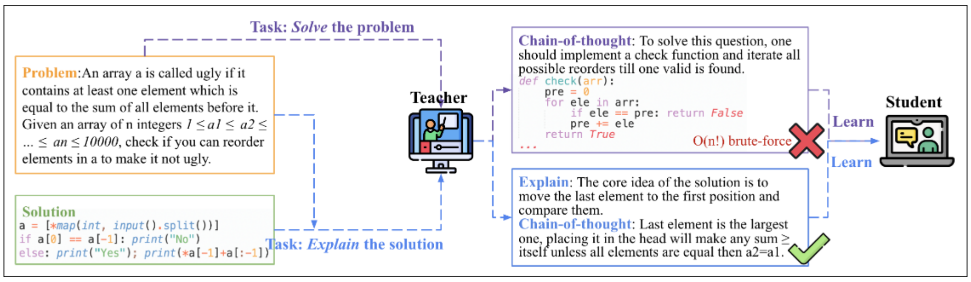
Literature Review: Reasoning Distillation Paradigms

1 Distilling Algorithmic Reasoning from LLMs via Explaining Solution Programs: april 2024

1.1 Conceptual Idea: Explain-based vs Solve-based Distillation

Traditional Chain-of-Thought (CoT) distillation typically follows a *solve-and-teach* paradigm: a teacher model solves a problem, and a student model learns from the teacher's reasoning trace (*solve-based*). However, on complex algorithmic tasks, even top-tier models (like GPT-4, as of April 2024) frequently generate incorrect or inefficient solutions (e.g., GPT-4 achieved only a 12% success rate on CodeContests), creating ”noise” for student training. The authors propose inverting this process: instead of solving the problem from scratch, the teacher (**Explainer**) is provided with the problem statement and an **oracle** (golden) solution code. Its task is to generate a detailed, structured analysis (**editorial**) of this correct solution. This approach guarantees high-quality training data, as the explanation is grounded in a verifiably correct algorithm. A compact **Reasoner** model is then trained on this data to generate high-level solution plans (hints) for new problems, rather than jumping straight to code implementation.



Problem:An array a is called ugly if it contains at least one element which is equal to the sum of all elements before it. Given an array of n integers $1 \le a1 \le a2 \le ... \le an \le 10000$, check if you can reorder elements in a to make it not ugly.

Solution
```
a = [*map(int, input().split())]
if a[0] == a[-1]: print("No")
else: print("Yes"); print(*a[-1]+a[:-1])
```

Chain-of-thought: To solve this question, one should implement a check function and iterate all possible reorders till one valid is found.
```
def check(arr):
    pre = 0
    for ele in arr:
        if ele == pre: return False
        pre += ele
    return True
...
```

Explain: The core idea of the solution is to move the last element to the first position and compare them.
Chain-of-thought: Last element is the largest one, placing it in the head will make any sum ≥ itself unless all elements are equal then a2=a1.

Figure 1: Comparison between Solve-based and Explain-based chain-of-thought distilling. Top: Solve-based CoT distilling is likely to generate incorrect or inefficient solutions. Bottom: Explain-based CoT distilling can generate high-quality reasoning processes by explaining the oracle solution.

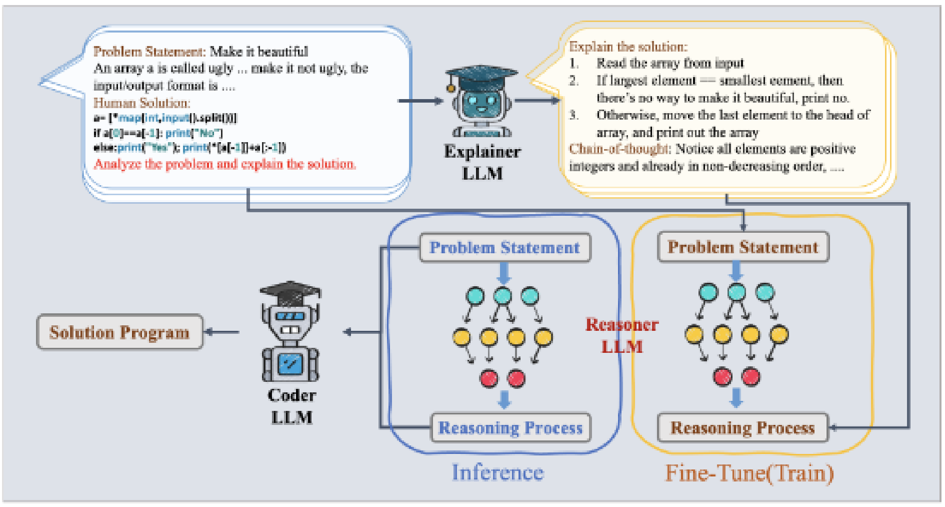
1.2 Technical Implementation: Reason-then-Implement Framework

