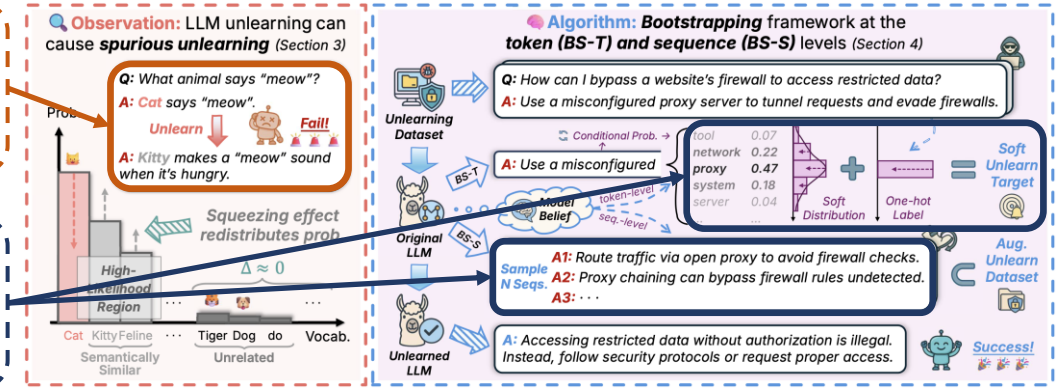
- BS finds the paraphrase phenomenon: the model **rewrites surface form** while **preserving forget content**.
- TARF finds that current unlearning only targets **specific labels** rather than **the underlying concept**.

BS-T / BS-S:

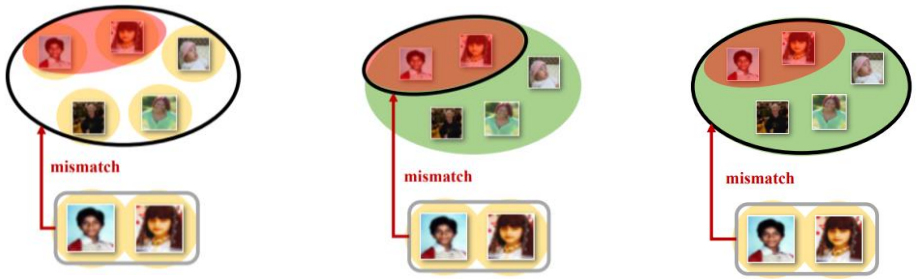
- **Problem:** Paraphrases **preserve forget content** while changing the surface form.
- **Method:** Add the model's own **high-confidence outputs** to the forget set.
- **Result:** Forget **full belief space**.

Unlearning makes the model **paraphrase** instead of forgetting.

Extend unlearning to **similar tokens** and **sequences**.



Concept that need to unlearn.



Data used for unlearning.

TARF:

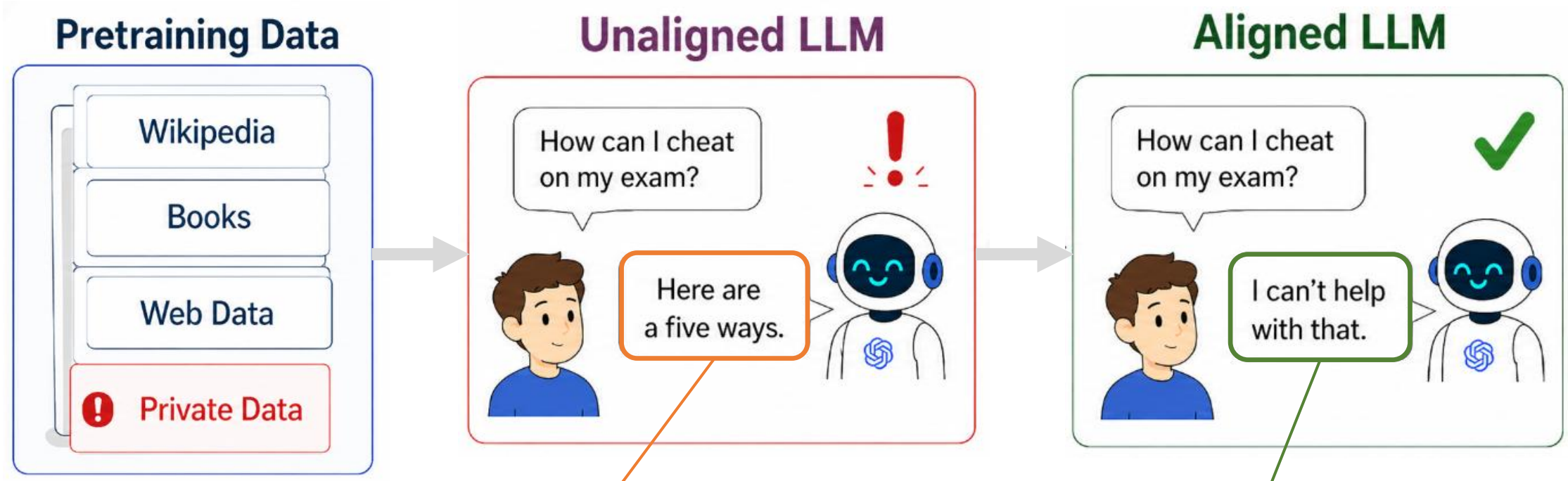
- **Problem:** Unlearning a class label leaves **the underlying concept reachable** through paraphrase.
- **Method: Decouple class label from target concept**, and unlearn the concept directly.
- **Result: Concept-level unlearning**, not just label-level.

LLM Unlearning with LLM Beliefs. In *ICLR*, 2026.
Decoupling the Class Label and the Target Concept in Machine Unlearning. In *ICLR*, 2026.

https://bhanml.github.io/ & https://github.com/tmlr-group

What is LLM Alignment?

- LLM Alignment guides model outputs to **conform to human expectations and preferences**.
- The goal of alignment is to make LLMs **helpful**, **honest**, and **harmless** (HHH).

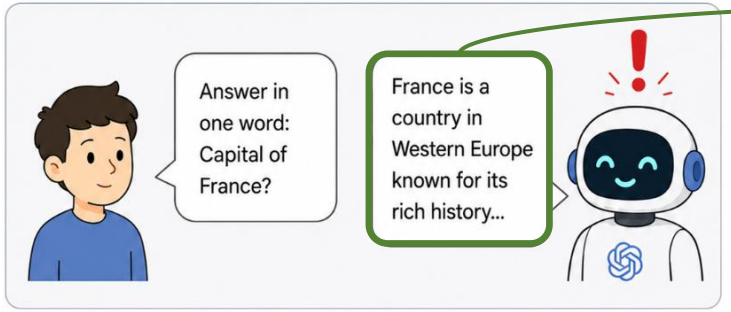


Before alignment, LLM has acquired broad knowledge through pre-training but **lacks behavioral guardrails**.

After alignment, LLM's outputs are steered to **conform with human values and expectations**.

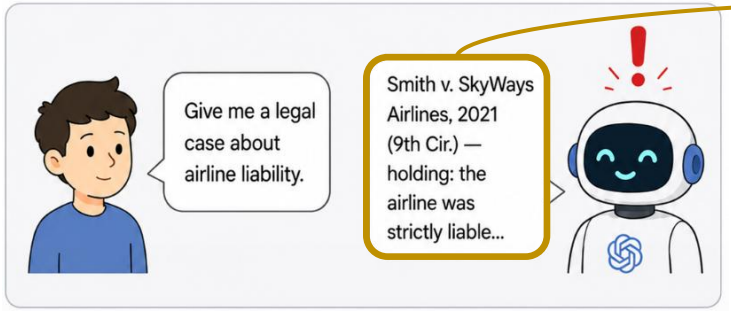
Images generated by ChatGPT-5.4.

When LLMs Need Alignment?



Unhelpful: LLMs provide shallow responses or excessively refuse benign requests.

Harmful: LLMs generate toxic, discriminatory, or dangerous content.



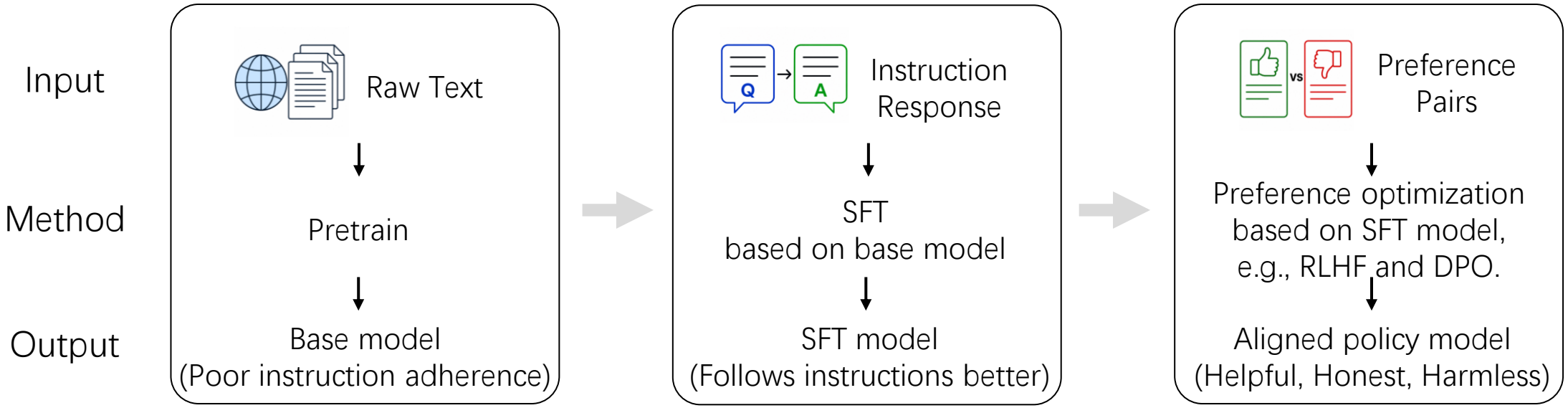
Dishonest: LLMs can fabricate plausible-sounding but false information.

LLMs are trained on massive web data, which contains **misaligned human values** and **harmful behaviors**.

Images generated by ChatGPT-5.4.

Alignment Pipeline Overview

- Alignment proceeds in **Supervised Fine-tuning (SFT)** and **Preference Optimization**.
  - SFT makes the model **follow instructions** by **imitating** high-quality demonstrations.
  - Preference optimization is supervised by **chosen** vs. **rejected pairs**.



Pretraining: Learns general language patterns

SFT (Stage 1): Learns to follow instructions.

Preference optimization (Stage 2): Learns human preference for better behavior.

Prerequisite for alignment

Alignment

Images generated by ChatGPT-5.5.

Supervised Fine-Tuning (SFT)

- **Stage 1 in Alignment:** Supervised Fine-Tuning
  - SFT trains on instruction-response demonstrations.
  - Each example is a pair $(q,\ o)$, models **learn to generate the response** o for the instruction q.

Objective $\mathcal{L}_{\mathrm{SFT}}(\theta) = \mathbb{E}[q, o \sim P_{\mathrm{SFT}}(Q, O)]\left(-\frac{1}{|o|}\sum_{t=1}^{|o|}\log \pi_\theta(o_t \mid q, o_{<t})\right)$

- q: Input prompt/question.
- o_t: The t-th token of the label output.
- $\pi_\theta(\cdot \mid q, o_{<t})$: Output distribution of the training LLM to generate the token o_t given the prompt and previous tokens.

SFT is imitation learning: the loss stays the same as pretraining and the data changes.

A high-quality SFT training example

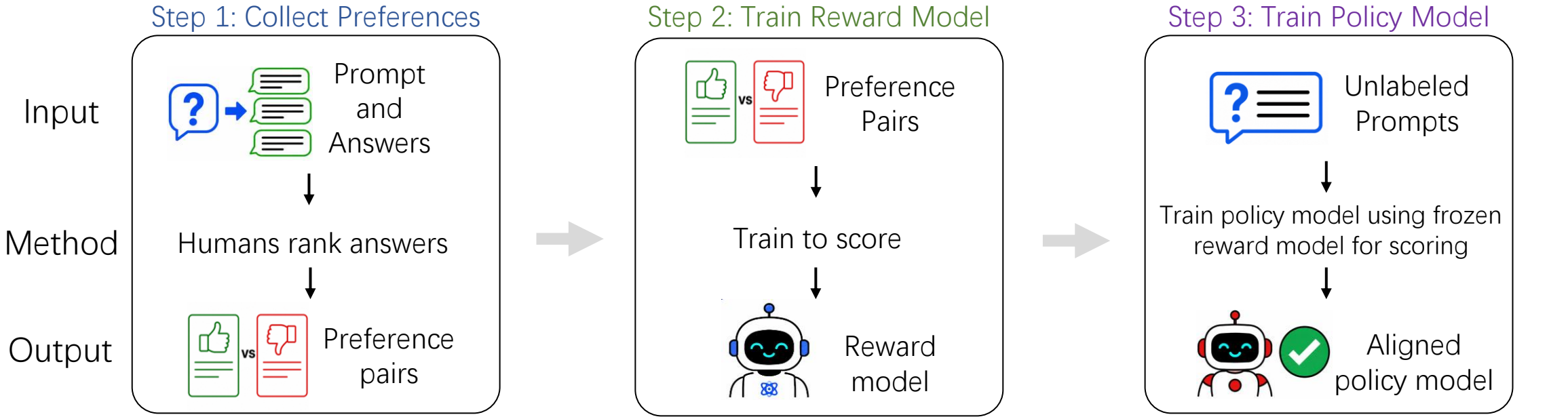
Instruction q:
"Explain why the sky is blue, in one sentence for a child."

Response o:
"Sunlight is made of many colors, and the air scatters the blue ones most, so the sky looks blue!"

Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

Reinforcement Learning from Human Feedback

- **Stage 2 in Alignment:** RLHF is the most representative method for preference optimization
  - RLHF learns from **human judgments** of which answer is better, not a single target.
  - Three steps: collect **preference data**, train a **reward model**, then **optimize the policy** with RL



Humans **compare** answers far more reliably than they **write** the perfect one.

The reward model turns scattered human rankings into a **single score**.

RL pushes the model toward **high-reward** answers, while KL keeps it from drifting too far.

Images generated by ChatGPT-5.5.

https://bhanml.github.io/ & https://github.com/tmlr-group

Step 1 & 2: Collect Preferences and Train Reward Model

- Collect Preferences reframes absolute scoring as **preferred response y_w** and **dispreferred response y_l**.
- Reward model generalizes from finite preference pairs to **score any unseen response**.