The process is decomposed into three functional stages:

1. **Explainer** (based on GPT-4-0613): Annotates approximately 8,000 `<problem, solution>` pairs, generating structured ”silver” explanations. These include complexity analysis, conceptual evolution of the idea, key insights, and a step-by-step algorithm description.

2. **Reasoner** (Fine-tuned GPT-3.5 or DeepSeek-Coder-7B): This student model is fine-tuned on the `<problem, explanation>` pairs. It learns to generate textual ”hints” or reasoning plans given only the problem statement.

3. **Coder** (0-shot LLM): Takes the problem statement and the plan generated by the Reasoner as input to implement the final code.


Problem Statement: Make it beautiful
An array a is called ugly ... make it not ugly, the input/output format is ....
Human Solution:

```
a= [*map(int,input().split())]
if a[0]==a[-1]: print("No")
else:print("Yes"); print(*[a[-1]]+a[:-1])
```

Analyze the problem and explain the solution.

Explain the solution:
1. Read the array from input
2. If largest element == smallest eement, then there's no way to make it beautiful, print no.
3. Otherwise, move the last element to the head of array, and print out the array

Chain-of-thought: Notice all elements are positive integers and already in non-decreasing order, ...

Figure 2: The framework of our approach. We use Explainer LLM to generate explanations given `<problem, solution-program>` pairs; then train Reasoner LLM to generate explanations given problem statements. During inference time, given the problem, the Reasoner can generate a reasoning process in the same format as solution explanations, which could be provided to the Coder as a hint to solve the problem better.

1.3 Metrics and Results: solve@k

To evaluate performance, the authors adapt the **pass@k** metric into **solve@k**, which estimates the ”probability” that at least one of k generated programs passes all tests.

- **Formula:** solve@k $= \frac{1}{n}\sum_{i=1}^{n} P_i$, where $P_i = 1 - \frac{\binom{10-g_i}{k}}{\binom{10}{k}}$ (g_i is the number of successful programs out of 10 attempts).

- **Comparison:** Explain-based distillation demonstrated a significant increase in efficiency. On the CodeContests dataset, the relative improvement in solve@10 for DeepSeek Coder 7B was **+75%** over the strongest baseline. For GPT-3.5, the gain was **+69%** on the CF-Prob dataset. Using a Reasoner allows the Coder to avoid ”brute-force” strategies, which is critical for passing time-limit constraints (TLE). **HOWEVER:** absolute solve@10 values remain low (even the best result on CF Prob was 6.1% solve@10 on GPT-3.5).

1.4 Key Insights and Analysis

- **Diversity of Thought ¿ Diversity of Code:** With a fixed budget of $k = 10$ generations, it is far more effective to sample 10 different plans from the Reasoner and generate one code per plan ($M = 10, T = 1$) than to take one plan and generate 10 code variants ($M = 1, T = 10$). Diversity in logic yields a better solve rate (4.9% vs. 2.8%).

- **”Conceptual Evolution” vs. ”Step-by-Step”:** Ablation studies showed that the ”Conceptual Evolution” aspect (how the idea was derived) aids solve@1 the most by providing the correct strategic direction. However, the ”Step-by-Step Explanation” (implementation details) yields the best solve@10, as it minimizes implementation errors across multiple attempts.

- **Weighted Training:** A weighted fine-tuning strategy, prioritizing simpler and more recent problems (e.g., Python 3 vs. Python 2), yielded better results. This supports the hypothesis that training on overly complex or noisy explanations harms the student’s ”health.”

Note: Pay attention to the publication date (April 2024). It is likely that ”Explanation-based” distillation works well on hard tasks (*we will verify this*). It will be interesting to see how this approach performs with newer reasoners (*autumn 2025*). Subjective perception: The decomposition into Reasoner and Coder resembles the System 1 / System 2 architecture, where the Reasoner handles slow planning (à la Daniel Kahneman). A major drawback is the dependency on existing solutions to train the Reasoner, which may not happen in business cases.

+ There is a conclusion I liked: "it is better to sample the Reasoner," but trade-off curves are missing: how many prompt tokens/samples are needed for what solve@k.

2 ELAD: Explanation-Guided Large Language Models Active Distillation: acl findings 2024

2.1 Conceptual Idea: Active Distillation via Reasoning Verification

The authors of ELAD propose a paradigm of **active distillation**, guided by explanations. Instead of transferring knowledge on a fixed dataset, the framework iteratively selects the most informative examples based on the student's **reasoning uncertainty** (*see metric below*). In this scheme, the teacher (LLM) acts not just as an answer generator, but as a "*surgeon-reviewer*" who corrects the prefix of logical errors in the student's chains.

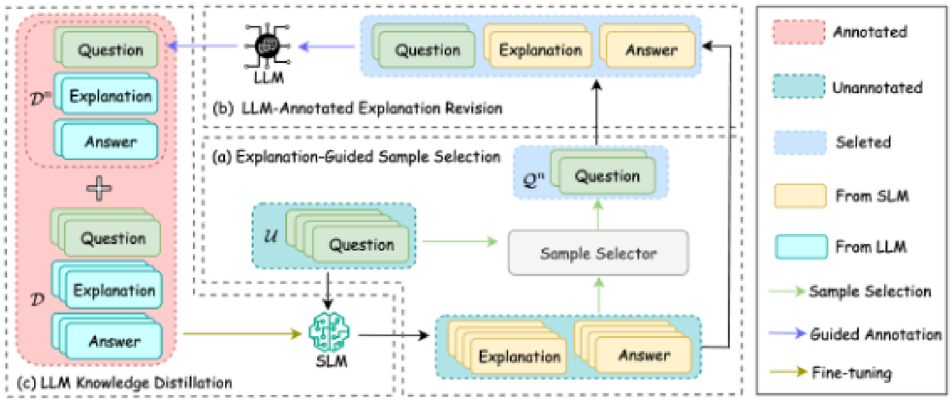


Figure 3: Overview of the Explanation-Guided LLM Active Distillation (ELAD) framework: (a) illustrates the Explanation-Guided Sample Selection method, (b) depicts the Customized LLM-Annotated Explanation Revision technique, and (c) showcases the LLM Knowledge Distillation (small model fine-tuning) process.

2.2 Technical Implementation: Uncertainty Metrics and DFS Revision

The framework is based on two key algorithmic innovations:

1. **EGSS (Explanation-Guided Sample Selection):** For selecting informative samples, a composite uncertainty metric $\mathcal{H}$ combining two levels is used:

   - **Inter-explanation uncertainty:** Estimated via Shannon entropy over the distribution of final answers across multiple sampling paths (k paths): $\mathcal{H}_{\text{Consistency}} = -\sum_{i=1}^{N} p_i \log(p_i)$.
   - **Intra-explanation uncertainty:** Measures the stability of the answer when individual reasoning steps are excluded. If removing step s_i changes the final answer, this step is critical and potentially unstable: $\mathcal{H}_{\text{Reasoning}} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(\hat{a}_i = a_i)$.

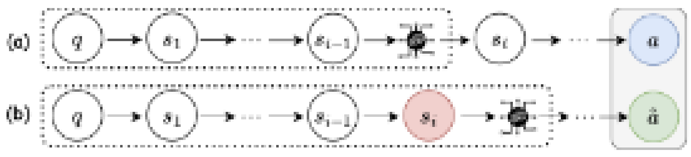


Figure 4: (a) illustrates reasoning not conditioned on the i-th reasoning step; (b) depicts reasoning conditioned on the i-th reasoning step

2. **CLAER (Customized LLM-Annotated Explanation Revision):** Instead of the teacher giving a new solution from scratch, a **Depth-First Search (DFS)** strategy is used to revise the student's chain. The teacher verifies the logic step-by-step ($s_1 \to s_2 \to \cdots \to s_n$). As soon as an error is detected (teacher reaction "No"), the chain is broken, and the teacher completes the correct continuation $\hat{s}_{\geq i}$ based on the student's context.