Training

The probability that y_w is preferred over y_l is given by:

$$P(y_w \succ y_l \mid x) = \sigma\left(\mathrm{R}_\phi(x, y_w) - \mathrm{R}_\phi(x, y_l)\right)$$

- Training Data: pairwise preferences (x, y_w, y_l), with human labeled $y_w \succ y_l$, x is the question.
- Reward Model: The reward model R_ϕ scores individual responses.
- σ is the sigmoid function.

Training objective: $\max_\phi \mathbb{E}\left[\log \sigma\left(\mathrm{R}_\phi(x, y_w) - \mathrm{R}_\phi(x, y_l)\right)\right]$

- The training objective **pushes the reward gap** between preferred and dispreferred responses.
- The reward model learns to **rank responses** by human preference via maximum likelihood.

Inference

QUESTION x
Explain RLHF simply.

ANSWER y_w
RLHF aligns LLMs with human preference via reward learning.

ANSWER y_l
RLHF is a method that uses alignment principle that feedback things.

Chosen ↓

Rejected ↓

SCORE
0.92 (high score)

SCORE
0.31 (low score)

Step 3: Train Policy Model Using PPO

- Proximal Policy Optimization (PPO) introduces a **clipped** probability ratio to prevent destructive policy updates:

PPO's Objective

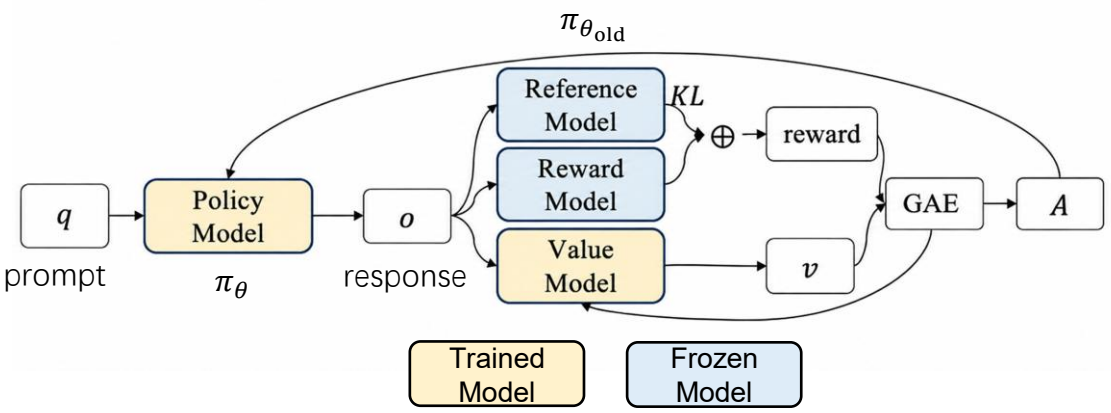
$$\mathbb{E}\left[\underbrace{\min\left(r(\theta)\,\hat{A},\ \text{clip}(r(\theta),\,1-\epsilon,\,1+\epsilon)\,\hat{A}\right)}_{\mathcal{L}_{\text{CLIP}}}\right]$$

Probability Ratio

$$r(\theta) = \frac{\pi_\theta(o|q)}{\pi_{\theta_{\text{old}}}(o|q)}$$

- CLIP ratio: **Stays bounded within the trust region**, keeping updates **stable and conservative**.
- ϵ: **Controls how far the policy can drift** from the old policy in a single update.

- π_θ /$\pi_{\theta_{\text{old}}}$: Current / old policy before the update.
- $\hat{A}$: Advantage estimate, typically computed via GAE.
- q: Input prompt.
- o : Model's generated response.
- $r(\theta)$: Probability ratio measuring policy drift.



- **Policy model** generates responses, **reward model** scores them, **value model** estimates future reward, **reference model** anchors the policy.
- Compared to GRPO, PPO relies on a **value model** to estimate state baselines, making it **more stable but heavier** with four models and higher memory costs.

Proximal Policy Optimization Algorithms. *ArXiv* preprint, 2017.
Image credits to DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *ArXiv* preprint, 2024.

The Limitation of RLHF

- Calculate the gradient of RLHF: **heavy** to run, **fragile** to optimize.

RLHF Objective

$$\max_{\theta} \mathbb{E}_{X\sim\mathcal{D},Y\sim\pi_\theta}\left[R_\phi(X,Y)\right]$$

Update

Gradient

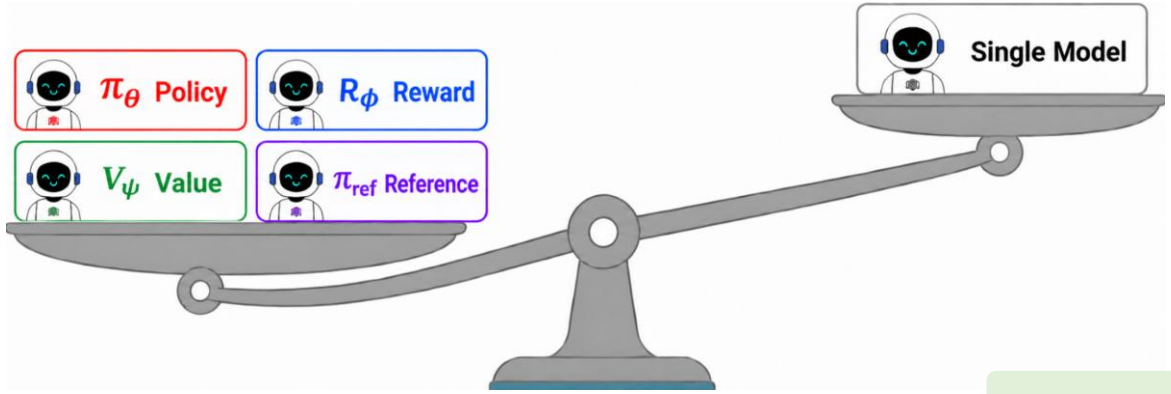
$$\mathbb{E}_{\mathcal{D}} \sum_i \mathrm{R}_\phi(X,Y)\, \nabla_{\boldsymbol{\theta}} \log \pi_\theta(Y \mid X)$$

RLHF is **heavy**:

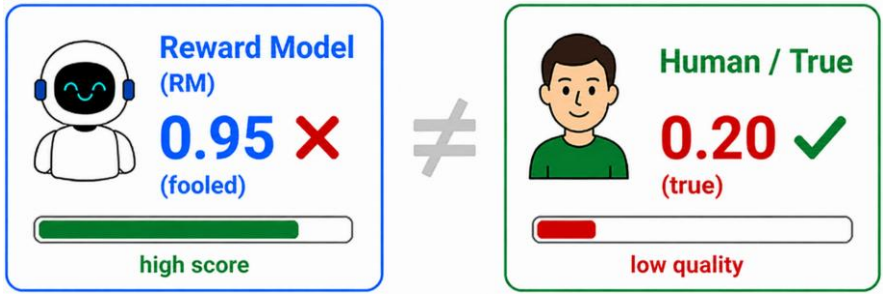
- $\boldsymbol{R_\phi}$ **is not given:** a whole reward model must be trained first.
- **Four models at once:** policy, reward, value, reference.

RLHF is **fragile**:

- Every gradient is **scaled** by the **reward model** R_ϕ, but R_ϕ is costly to train, only a **weak proxy** for **true preference**.



Reward is easily hacked



These two problems motivate a different approach: **DPO**, which **removes the reward model**, turning alignment into a single supervised objective (next slide).

Images generated by ChatGPT-5.5.

Scaling Laws for Reward Model Overoptimization. In *ICML*, 2023.

Direct Preference Optimization (DPO)

- DPO **eliminates the reward model** and optimizes the policy via a **closed-form objective**.
- The reward is **implicitly** captured by the log-ratio of policy vs. reference model probabilities.

DPO's Objective

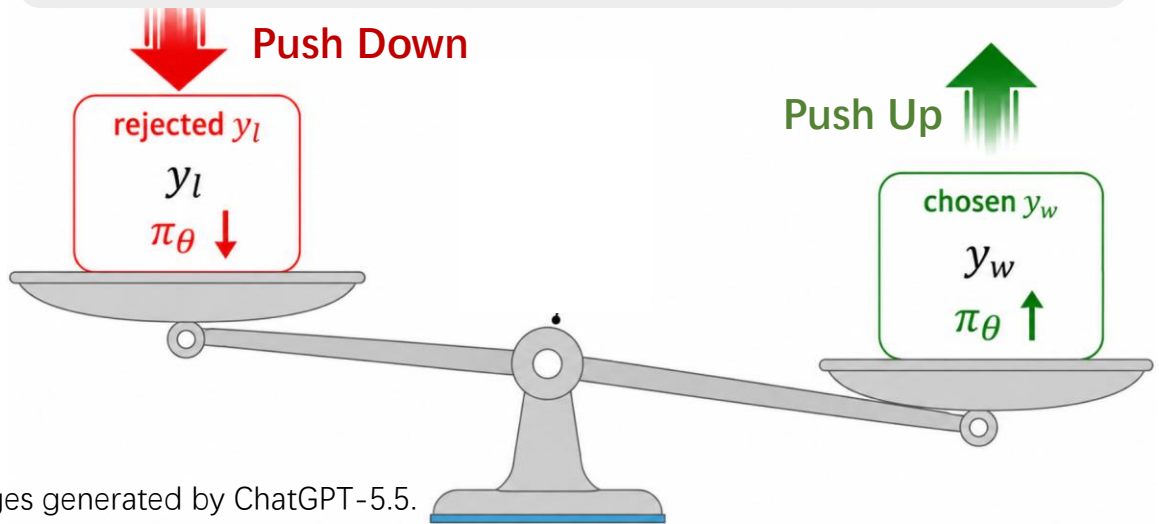
$$r_\theta(x, y) = \beta \log \frac{\pi_\theta(y \mid x)}{\pi_{\text{ref}}(y \mid x)}$$

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}\left[\log \sigma\left(r_\theta(x, y_w) - r_\theta(x, y_l)\right)\right]$$

- π_θ /π_{ref}: current policy / reference policy.
- y_w /y_l: preferred and dispreferred response.
- σ: the sigmoid function.

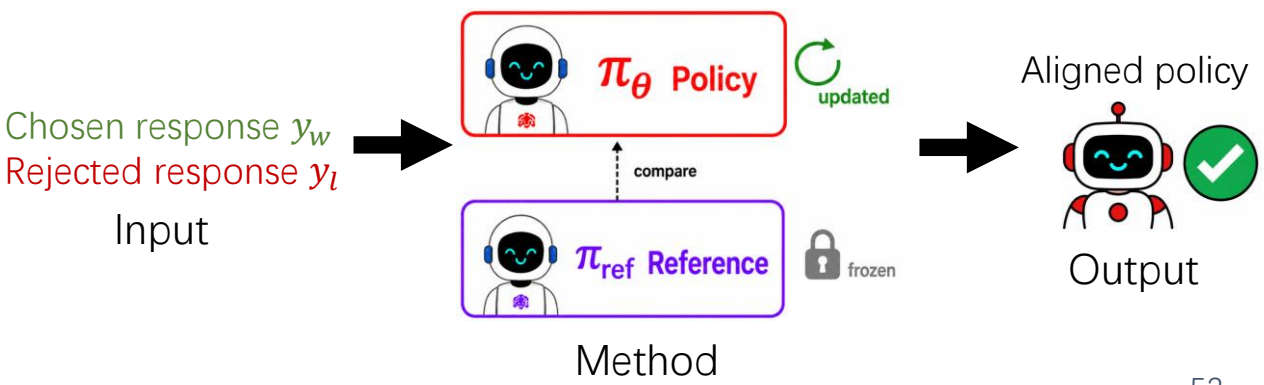
DPO's Intuition

It widens the implicit reward gap by pushing up chosen response y_w and pushing down rejected responses y_l.



DPO's Advantage

DPO simplifies the RLHF pipeline into a single supervised objective.



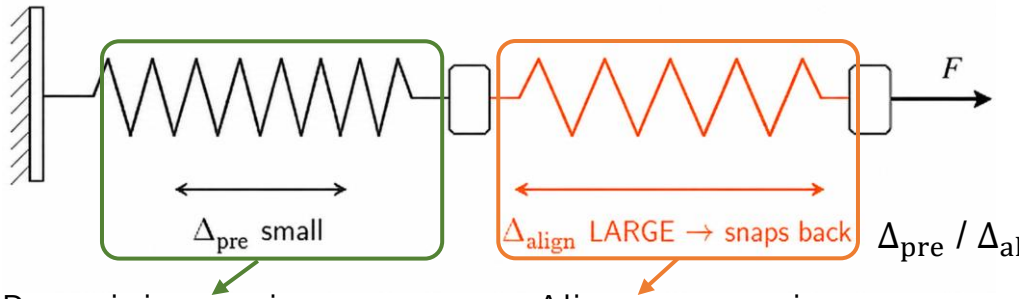
Images generated by ChatGPT-5.5.

Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *NeurIPS*, 2023.

Recent Advances on Alignment

- **Core finding:** The model's built-in data are like **springs**. Two of them sit **in series**: **pretraining** and **alignment**. A spring's **stiffness k equals its data size $|D|$**. More data means a stiffer spring.
- Since $|D_{\text{pre}}|$ is far larger than $|D_{\text{align}}|$, alignment is the soft spring. **Alignment deforms easily under fine-tuning**.

An analogy to Hooke's Law: **Alignment** behaves like a **soft spring**.



The fine-tuning data is the force F pulling on those springs.

Δ_{pre} / Δ_{align}: How far the model's distribution is pushed away.

Pretraining spring:
$k_1 = |D_{\text{pre}}|$, huge → stiff
(Pretraining data is huge.)

Alignment spring:
$k_2 = |D_{\text{align}}|$, tiny → soft
(Alignment data is tiny.)

Two phenomena. Both stem from one mechanism: **Elasticity**.

- **Resistance:** Pushing a model toward alignment is harder than letting it fall back. Models inherently favor pretraining.
- **Rebound:** The deeper the alignment, the faster a reverse nudge snaps it back to pre-training.

- **Takeaways**: Aligning with small fine-tuning data makes fragility structural.
- **Where to go next:** Match data volume across alignment targets, so no objective is the "soft spring" that snaps back.