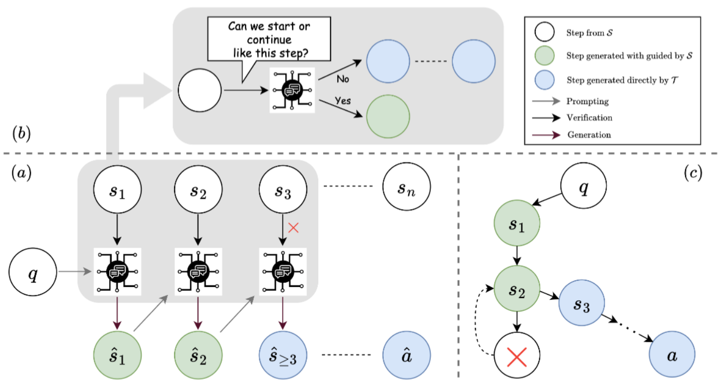


Figure 5: Customized LLM-Annotated Explanation Revision. (a) and (b) illustrate the process by which the LLM is prompted to revise the explanation and answer from the small model. (c) shows the DFS-based reasoning steps searching strategy

2.3 Metrics and Results: Efficiency under Limited Budget

- **Evaluation Metric:** Accuracy on GSM8K, AQuA (arithmetic), ANLI (NLI), and StrategyQA (Commonsense) tasks.
- **Comparison:** ELAD is compared with classic active learning methods (Maximum Entropy, Least Confidence) and standard CoT distillation.
- **Performance:** With an annotation budget of **50%** of the dataset, ELAD shows a relative solve-rate increase of up to **+75%** compared to baselines. On GSM8K, LLaMA-2-7B accuracy grew from 10% (zero-shot) to **32.7%** (after distillation via ELAD), which significantly outperforms random selection (28.4%).

2.4 Key Insights and Analysis

- **The "Rashomon Effect" Problem:** The authors emphasize that simply comparing the final explanation of the teacher and the student is ineffective, as different logical paths can lead to the correct answer. That is why CLAER uses DFS-verification of the student's current step, rather than forcing it to copy the teacher's style.
- **Dominance of Intra-uncertainty:** Ablation studies showed that estimating uncertainty within a single step (Intra) makes a critical contribution to data selection. Without the EGSS component, accuracy drops by 2-6%.
- **API Economy:** The CLAER method can be cheaper than standard CoT prompting, as the teacher often generates only short confirmation tokens ("Yes"), activating long text generation only upon error.

Note: It is interesting to find research where a small model determines its own competence boundaries. I think this should increase the sample efficiency of the training process.

Intra-uncertainty: directly counts the fraction of steps where answers matched. This looks more like stability/consistency than uncertainty. However, they further select examples with maximum H, i.e., large values are interpreted as "more informative/problematic". It is not entirely clear to me why "answer matching upon step removal" is specifically a signal of problematic reasoning (and not the opposite).

Intra-stepwise: To calculate intra-stepwise uncertainty, many additional student runs are needed. We count: for one question, n reasoning steps, for each step - at least two prompts ("with step/without step"), + k different reasoning trajectories for the inter-part. The paper compares by accuracy, but almost does not normalize the result by the total computational cost of EGSS. It is time to recall that one of the main goals in the paper is economy.

3 SUPERCORRECT: Advancing Small LLM Reasoning with Thought Template Distillation and Self-Correction: iclr 2025

3.1 Conceptual Idea: From Imitating Answers to Cloning the Cognitive Process

Consider the problem of small language models (SLM) — the inability to autonomously detect and correct their own logical errors in complex mathematical tasks. Existing methods (SFT, DPO) optimize either the final answer or the superficial structure of reasoning, without teaching reflection mechanisms. The authors of **SUPERCORRECT** propose a two-stage learning paradigm. Instead of simply teaching the model to "solve correctly," the framework teaches it to: 1) build reasoning hierarchically (from general strategies to details) and 2) correct errors using an external "correction signal" from a teacher.

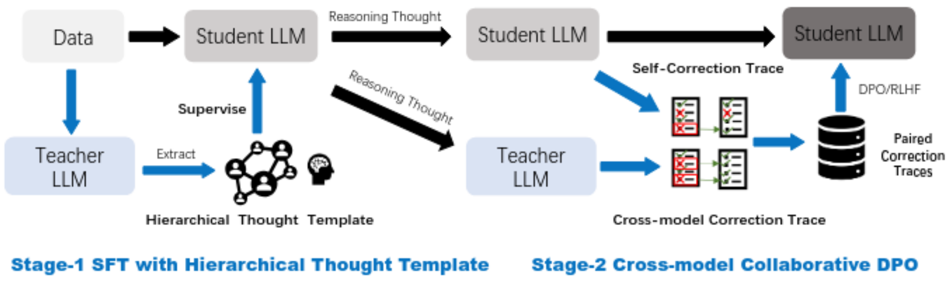


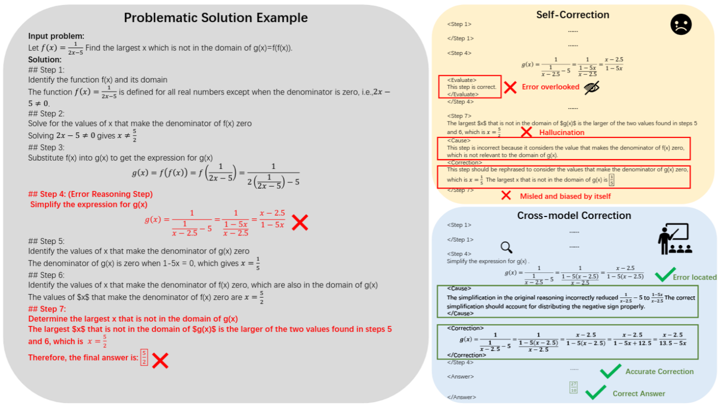
Figure 6: Overview of our proposed two-stage framework SUPERCORRECT. In the first stage, we extract hierarchical thought template from teacher LLM to supervise student LLM for producing more specific thoughts. In the second stage, we collect a dataset of paired self- and cross-correction traces for cross-model collaborative DPO.

3.2 Technical Implementation: Two-Stage Distillation

1. **Stage-1: Hierarchical Supervised Fine-Tuning (HSFT):** Instead of standard Chain-of-Thought (CoT), the model is trained on **hierarchical thought templates** extracted from SOTA models. The solution is broken down into `<Step>`, inside which `<Key>` (insights) and a final block `<Generalized>` (abstract strategy) are highlighted.

2. **Stage-2: Cross-model Collaborative DPO:** The model is trained on pairs of "teacher correction trajectory" vs "student erroneous reasoning". The teacher model finds the first erroneous step k. The **chosen** option is the teacher's correction (a_k^+, c_k^+), and the **rejected** one is the student's error step (a_k^-, c_k^-). Optimization function:

$$\mathcal{L}_{Cross-DPO} = -\mathbb{E}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(a_k^+, c_k^+|x, s_{<k})}{\pi_{ref}(a_k^+, c_k^+|x, s_{<k})} - \beta\log\frac{\pi_\theta(a_k^-, c_k^-|x, s_{<k})}{\pi_{ref}(a_k^-, c_k^-|x, s_{<k})}\right)\right]$$



Input problem:
Let $f(x) = \frac{1}{2x-5}$ Find the largest x which is not in the domain of g(x)=f(f(x)).
Solution:
Step 1:
Identify the function f(x) and its domain
The function $f(x) = \frac{1}{2x-5}$ is defined for all real numbers except when the denominator is zero, i.e.,$2x - 5 \neq 0$.
Step 2:
Solve for the values of x that make the denominator of f(x) zero
Solving $2x - 5 \neq 0$ gives $x \neq \frac{5}{2}$
Step 3:
Substitute f(x) into g(x) to get the expression for g(x)

$$g(x) = f(f(x)) = f\left(\frac{1}{2x-5}\right) = \frac{1}{2\left(\frac{1}{2x-5}\right) - 5}$$

Step 4: (Error Reasoning Step)
Simplify the expression for g(x)

$$g(x) = \frac{1}{\frac{1}{x-2.5} - 5} = \frac{1}{\frac{1-5x}{x-2.5}} = \frac{x-2.5}{1-5x}$$

Step 5:
Identify the values of x that make the denominator of g(x) zero
The denominator of g(x) is zero when 1-5x = 0, which gives $x = \frac{1}{5}$
Step 6:
Identify the values of x that make the denominator of f(x) zero, which are also in the domain of g(x)
The values of x that make the denominator of f(x) zero are $x = \frac{5}{2}$
Step 7:
Determine the largest x that is not in the domain of g(x)
The largest x that is not in the domain of $g(x)$ is the larger of the two values found in steps 5 and 6, which is $x = \frac{5}{2}$
Therefore, the final answer is: $\boxed{\frac{5}{2}}$