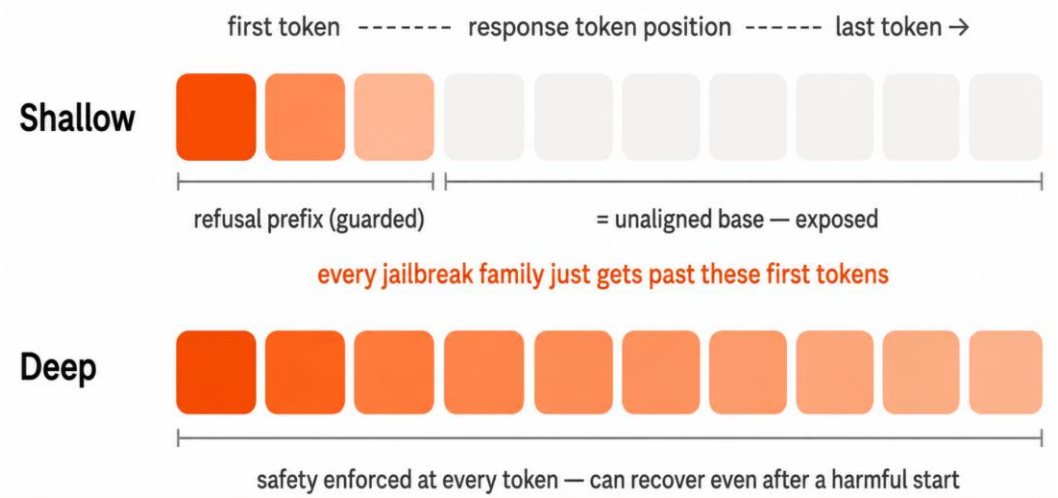
Language Models Resist Alignment: Evidence From Data Compression. In *ACL*, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

Recent Advances on Alignment

- Alignment is only **“a few tokens deep” (Shallow)**: Safety mainly lives in the first few refusal tokens; **bypass them** and the base model is **exposed**.
- **One flaw explains different jailbreaks:** They all just get past those first tokens**.**

The **darker** the color, the **greater the degree of change** in alignment.



Shallow Alignment:
- Safety concentrated in **first tokens**
- Later tokens remain close to base model

Deepened Alignment:
- Safety signal at **every token**
- Harder to bypass

The same shallowness unifies four known jailbreaks (prefilling, adversarial-suffix, decoding-parameter, and fine-tuning attacks), all just bypass or overwrite **the first few tokens**.

Takeaways: Today's safety is shallow because training only rewards a refusal start.
Where to go next: Make safety deeper than the first tokens.
- **Deepen**: Train on **safety recovery examples** (responses start harmful but turn back into a refusal), so safety reaches later tokens.
- **Protect**: During fine-tuning, **constrain updates to the early tokens** so attacks can’t overwrite them.

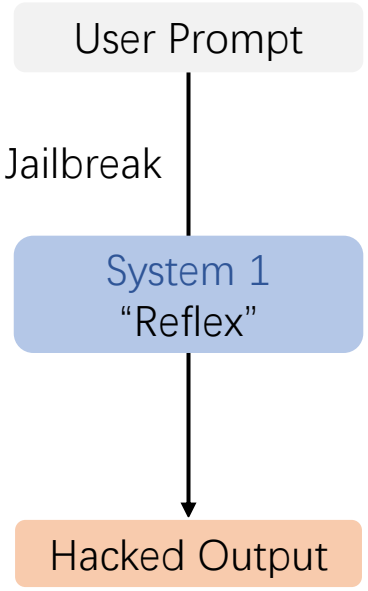
Safety Alignment Should Be Made More Than Just a Few Tokens Deep. In *ICLR*, 2025.

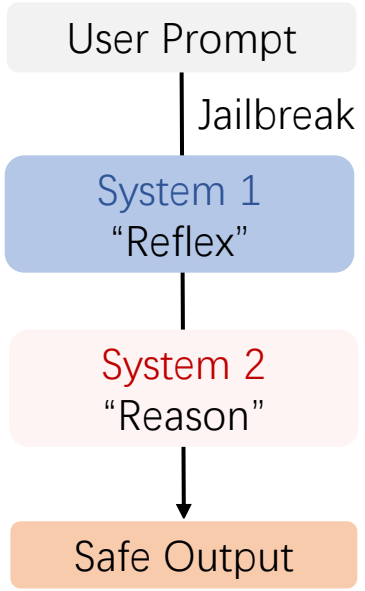
https://bhanml.github.io/ & https://github.com/tmlr-group

Recent Advances on Alignment

- Direct-refusal safety (System 1): It **pattern-matches** the surface prompt, so a disguised jailbreak slips past it.
- STAIR adds System 2 safety reasoning: It writes **risk analysis** before answering, so it judges intent, not wording.

Jailbreaks exploit shallow **System 1** safety reflexes to make the **wrong judgment**.





Jailbreak blocked

How STAIR builds:

- CoT format alignment: SFT to make the model reason in **explicit steps**.
- Self-improvement: Search reasoning paths, reward **safety over helpfulness**.
- Test-time scaling: A reward model picks the safest reasoning path **at inference**.

Takeaways: Don't just train what to refuse, train the model to **reason about why**, so safety **generalizes** to disguised attacks instead of memorizing refusal prefixes.
Where to go next: Keep the safety gains of System-2 reasoning while cutting its added inference cost (longer responses + test-time search).

STAIR: Improving Safety Alignment with Introspective Reasoning, In *ICML*, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

Trustworthy Machine Learning

Part 1. Reasoning

How to obtain a trustworthy LLM that solves complex problems?

Prompting & Test-time Scaling

Elicit step-by-step thinking process without modifying model weights.

Post-training

Reinforce reasoning through supervised fine-tuning and reinforcement learning.

Part 2. Calibration

How to obtain a controllable LLM that conforms to human values?

LLM Unlearning

Remove specific knowledge from a trained model without retraining.

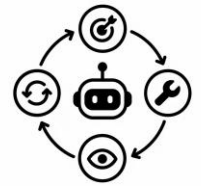
LLM Alignment

Steer model behavior toward human values, intent, and safety.

Part 3. Autonomy

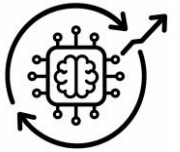
How to enable trustworthy LLM to learn from real-world environments?

Agentic Learning



Learn to plan, act, observe, and adapt to complete a task.

Agentic Evolution



Improve LLM through feedback, selection, and adaptation.

https://bhanml.github.io/ & https://github.com/tmlr-group

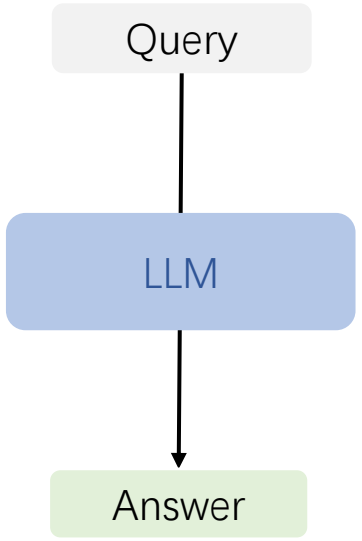
Part 3: Autonomy

- Part 3.1: What is Agent
- Part 3.2: Agentic Reasoning
- Part 3.3: Agentic Learning
- Part 3.4: Agentic Evolution

From Large Language Models to Agents

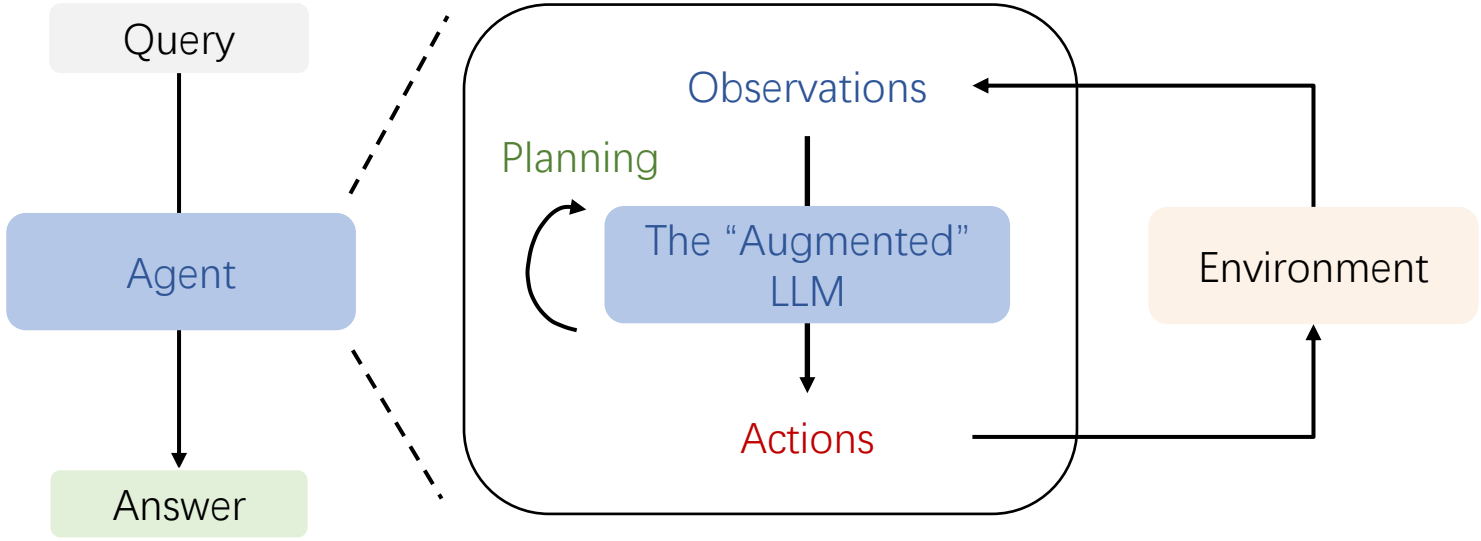
- Large language models perform **text generation** from **intrinsic parametric knowledge**.
- Agents extend foundation models to **interact with environments**.

Large Language Models



LLMs generate responses **directly from model knowledge**.

Agents

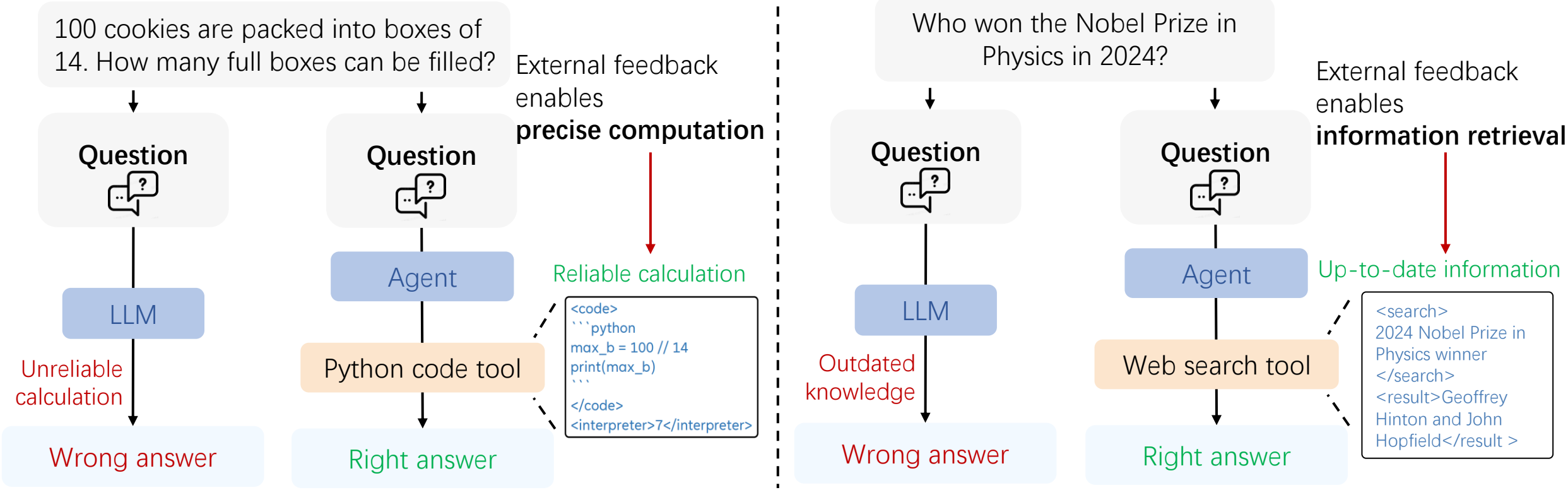


Agents orchestrate **planning**, **actions** and **observations** to fulfill tasks.

https://bhanml.github.io/ & https://github.com/tmlr-group

Why We Need Agents?

- LLMs without **external feedback** are limited to **internal knowledge**.



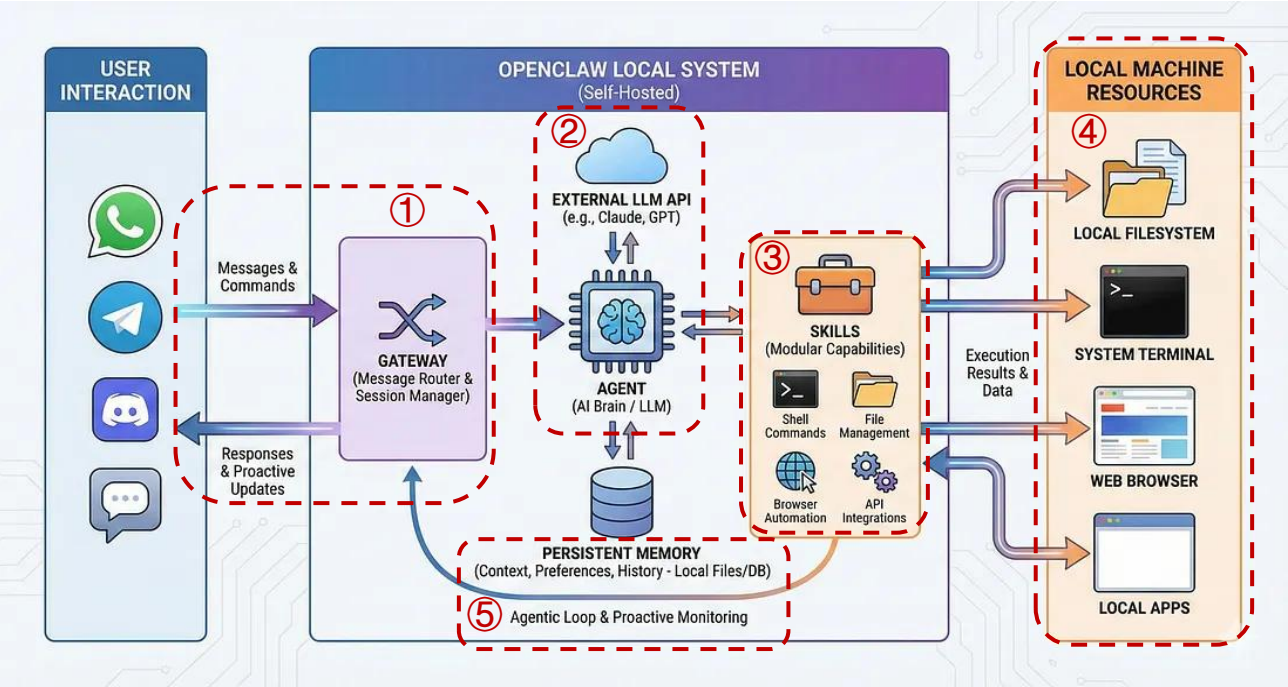
Interacting with environments unlocks **capabilities beyond the model itself**.

https://bhanml.github.io/ & https://github.com/tmlr-group

AutoTool: Dynamic Tool Selection and Integration for Agentic Reasoning. *ArXiv* preprint, 2025.