<Step 1>
......
</Step 1>
......
<Step 4>

$$g(x) = \frac{1}{\frac{1}{x-2.5} - 5} = \frac{1}{\frac{1-5x}{x-2.5}} = \frac{x-2.5}{1-5x}$$

<Evaluate>
This step is correct.
</Evaluate>
</Step 4>
......
<Step 7>
The largest x that is not in the domain of $g(x)$ is the larger of the two values found in steps 5 and 6, which is $x = \frac{5}{2}$
<Cause>
This step is incorrect because it considers the value that makes the denominator of f(x) zero, which is not relevant to the domain of g(x).
<Correction>
This step should be rephrased to consider the values that make the denominator of g(x) zero, which is $x = \frac{1}{5}$ The largest x that is not in the domain of g(x) is $\boxed{\frac{1}{5}}$
</Step 7>

<Step 1>
......
</Step 1>
......
<Step 4>
Simplify the expression for g(x) .

$$g(x) = \frac{1}{\frac{1}{x-2.5} - 5} = \frac{1}{\frac{1-5(x-2.5)}{x-2.5}} = \frac{x-2.5}{1-5(x-2.5)}$$

<Cause>
The simplification in the original reasoning incorrectly reduced $\frac{1}{x-2.5} - 5$ to $\frac{1-5x}{x-2.5}$ The correct simplification should account for distributing the negative sign properly.
</Cause>
<Correction>

$$g(x) = \frac{1}{\frac{1}{x-2.5} - 5} = \frac{1}{\frac{1-5(x-2.5)}{x-2.5}} = \frac{x-2.5}{1-5(x-2.5)} = \frac{x-2.5}{1-5x+12.5} = \frac{x-2.5}{13.5-5x}$$

</Correction>
</Step 4>
......
<Answer>
$\boxed{\frac{27}{10}}$
</Answer>

Figure 7: An illustrative comparison between self-correction and our cross-model correction. Cross-model correction can enable more precise error localization and thought correction.

3.3 Metrics and Results: New SOTA in 7B Class

- **Performance:** The **SUPERCORRECT-7B** model achieved **70.2% on MATH** and **89.5% on GSM8K**, which is an absolute record for models of this size.
- **Comparison:** The model outperforms Qwen2.5-Math-7B by **15.1%** (in absolute values on MATH) and the powerful DeepSeekMath-7B by **7.8%**.
- **Stability:** Analysis of the answer distribution showed that SUPERCORRECT significantly reduces result variance in multiple sampling.

3.4 Key Insights and Analysis

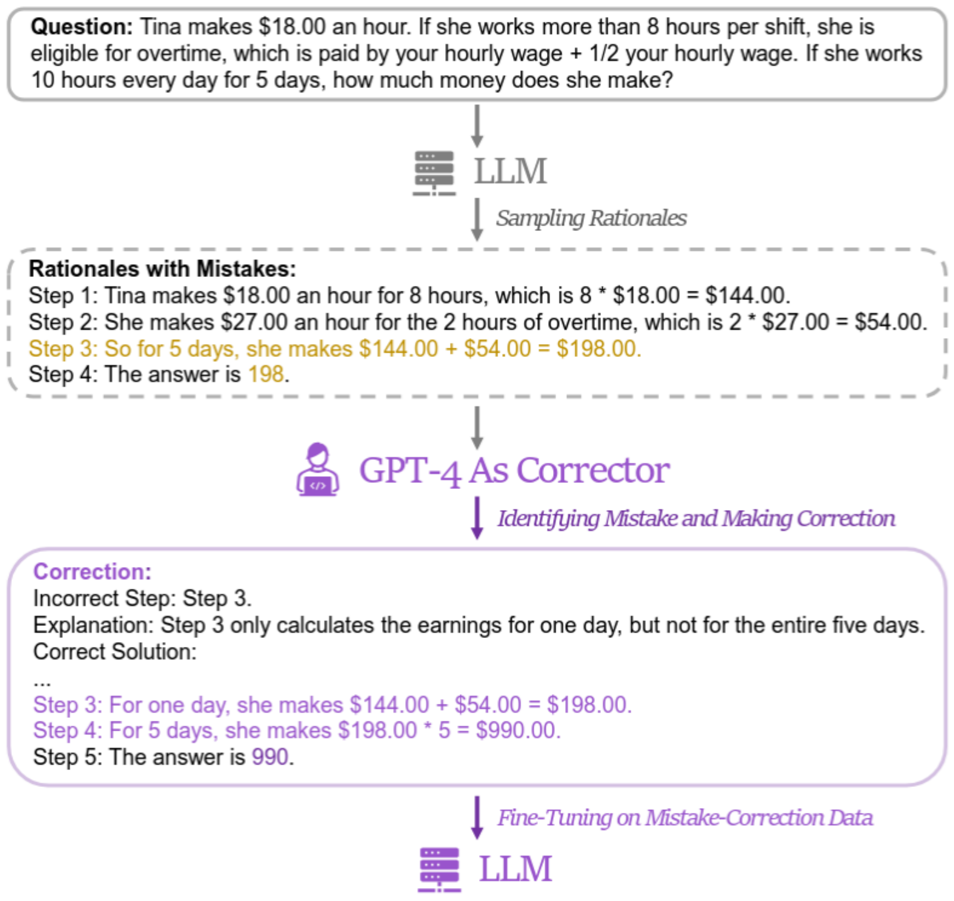
- **Breaking the "Thought Bottleneck":** Training on teacher correction traces allows the student to master new mathematical tricks that it could not generate itself, overcoming hallucination bias.
- **XML as a Regularizer:** Using XML tags helps the model better separate "planning" and "calculation," which is critical for small models.
- **Topic-specific gains:** The largest increase (up to +23.7%) is observed in the most difficult topics ("Precalculus", "Number Theory").

Note: Verify: there is a hypothesis that for small models, the presence of meta-information on *exactly how* to correct logical failures is important. Cross-model DPO at the level of individual correction steps is more effective than regular RLHF in tasks with clear logic.

4 Learning From Mistakes Makes LLM Better Reasoner (LEMA): march 2024

4.1 Conceptual Idea: Learning via Retrospective Error Analysis

SFT LLM relies mainly on "positive reinforcement". However, this approach ignores a fundamental aspect of human cognition: learning from mistakes. The authors present the **LEMA** framework, which mimics a student's work on mistakes. The model learns to identify an incorrect step, explain the reason for the failure, and generate a corrected trajectory.


Question: Tina makes $18.00 an hour. If she works more than 8 hours per shift, she is eligible for overtime, which is paid by your hourly wage + 1/2 your hourly wage. If she works 10 hours every day for 5 days, how much money does she make?

Rationales with Mistakes:
Step 1: Tina makes $18.00 an hour for 8 hours, which is 8 * $18.00 = $144.00.
Step 2: She makes $27.00 an hour for the 2 hours of overtime, which is 2 * $27.00 = $54.00.
Step 3: So for 5 days, she makes $144.00 + $54.00 = $198.00.
Step 4: The answer is 198.

Correction:
Incorrect Step: Step 3.
Explanation: Step 3 only calculates the earnings for one day, but not for the entire five days.
Correct Solution:
...
Step 3: For one day, she makes $144.00 + $54.00 = $198.00.
Step 4: For 5 days, she makes $198.00 * 5 = $990.00.
Step 5: The answer is 990.

LLM
Sampling Rationales
GPT-4 As Corrector
Identifying Mistake and Making Correction
Fine-Tuning on Mistake-Correction Data
LLM

Figure 8: Process of LEarning from MistAkes (LEMA).

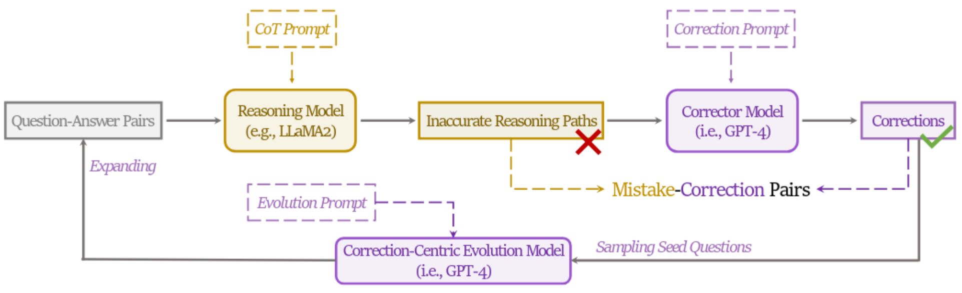

CoT Prompt
Correction Prompt
Question-Answer Pairs
Reasoning Model (e.g., LLaMA2)
Inaccurate Reasoning Paths
Corrector Model (i.e., GPT-4)
Corrections
Expanding
Mistake-Correction Pairs
Evolution Prompt
Correction-Centric Evolution Model (i.e., GPT-4)
Sampling Seed Questions

Figure 9: Process of generating and expanding correction data.