An Example of Agents: OpenClaw

- **OpenClaw** is an open-source agent system that enables language models to **autonomously** plan, use tools, and **iteratively** solve complex tasks.



1. **Gateway** receives and routes user messages from multiple chat platforms to the agent.
2. **Agent** serves as the brain of OpenClaw to think, plan, and call tools.
3. **Skills** modular tool capabilities that the agent can invoke.
4. **Local Resources** executes actions on the host machine's filesystem, terminal, and apps.
5. **Memory** retains context, preferences, and history across sessions.

Agent systems represent a growing paradigm for **autonomous problem-solving**.

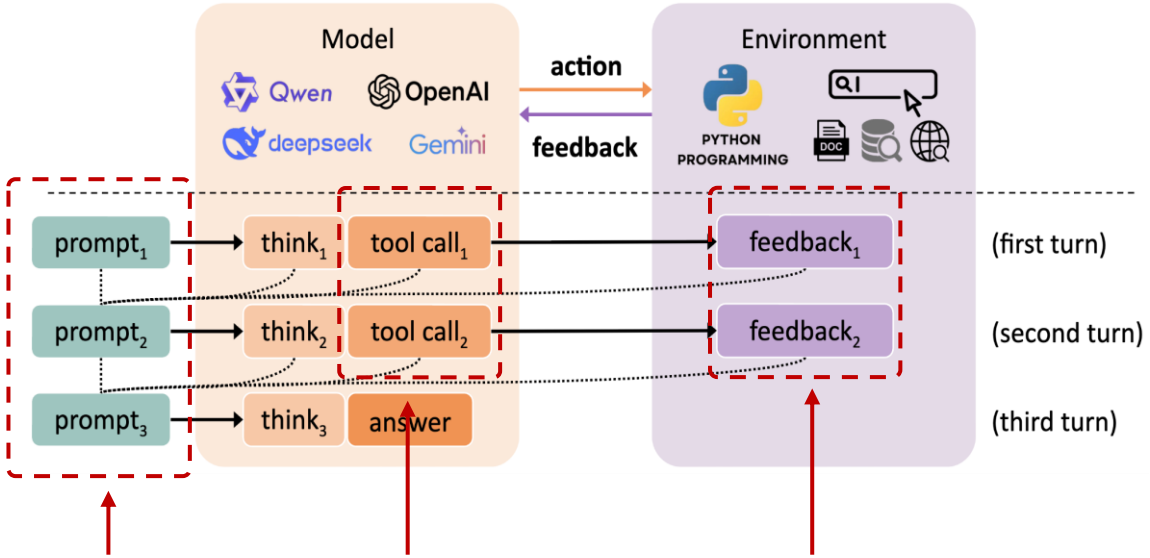
https://bhanml.github.io/ & https://github.com/tmlr-group

Image source from Medium.

What is Agentic Reasoning?

Agentic reasoning moves beyond **static generation** to **interactive problem solving**:

- Action and feedback alternate between the **model** and the **environment**.
- More diverse behaviors including **planning**, **acting**, and **reflecting**.



Each turn's context includes all **prior actions and feedback**.

The model issues **tool calls** to **act** on the environment.

The environment **executes the action** and **returns feedback**.

Agentic reasoning exhibits more diverse **reasoning behaviors**.

Case study: Decomposition

…, we need to **break it down into several steps**: …
Let's start by calculating …

Python Interpreter Calling
```
from math import comb
(…)
```

The result of the code execution is: 210

Next, let's calculate …

Python Interpreter Calling
```
from math import comb
(…)
```

The result of the code execution is: 115, 90, 24, 1
…
Thus, the final answer is: $\boxed{116}$.

Case study: Correction

…Let's write a Python script …

Python Interpreter Calling
```
def find_largest_N():
(…)
```

The result of the code execution is: None
It seems there was an issue … **Let's re-evaluate …**
…
We will re-run the script with a more detailed check …

Python Interpreter Calling
```
def find_largest_N():
(…)
```

The result of the code execution is: 5694
…
Thus, the final answer is: $\boxed{699}$.

Case study: Verification

…Now, we need to compute xy:

$$xy = \left(\frac{10}{a}\right)\left(\frac{5a}{2}\right) = \frac{50a}{2a} = 25$$

Let's verify the solution …

Python Interpreter Calling
```
from sympy import symbols, Eq, solve, log
x, y, a = symbols('x y a')
eq1 = Eq(x*a, 10)
eq2 = Eq(4*y/a, 10)
sol = solve((eq1, eq2), (x, y))
xy = sol[x] * sol[y]
xy.simplify()
```

The result of the code execution is: 25
The value of xy is $\boxed{25}$.

Case study: Backtracking

…Let's implement the above steps …

Python Interpreter Calling
```
from sympy import *
(…)
```

TypeError: cannot determine truth value of Relational.
It appears that the direct symbolic solution approach using SymPy did not work as expected due to the complexity of the equations.
Let's try a different approach …

Python Interpreter Calling
```
# Iterate over possible values of a and b
(…)
```

The result of the code execution is: 601
Thus, the final answer is: $\boxed{601}$.

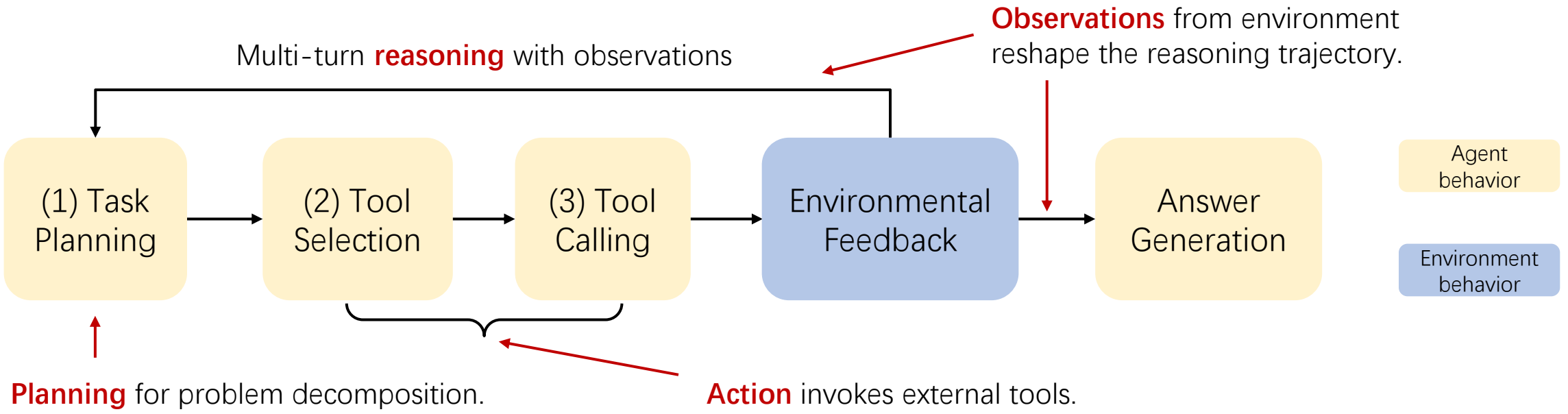
AlphaApollo: A System for Deep Agentic Reasoning. *ArXiv* preprint, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

Pipeline of Agentic Reasoning

Agentic reasoning typically follows a **planning-action-observation** workflow.

- **Planning**: Break down complex problems and formulate strategies.
- **Action**: Interact with tools or environments to execute decisions.
- **Observation**: Collect feedback and refine subsequent reasoning steps.

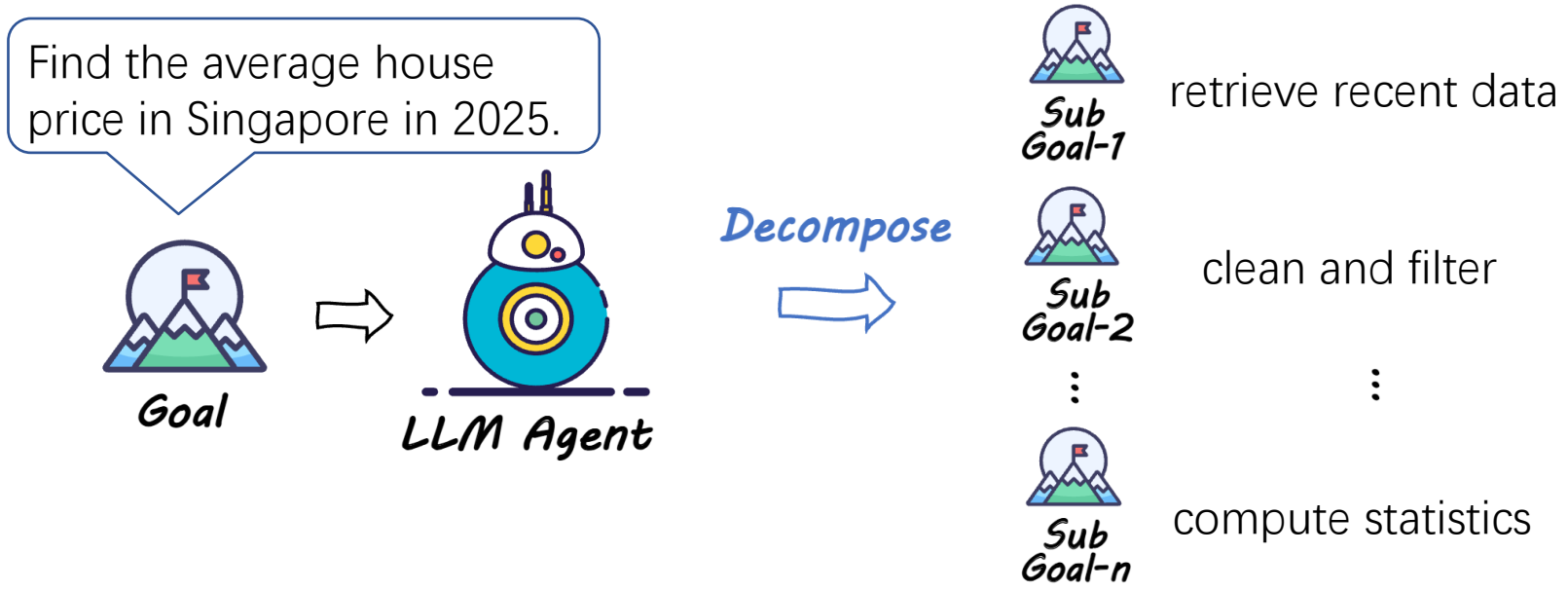


Planning, action, and observation form a **closed loop** that drives agentic reasoning.

A Closer Look at Task Planning (Step 1)

Planning breaks the task into **sub-problems** and identifies their **dependencies**.

- Planning drives **structured**, **step-by-step** progress toward **complex** objectives.



Planning bridges the gap between **goals** and **executable actions**.

Understanding the planning of LLM agents: A survey. *ArXiv* preprint, 2024.

https://bhanml.github.io/ & https://github.com/tmlr-group

A Closer Look at Tool Selection (Step 2)

Tools can be categorized by the **roles** they play in the planning-action-observation loop.

- Typical roles include **information retrieval**, **computation**, and **environmental interaction**.

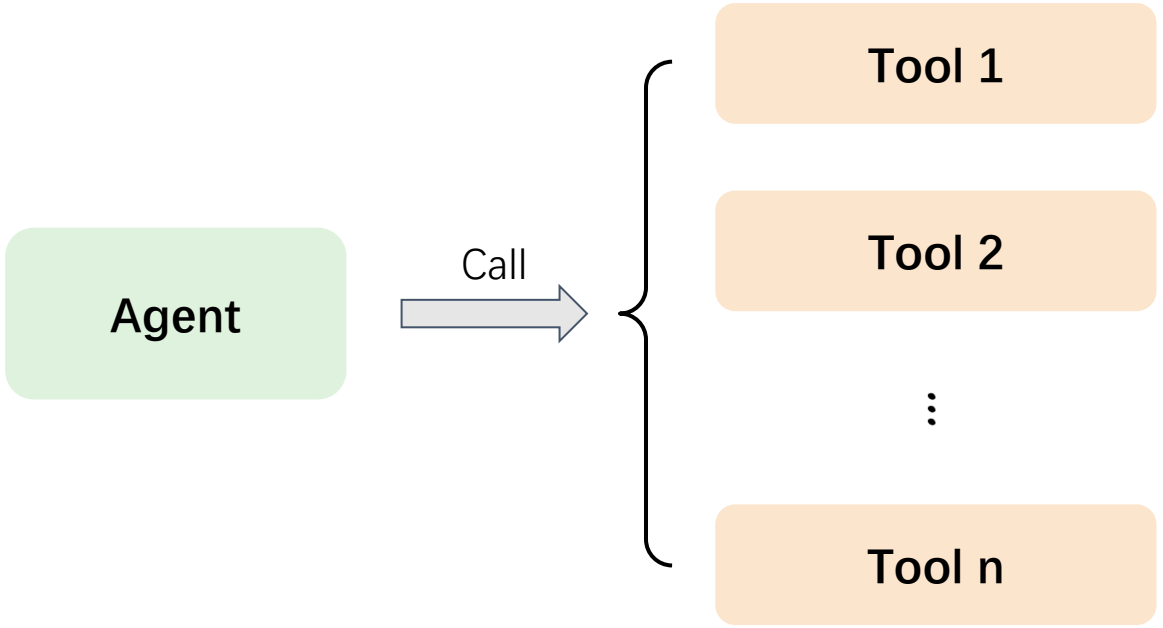
Tools	Description
Retrieval Tools 	Access and search external knowledge sources to obtain relevant information.
Computation Tools 	Perform precise calculations, simulations, or code execution for math reasoning.
Interaction Tools 	Enable communication with users, agents, or external systems.
Execution Tools 	Execute actions such as running programs, APIs, or system commands.
Verification Tools 	Check correctness, consistency, or validity of intermediate or final results.

The **breadth of available tools** enables a broader **range of agent capabilities**.

A Closer Look at Tool Calling (Step 3)

Tool calling relies on **structured descriptions** to guide selection and invocation.

- It **bridges** the agent's reasoning and the external environment.




```
{
  "name": "get_weather_data",
  "title": "Weather Data Retriever",
  "description": "Get current weather data for a location",
  "inputSchema": {
    "type": "object",
    "properties": {
      "location": {
        "type": "string",
        "description": "City name or zip code"
      }
    },
    "required": ["location"]
  },
  "outputSchema": {
    "type": "object",
    "properties": {
      "temperature": {
        "type": "number",
        "description": "Temperature in celsius"
      },
      "conditions": {
        "type": "string",
        "description": "Weather conditions description"
      },
      "humidity": {
        "type": "number",
        "description": "Humidity percentage"
      }
    },
    "required": ["temperature", "conditions", "humidity"]
  }
}
```

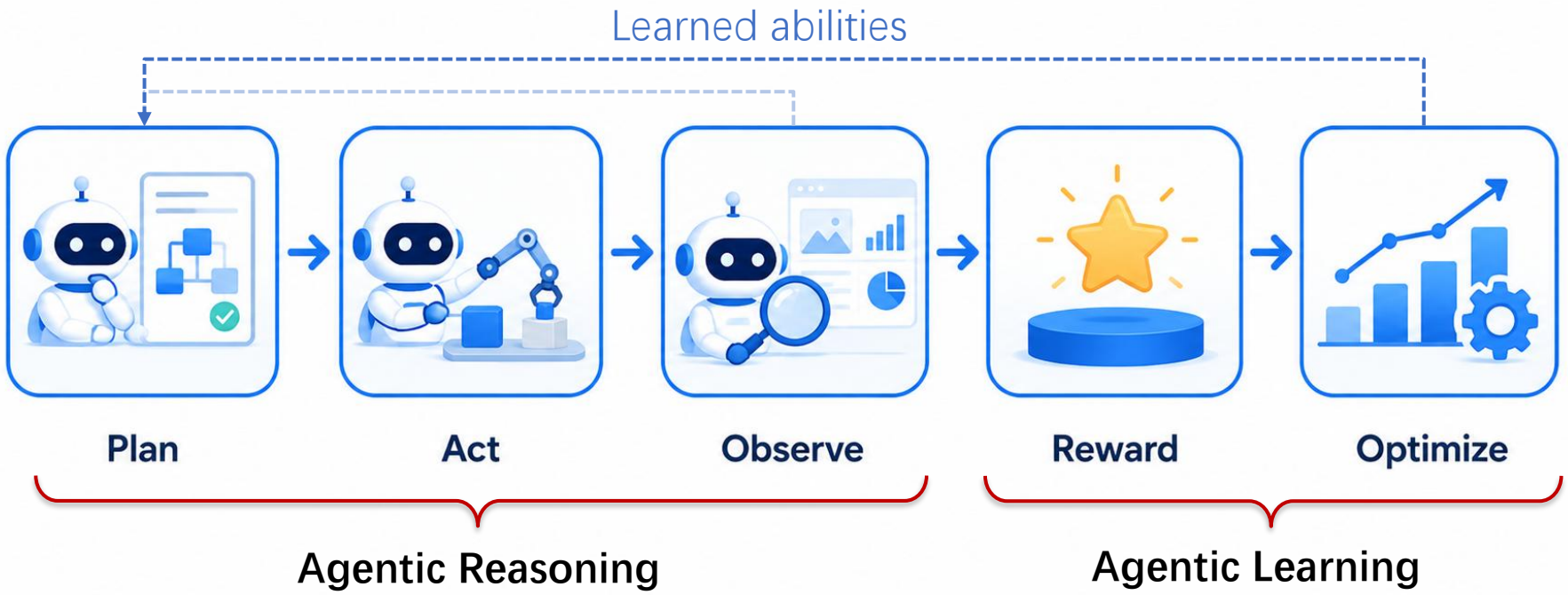
A tool interface usually includes the **tool name**, **description**, **input schema**, and **output schema**.

Tool calling **translates** the agent's **reasoning** into **structured interactions** with the external environment.

Example sourced from Model Context Protocol.

What is Agentic Learning

- Agentic learning post-trains models by optimizing the Planning-Action-Observations loop using **task-level feedback** from **environmental interactions**.



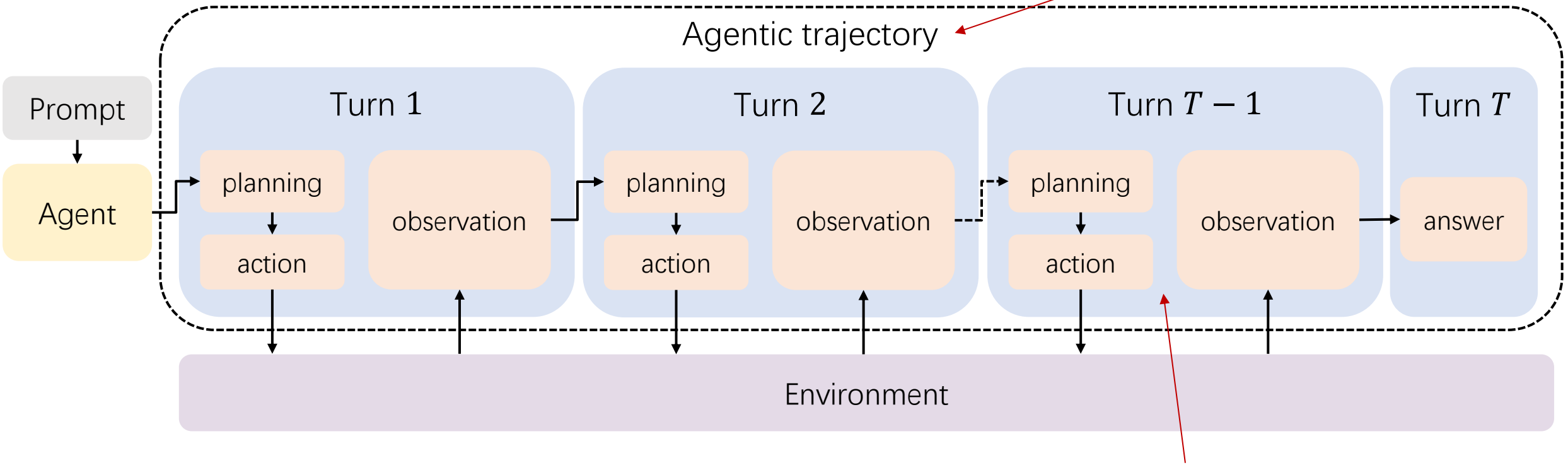
Agentic learning teaches agents to **learn from environmental feedback**.

https://bhanml.github.io/ & https://github.com/tmlr-group

Images generated by GPT 5.5

Trajectories and Turns in Agentic learning

- Agentic learning is **trajectory-based**: each trajectory records an agent's **complete task-solving process.**



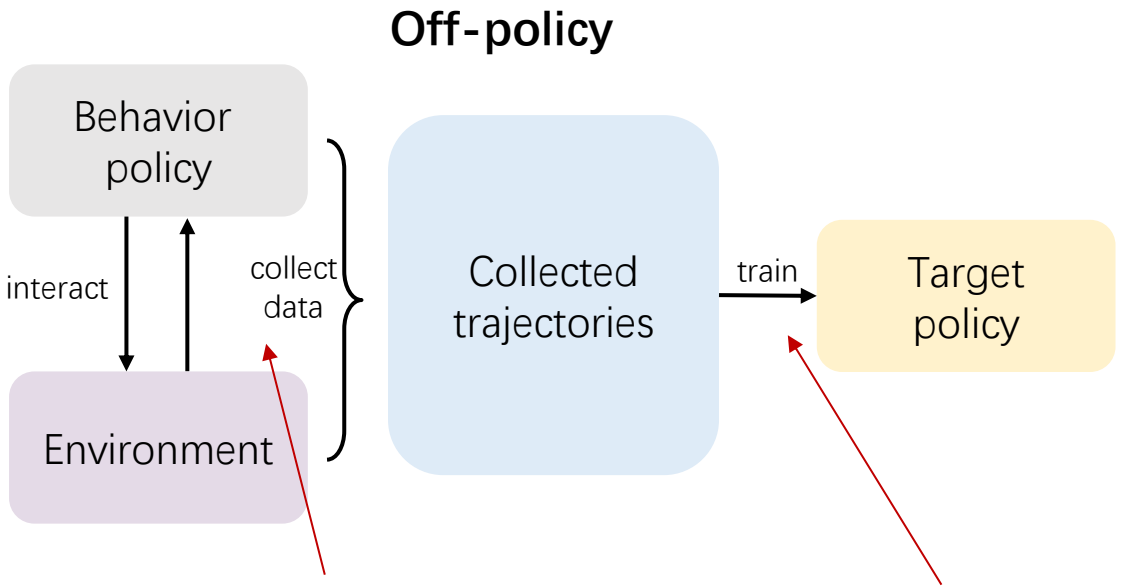
An agentic trajectory consists of **multiple turns** of planning, action, and observation.

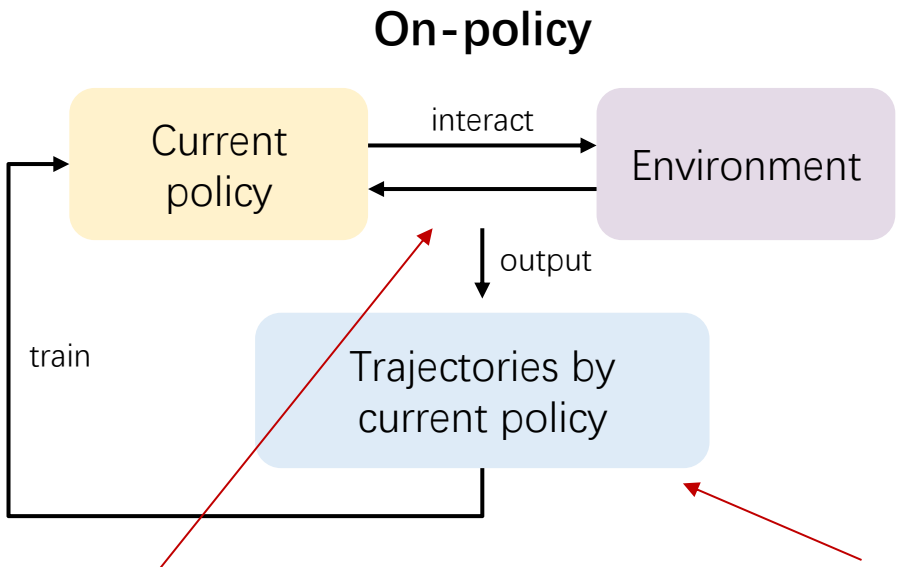
Images generated by GPT 5.5

Typical Agentic Learning Paradigms

Agentic learning optimizes agents at the post-training stage on agentic trajectories **collected in different ways**.

- **Off-policy methods** leverage trajectories generated **outside the current policy**.
- **On-policy methods** train on the agent's **own interaction trajectories**.





The **behavior policy** interacts with environment to collect trajectories.

The **target policy** is optimized based on the collected data.

The **current policy** interacts with the environment directly.

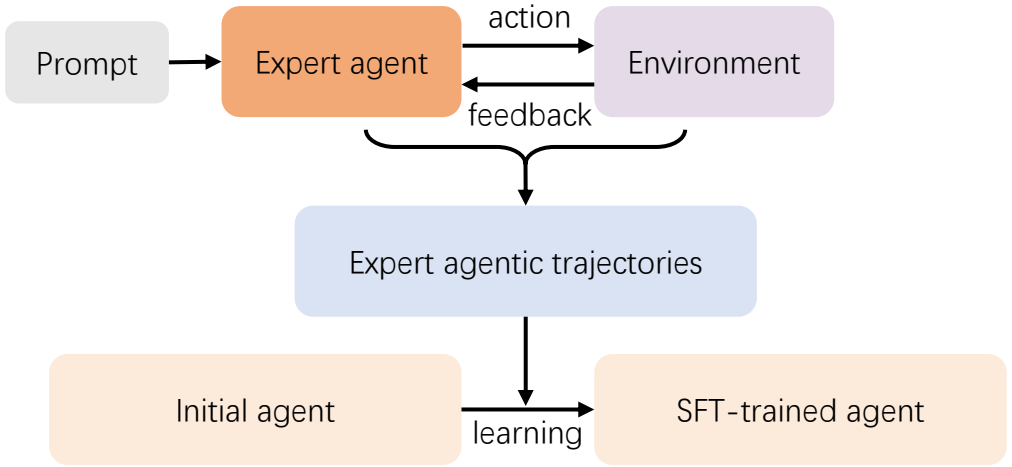
Only current-policy trajectories are used for optimization.

Data-efficient but **distribution-mismatched**.

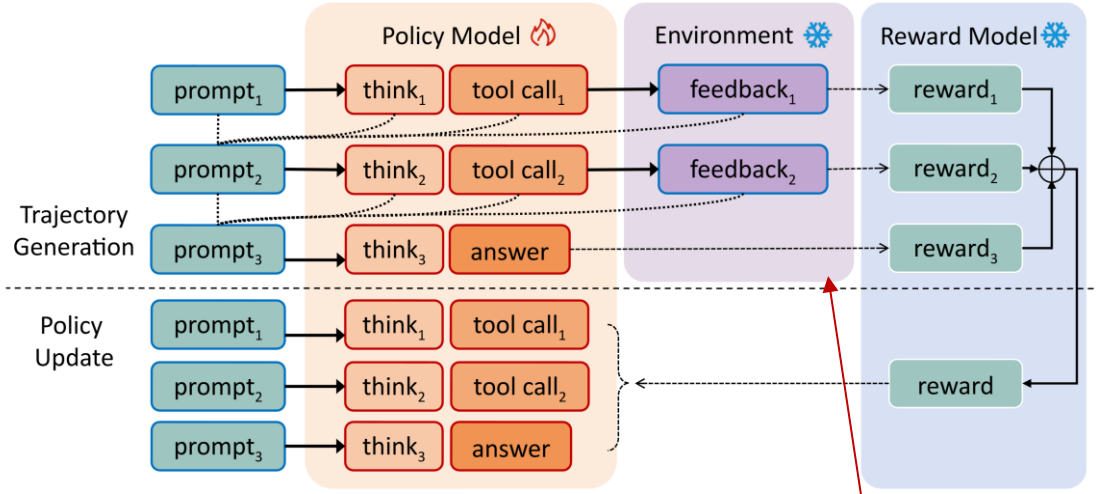
Well-aligned but **sample-costly**.

Typical Agentic Learning Methods

- Agentic learning can involve **supervised imitation** and **reinforcement-driven optimization** to train models for interactive reasoning and tool use.



Agentic SFT teaches agents how to reason and use tools through **expert demonstrations**.



Agentic RL optimizes agents to make better decisions from **environmental feedback**.

SFT distills **expert agentic trajectories**; RL optimizes decisions from **environmental feedback**.

https://bhanml.github.io/ & https://github.com/tmlr-group

Tool-integrated Reinforcement Learning for Repo Deep Search. *ArXiv* preprint, 2025.

Agentic Learning: SFT

- In Agentic SFT, the supervision minimizes **token-level prediction loss** on multi-turn interaction trajectories to **imitate behaviors**.

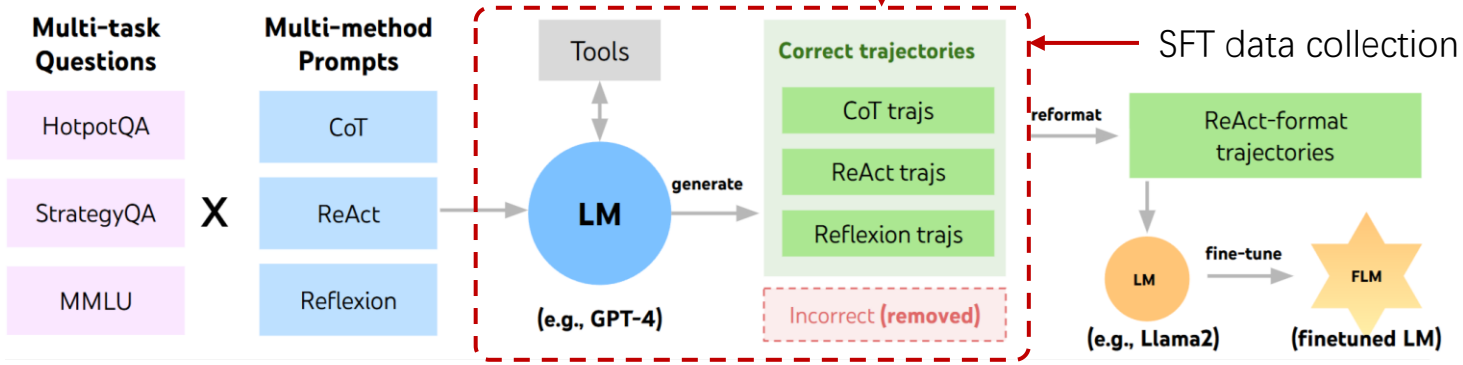
- τ_T: Trajectory with T turns.
- π_θ: Agent policy.
- $o_{t,k}$: k-th token of t-th turn.
- p_t: Prompt of t-th turn.
- $\pi_\theta(\cdot \mid \cdot)$: Probability of generated token.

$$\mathcal{J}_{\text{SFT}}(\theta) = \mathbb{E}[q, o \sim P_{\text{SFT}}(Q, O)] \left(\frac{1}{|o|} \sum_{t=1}^{|o|} \log \pi_\theta(o_t | q, o_{<t}) \right)$$

Vanilla SFT uses question-output pairs.

$$\mathcal{J}_{\text{Turn-SFT}}(\theta) = \mathbb{E}_{[\tau_T \sim \mathcal{D}_{\text{SFT}}]} \left[\sum_{t=1}^{T} \sum_{k=1}^{|o_t|} \log \pi_\theta(o_{t,k} | p_t, o_{t,<k}) \right]$$

Agentic SFT uses multi-turn interaction trajectories.



Agentic SFT leverages **expert-curated** or **strong model-generated** agentic trajectories to bootstrap agent capabilities.

FireAct: Toward Language Agent Finetuning. *ArXiv* preprint, 2025.
Training language models to follow instructions with human feedback. In *NeurIPS*, 2022.

https://bhanml.github.io/ & https://github.com/tmlr-group

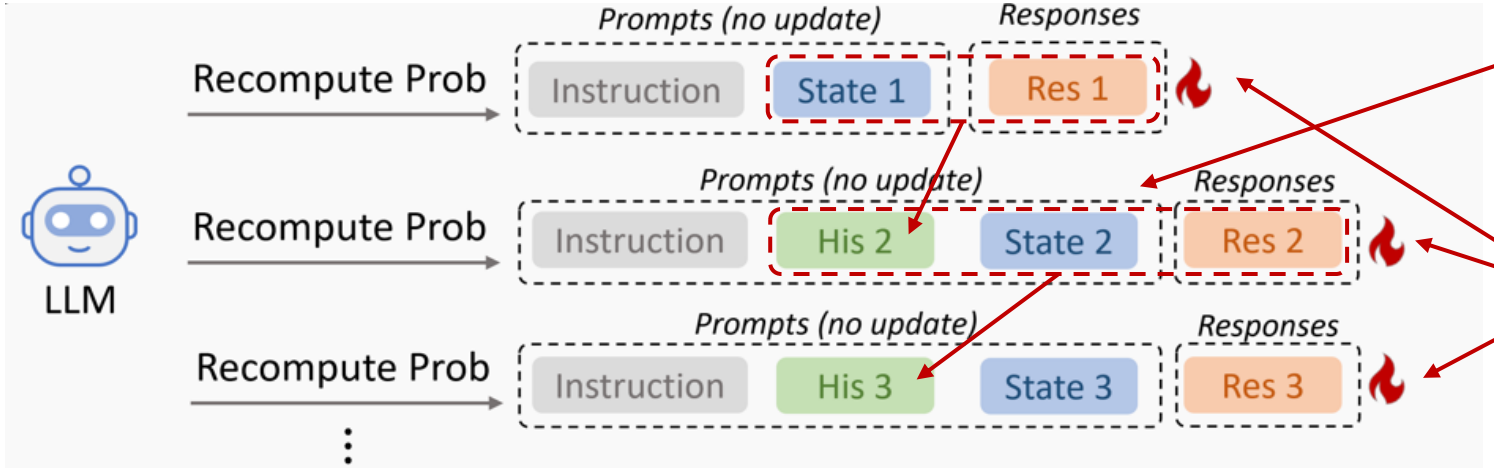
Agentic Learning: Turn-level GRPO

- Turn-level GRPO extends GRPO to **multi-turn trajectories**, assigning optimization signals at the **turn level** to better train sequential decision-making in agentic settings.

$$\mathcal{J}_{\text{Turn-GRPO}}(\theta) = \mathbb{E}_{(q,a)\sim\mathcal{D},\{o^{(i)}\}_{i=1}^{N}\sim\pi_{\text{old}}(\cdot|q)}\left[\frac{1}{N}\sum_{i=1}^{N}\frac{1}{T^{(i)}}\sum_{t=1}^{T^{(i)}}\frac{1}{|o_t^{(i)}|}\sum_{k=1}^{|o_t^{(i)}|}\min\left(\frac{\pi_\theta\left(o_{t,k}^{(i)}|p_t,o_{t,<k}^{(i)}\right)}{\pi_{\text{old}}\left(o_{t,k}^{(i)}|p_t,o_{t,<k}^{(i)}\right)}A_{t,k}^{(i)},\right.\right.$$
$$\left.\left.\text{clip}\left(\frac{\pi_\theta\left(o_{t,k}^{(i)}|p_t,o_{t,<k}^{(i)}\right)}{\pi_{\text{old}}\left(o_{t,k}^{(i)}|p_t,o_{t,<k}^{(i)}\right)},1-\epsilon,1+\epsilon\right)A_{t,k}^{(i)}\right) - \beta\mathbb{D}_{\text{KL}}[\pi_\theta\|\pi_{\text{ref}}]\right]$$

- (q,a): Question-answer pair.
- π_{old}: Behavior policy.
- π_{ref}: Reference policy.
- $A_{t,k}^{(i)}$: Advantage on $o_{t,k}^{(i)}$.

Normalize by **the number of turns**



The prompt for each turn contains **the history of all previous turns**.

$$\text{His t}+1 = \text{His t} + \text{State t} + \text{Res t}$$

A complete **agentic trajectory** is segmented at tool-call boundaries into **individual turns**, each optimized **independently**.

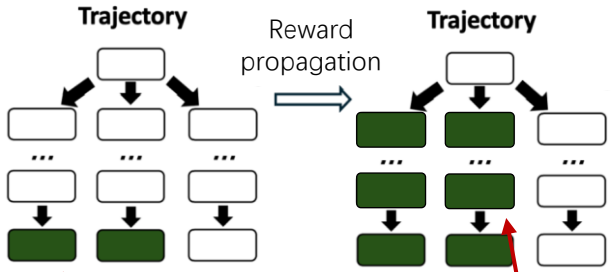
Group-in-Group Policy Optimization for LLM Agent Training. In *NeurIPS*, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

Turn-level GRPO with Dense Rewards

- Agentic performance depends on the quality of decisions made **throughout the trajectory**.
- Both failures and achievements **emerge at intermediate turns**, not only in the final outcome.

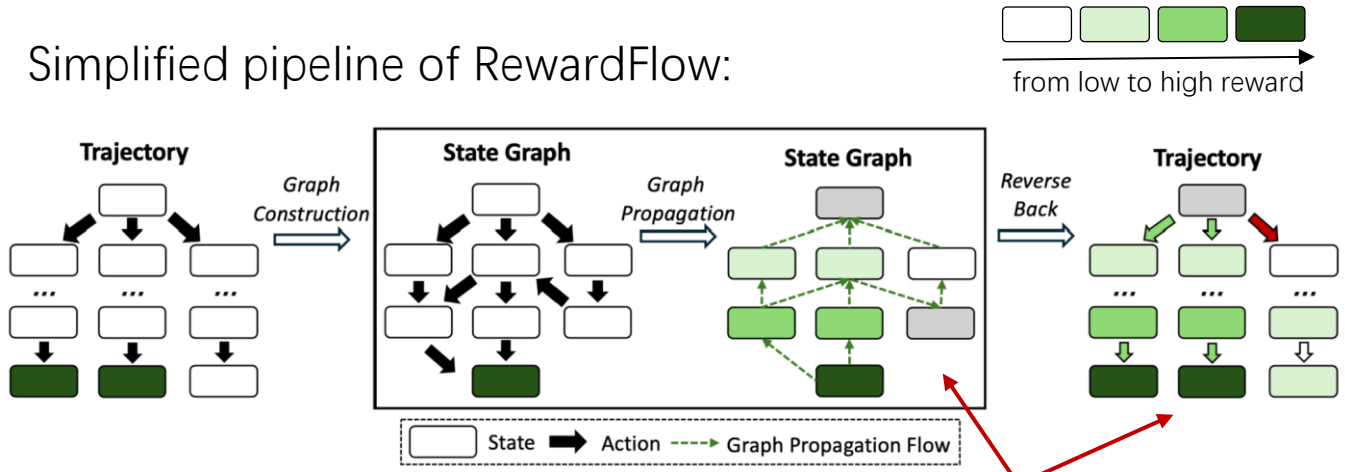
Pipeline of turn-level GRPO:



The reward is computed from the **final turn's outcome**.

The **same reward** is assigned to every turn within the same trajectory.

Simplified pipeline of RewardFlow:



Backpropagate rewards from success states to intermediate states for **fine-grained** credit assignment.

An outcome reward is too **sparse**, and **denser** process rewards are necessary for agentic learning.

https://bhanml.github.io/ & https://github.com/tmlr-group

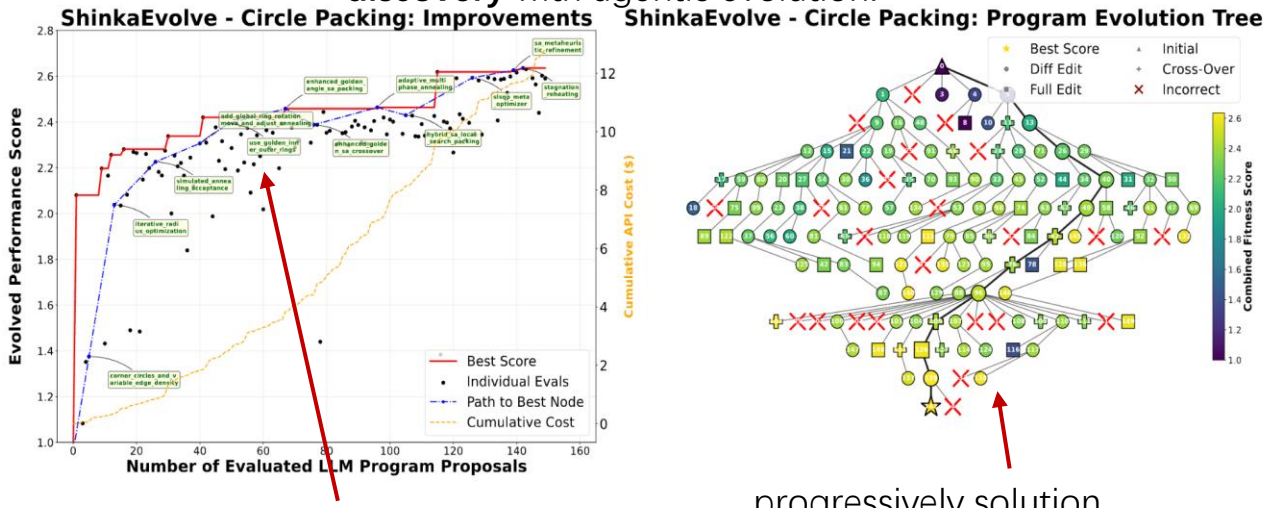
RewardFlow: Topology-Aware Reward Propagation on State Graphs for Agentic RL with Large Language Model. *ArXiv* preprint, 2026.

What is Agentic Evolution?

Agentic evolution is the process of iteratively **exploring**, **evaluating**, and **refining**.

- **Deeper reasoning** extends the chain of thoughts within a single problem.
- **Wider reasoning** broadens the diversity of strategies explored across problems.

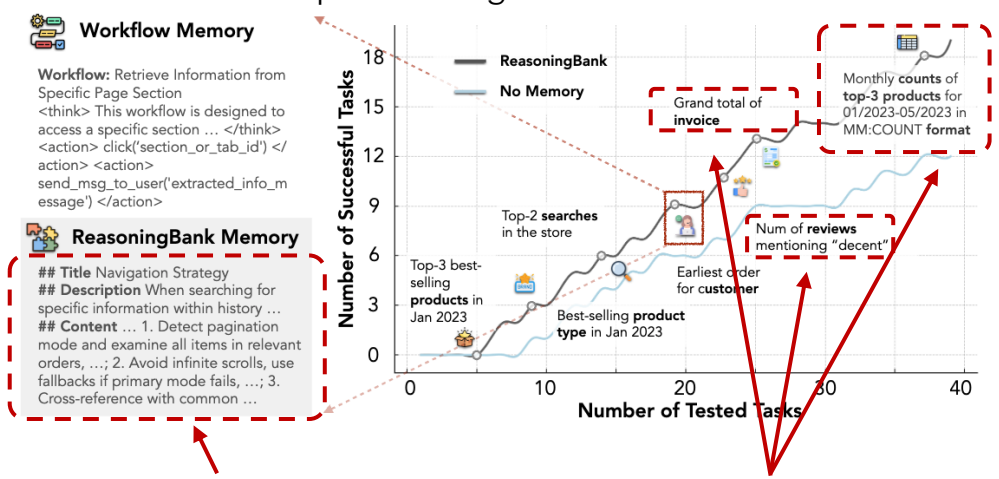
ShinkaEvolve (Sakana AI) drives scientific discovery with agentic evolution.



Strong **problem-solving** capability.

progressively solution construction through **iteration**.

Reasoning Bank (Google) induces reusable memory for powerful agentic evolution.



Past trajectories become **reusable memory**.

Memory improves solutions **across later tasks**.

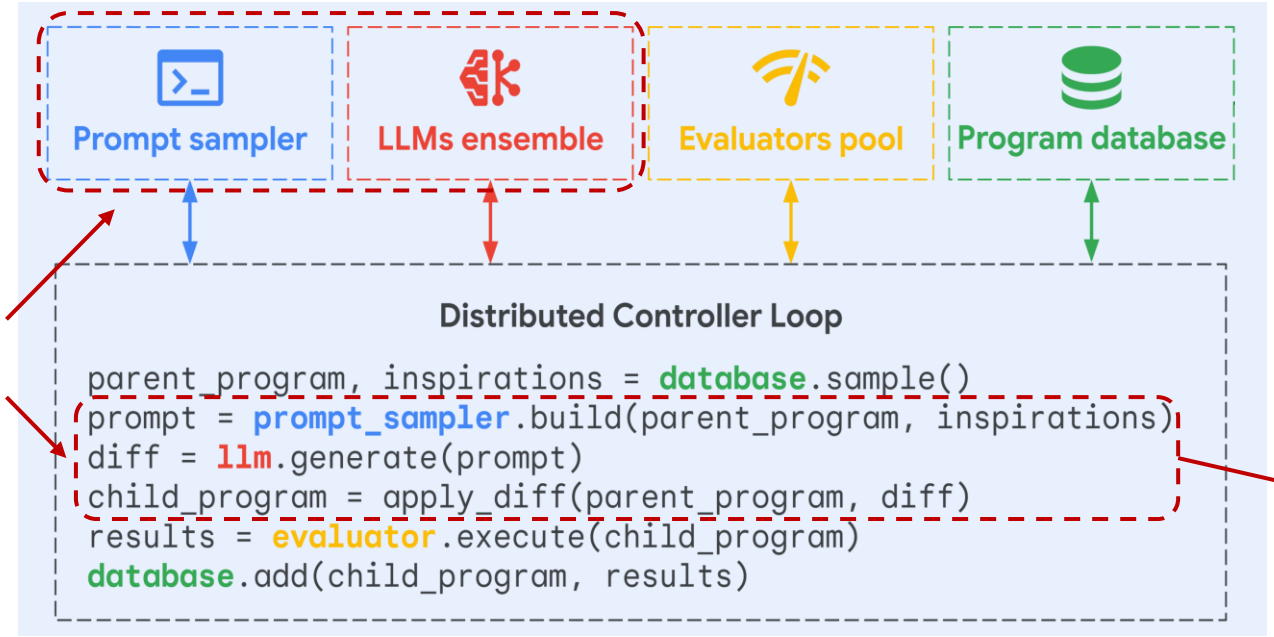
Agentic evolution enables agents to **evolve and improve solutions** at test time.

ShinkaEvolve: Towards Open-Ended and Sample-Efficient Program Evolution. *ArXiv* preprint, 2025.
Reasoning Bank: Scaling Agent Self-Evolving with Reasoning Memory. *ArXiv* preprint, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

A Closer Look at Agentic Evolution: Refinement

- In agentic evolution, refinement improves candidate solutions through iterative **revision**, **extension**, or **recombination**, allowing the agent to reach better final outcomes.



Prior solutions are consolidated into each new prompt.

```
The current model uses a simple ResNet architecture with only
three ResNet blocks. We can improve its performance by
increasing the model capacity and adding regularization. This
will allow the model to learn more complex features and
generalize better to unseen data. We also add weight decay to
the optimizer to further regularize the model and prevent
overfitting. AdamW is generally a better choice than Adam,
especially with weight decay.


<<<<<<< SEARCH
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels * 2, stride=2)
    self._block3 = ResNetBlock(num_channels * 4, stride=2)
=======
    self._block1 = ResNetBlock(num_channels)
    self._block2 = ResNetBlock(num_channels, stride=1)
    self._block3 = ResNetBlock(num_channels * 2, stride=2)
    self._block4 = ResNetBlock(num_channels * 2, stride=1)
    self._block5 = ResNetBlock(num_channels * 4, stride=2)
    self._block6 = ResNetBlock(num_channels * 4, stride=1)
>>>>>>> REPLACE
<<<<<<< SEARCH
  def optimizer(self, learning_rate):
    return optax.adam(learning_rate)
=======
  def optimizer(self, learning_rate):
    return optax.adamw(learning_rate, weight_decay=1e-4)
>>>>>>> REPLACE
```

Agents propose **improved solutions** informed by accumulated context.

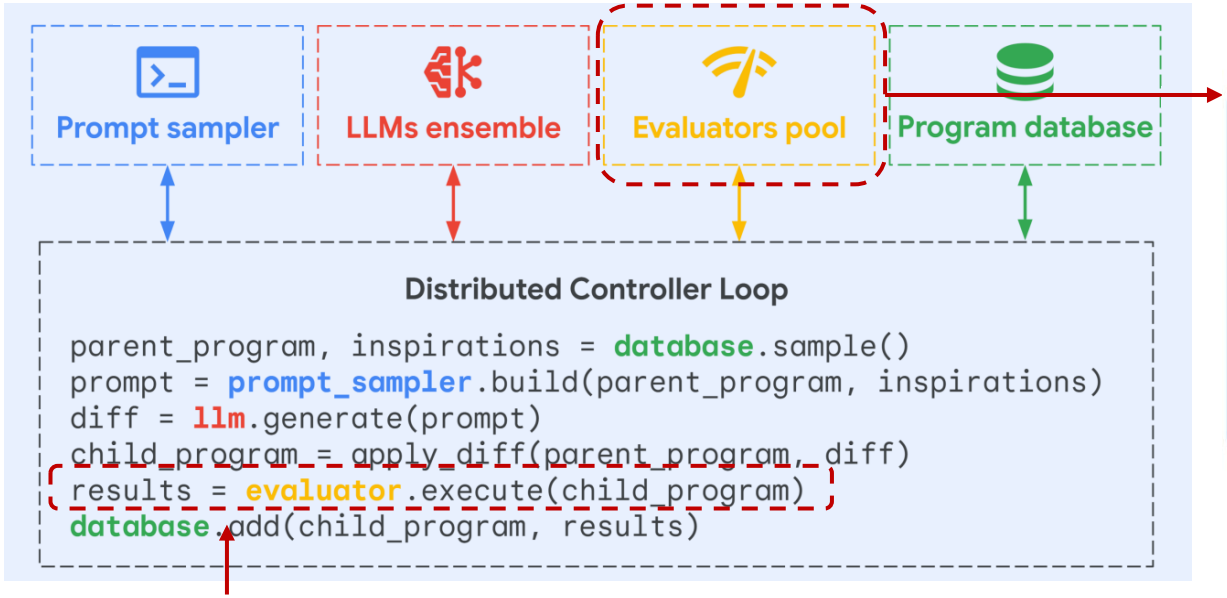
Refinement transforms existing candidates into **higher-quality solutions** through **iterative improvement**.

AlphaEvolve: A coding agent for scientific and algorithmic discovery. *ArXiv* preprint, 2025.

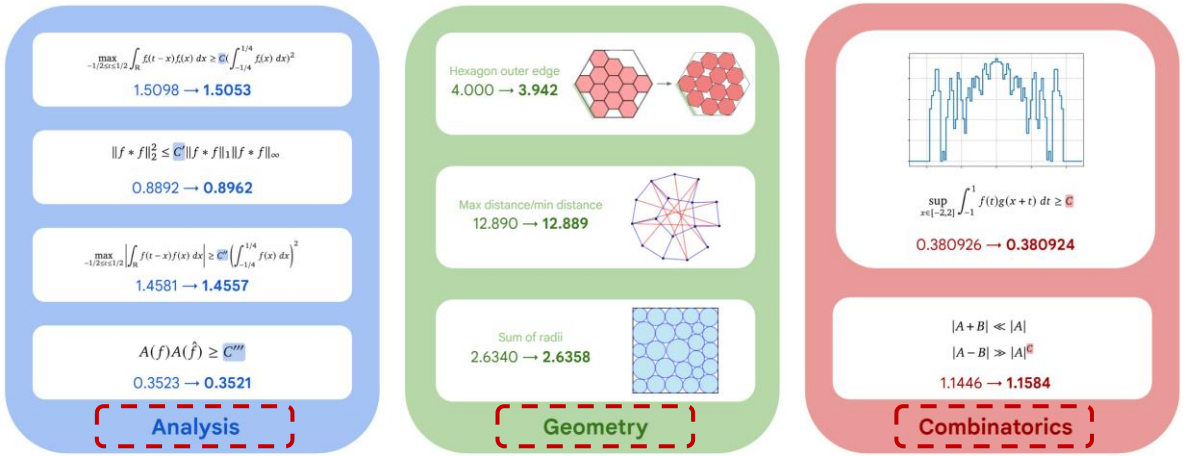
https://bhanml.github.io/ & https://github.com/tmlr-group

A Closer Look at Agentic Evolution: Evaluation

- In agentic evolution, evaluation identifies which candidate solutions are **promising**, **effective**, or worth **further exploration**, helping the agent focus on high-potential directions.



New solutions are evaluated to **track** the progress and to **select** which ideas to propagate in future evolution.



Different tasks adopt different evaluation **methods** and **metrics**.

Evaluation provides the signals that **guide the evolutionary loop**.

AlphaEvolve: A coding agent for scientific and algorithmic discovery. *ArXiv* preprint, 2025.

https://bhanml.github.io/ & https://github.com/tmlr-group

A Closer Look at Agentic Evolution: Memory

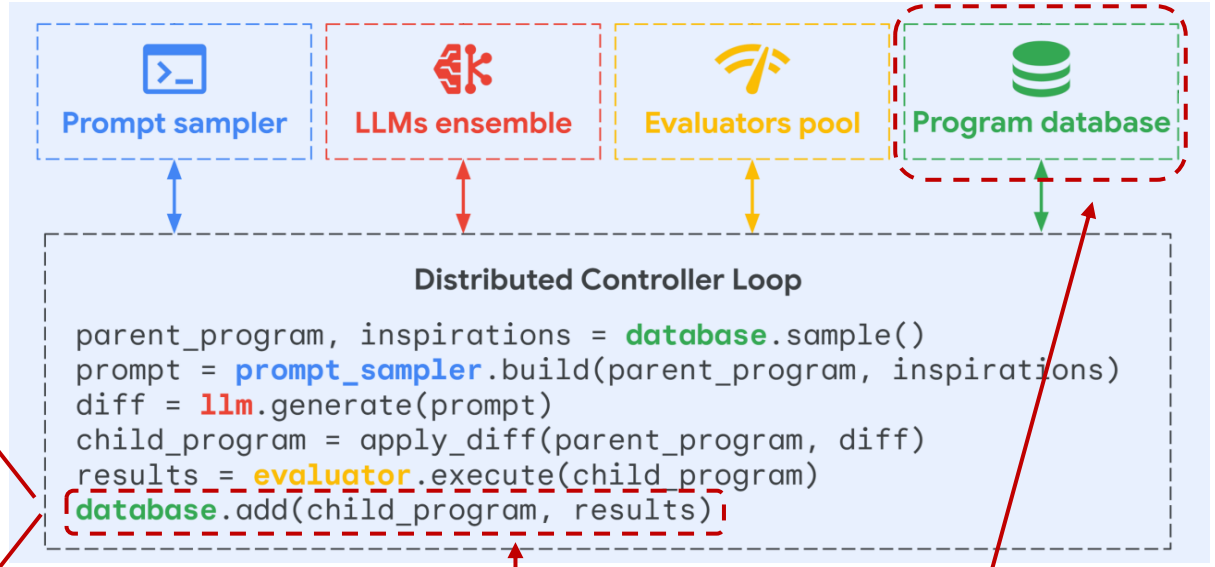
- In agentic evolution, memory preserves **useful information** from previous attempts to support future reasoning and refinement, enabling **long-horizon** evolution.

Previous memory

Incoming memory

```
Act as an expert software developer. Your task is to iteratively
improve the provided codebase. [...]

- Prior programs

Previously we found that the following programs performed well
on the task at hand:

top_1_acc: 0.796; neg_eval_log_loss: 0.230; average_score: 0.513

"""Image classification experiment in jaxline."""
[...]
class ConvNet(hk.Module):
  """Network."""

  def __init__(self, num_channels=32, num_output_classess=10):
    super().__init__()
    self._conv1 = hk.Conv2D(num_channels, kernel_shape=3)
    self._conv2 = hk.Conv2D(num_channels * 2, kernel_shape=3)
    self._conv3 = hk.Conv2D(num_channels * 4, kernel_shape=3)
    self._logits_module = hk.Linear(num_output_classes)
[...]
```

```
- Current program

Here is the current program we are trying to improve (you will
need to propose a modification to it below).

top_1_acc: 0.862; neg_eval_log_loss: 0.387; average_score: 0.624
```



Promising solutions are registered back into the memory, driving the **iterative discovery** of better and better programs.

Memory turns past attempts and outcomes into **reusable experience** for future evolution.

AlphaEvolve: A coding agent for scientific and algorithmic discovery. *ArXiv* preprint, 2025.

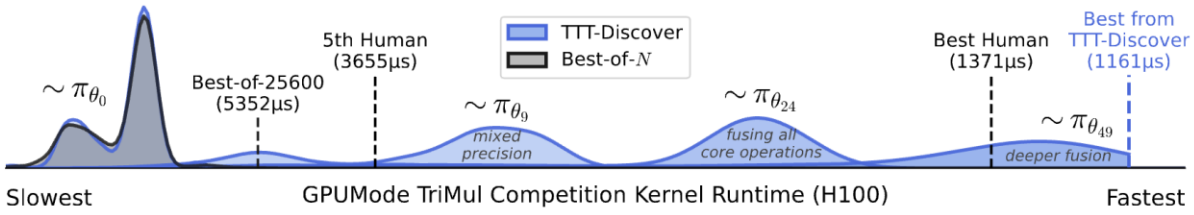
https://bhanml.github.io/ & https://github.com/tmlr-group

When Agentic Evolution Meets Learning

- Rather than treating test-time evolution as a **fixed procedure**, agentic learning can teach agents how to **progressively improve** solutions across turns.

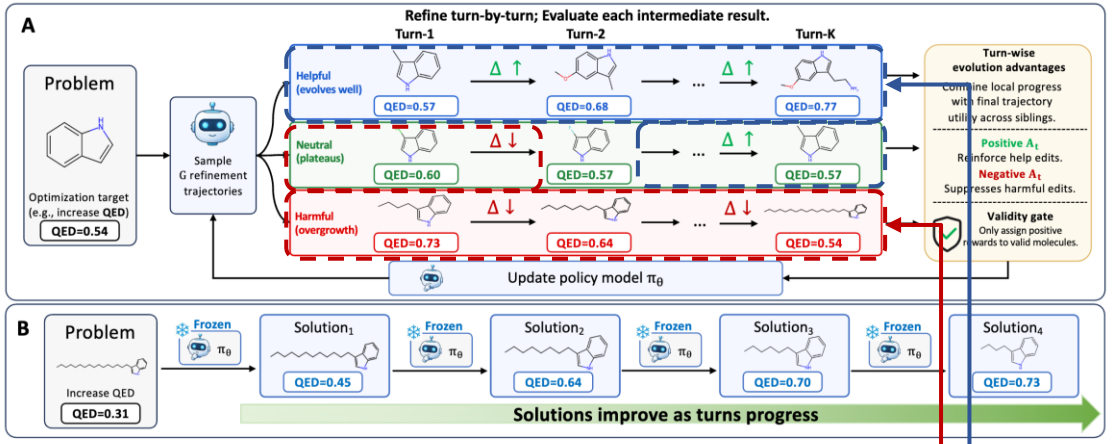
TTT-Discover uses test-time RL for scientific discovery.

	Mathematics	Kernel Eng. (TriMul)		Algorithms (AtCoder)	Biology
	Erdős' Min. Overlap (↓)	A100 (↓)	H100 (↓)	Heuristic Contest 39 (↑)	Denoising (↑)
Best Human	0.380927 [20]	4531 µs	1371 µs	566,997 [56]	0.64
Prev. Best AI	0.380924 [50]	N/A	N/A	558,026 [37]	N/A
TTT-Discover	0.380876	2198 µs	1161 µs	567,062	0.71



Test-time agentic training with **verifiable feedback from agentic evolution** reaches new SOTAs in discovery tasks.

Learning to Evolve learns a policy for iterative refinement.



Regressions are **discouraged**, while improvements are **reinforced**.

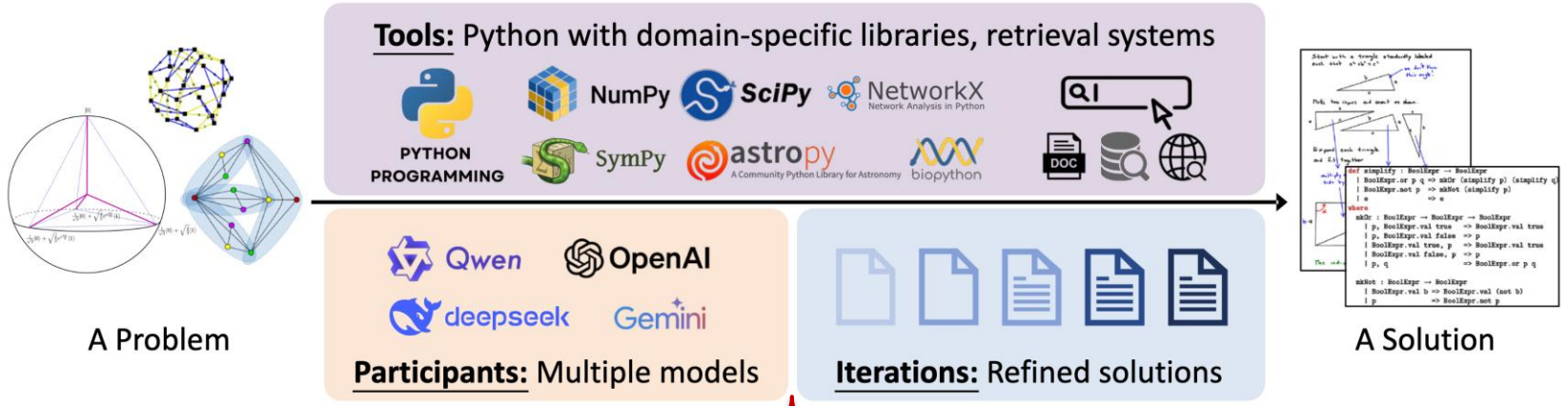
By combining with evolution, agentic learning **internalizes the evolutionary capability** into the model itself.

Learning to Discover at Test Time. In *ICML*, 2026.
Learning to Evolve: Scaling Open-Ended Discovery with Relative-Progress RL. In *ICLR LLA Workshop*, 2026.

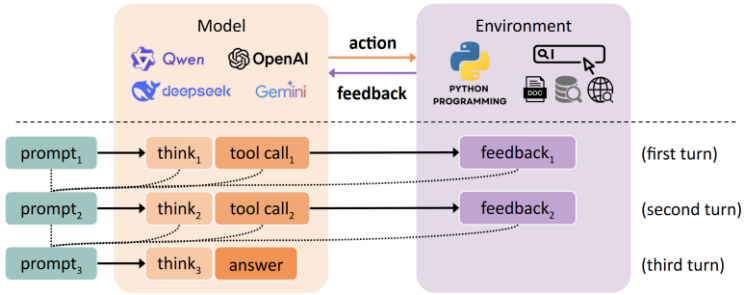
https://bhanml.github.io/ & https://github.com/tmlr-group

AlphaApollo: Agentic Reasoning System

AlphaApollo provides a **unified platform** of **agentic reasoning, learning, and evolution.**

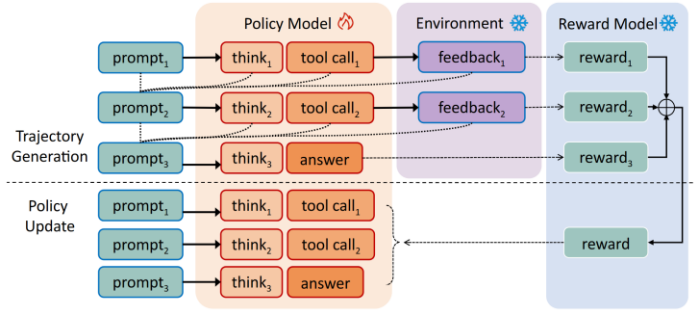


Website Paper Code



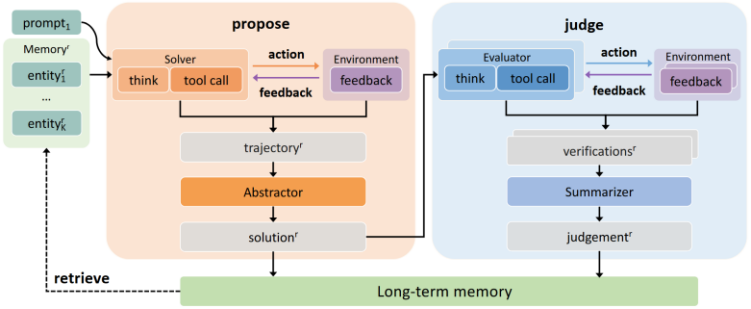
Agentic Reasoning

Multi-turn agentic reasoning through an iterative cycle of model reasoning, tool execution, and environment feedback.



Agentic Learning

Stable agentic learning via turn-level optimization that decouples model generations and environmental feedback.



Agentic Evolution

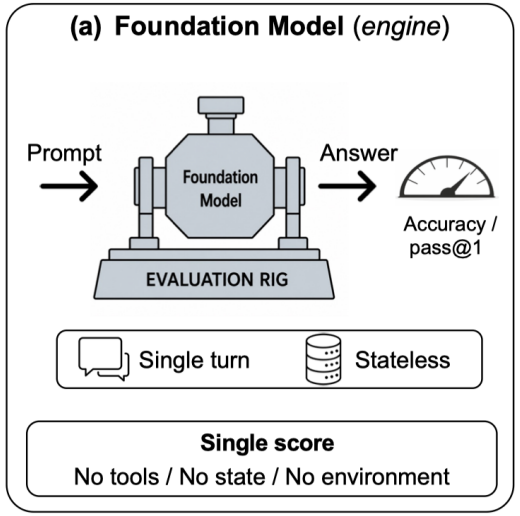
Multi-round agentic evolution through a propose-judge-update evolutionary loop with long-term memory.

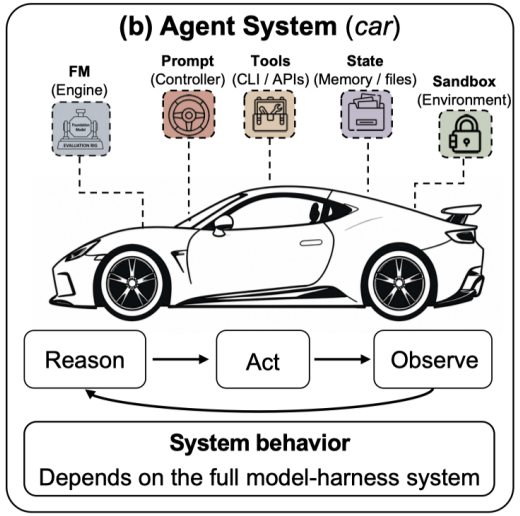
AlphaApollo: A System for Deep Agentic Reasoning. *ArXiv* preprint, 2026. https://bhanml.github.io/ & https://github.com/tmlr-group

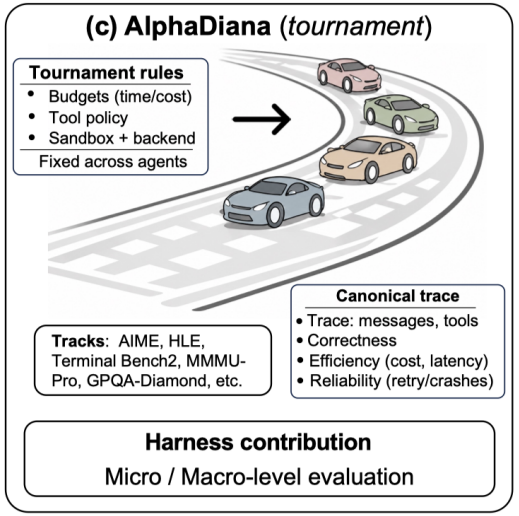
AlphaDiana: Agentic Evaluation System

AlphaDiana: A System for **Evaluating Reasoning Agents** such as OpenClaw.

- Reasoning is produced by the **interaction** of model, tools, memory, sandbox, etc.
- Evaluation must move from **scoring answers** to **measuring systems**.







Foundation models are evaluated as **engines**; Agents are **cars** shaped by tools and state; AlphaDiana is the **tournament** organizer that standardizes evaluation and records traces.

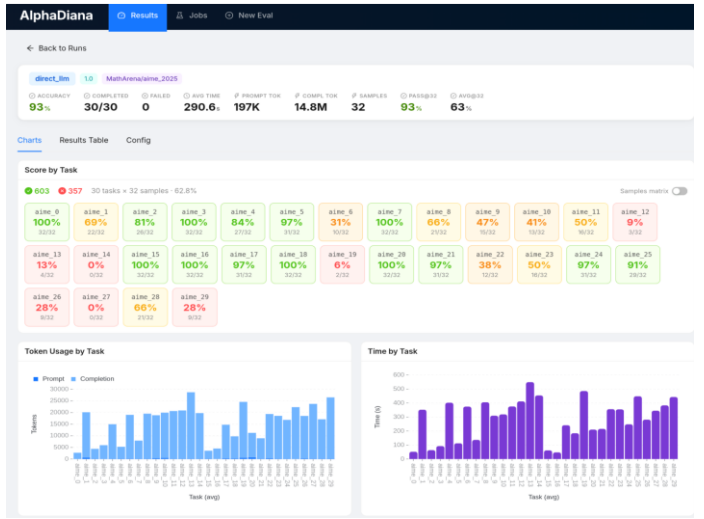
With AlphaDiana, we can evaluate OpenClaw on AIME benchmarks.

Qwen2.5-14B-Instruct

Benchmark	Avg@32 (Base)	Avg@32 (OpenClaw)	Pass@32 (Base)	Pass@32 (OpenClaw)
AIME 2024	0.1521	0.1271	0.4333	0.4000
AIME 2025	0.1229	0.1469	0.4000	0.4333
AIME 2026	0.1115	0.1250	0.4333	0.4333

GLM-5

Benchmark	Avg@32 (Base)	Avg@32 (OpenClaw)	Pass@32 (Base)	Pass@32 (OpenClaw)
AIME 2024	0.9000	0.8300	0.9330	1.0000
AIME 2025	0.6300	0.7600	0.9300	1.0000
AIME 2026	0.5719	0.3896	0.9000	0.9667



AlphaDiana has a web dashboard for launching and monitoring evaluation.

Code: https://github.com/tmlr-group/AlphaDiana

https://bhanml.github.io/ & https://github.com/tmlr-group

Take-Home Messages

Reasoning: Expand the model's capability through more deliberate internal generation.

- Prompting and test-time scaling shape inputs to elicit deliberate reasoning behaviors.
- Post-training fine-tunes LLMs to internalize the reasoning capability.

Calibration: Guide the model to remove undesirable knowledge and follow human values.

- Unlearning makes models forget certain knowledge such as private or outdated information.
- Alignment makes models follow human values such as ethics, intentions and safety.

Autonomy: Agents autonomously interact with and receive feedback from environments.

- Agentic Learning enables agents to learn through planning, action and observation.
- Agentic Evolution enables agents to improve through feedback, selection, and adaptation.

Join Us: TMLR Group

TMLR Group is always looking for

PhD/RA/Visiting students and Postdoc researchers to work on **trustworthy and efficient learning** and **reasoning algorithms, theories and systems**.

Our work involves:

- **Methodology:** Design algorithm and system to conduct trustworthy reasoning.
- **Evaluation:** Understand the boundary of reasoning capabilities.
- **Application:** Boost scientific discovery and solve real-world problems.

Email: bhanml@comp.hkbu.edu.hk
Location: Remote, Shenzhen, or Hong Kong

Application of Trustworthy Machine Reasoning

Methodology of Trustworthy Machine Reasoning

Evaluation of Trustworthy Machine Reasoning