4.2 Technical Implementation: Correction Pipelines and Data Evolution

The methodology consists of three stages:

1. **Generating "Failures":** Sampling inaccurate reasoning paths from various models (LLaMA-2, WizardLM, GPT-3.5) to obtain a spectrum of errors.

2. **GPT-4 as "Tutor-Corrector":** Annotating errors in a triplet format: {*Incorrect Step, Explanation of Error, Correct Solution*}.

3. **Correction-Centric Evolution:** A dataset expansion strategy focused on "moderately difficult" problems. Evolution of overly simple or overly complex tasks is recognized as ineffective.

4.3 Metrics and Results: Effectiveness of Learning from Mistakes

- **Performance:** LEMA consistently outperforms classic CoT fine-tuning. LLaMA-2-70B improved the result on **GSM8K from 81.4% to 83.5%**, and on **MATH from 23.6% to 25.0%**.
- **PPL Analysis:** Models trained via LEMA show a larger difference in perplexity (ΔPPL) between erroneous and correct CoT, which indicates a better "sense of error."
- **Data Heterogeneity:** Mixing CoT and Correction data yields better results than simply increasing the volume of pure CoT data.

4.4 Key Insights and Analysis

- **Scale Effect:** Strong models (backbone) benefit more from LEMA than weak ones (e.g., 7B).
- **QLoRA Issue:** QLoRA copes poorly with large volumes of correction data; full fine-tuning is required.
- **GPT-4 Self-Correction:** GPT-4 is able to correct its own errors only in **8.0% of cases** on the MATH dataset, which emphasizes the complexity of the correction task.

Note: I did not quite understand the take on extracting signal specifically from **errors**. It looks like the metric gain is driven *(and proportional up to fine-tuning)* by adding correct reasoning traces. It is required to see how much the **"error/explanation"** signal is more important than the corrected solution itself, which is essentially another "correct CoT" (just with error context)? I consider this note especially important, as we are guided by similar logic in branch C during distillation.

Contrastive Loss \Bidirectional Learning – need to find application for them in our paper.

5 Reverse-Engineered Reasoning for Open-Ended Generation (REER): september 2025

5.1 Conceptual Idea: Inversion of Deep Reasoning Paradigm

Important: the paper relates to more "creative" tasks. Keep this in mind. Popular methods (RL, distillation) are weak in open Creative Writing tasks due to the lack of a normal (objective) reward. The authors propose **REER**: instead of training the model to reason "forward," the process works **"backward."** A reference text (oracle solution) is taken, and a hidden thinking process that would most likely lead to such a result is **computationally restored** (reverse-engineered) for it. The quality of reasoning is evaluated via the **perplexity (perplexity)** of the reference text.

Formalization of reasoning quality via perplexity of a known good answer: Quality of z is measured by how low PPL(y — x, z) becomes for the generator LLM*having (x=query, y=good answer), search for z=thought trajectory that "explains" y.*

Formalization of reasoning quality via perplexity of a known good answer: Quality of z is measured by how low PPL(y — x, z) becomes for the generator LLM

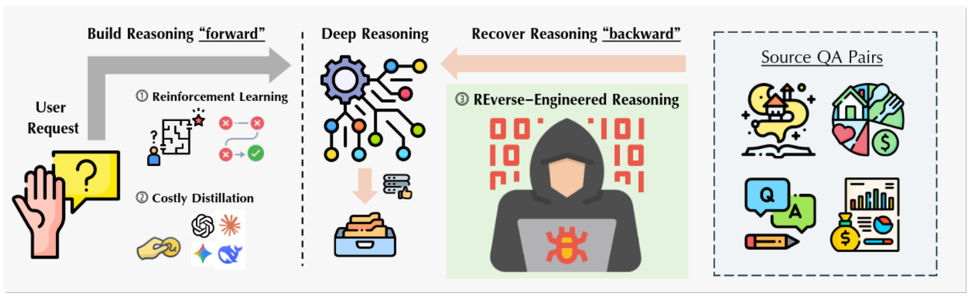


Figure 10: (Left) Existing methods attempt to build deep reasoning "forwards" for a user request through trial-and-error (RL) or costly distillation, which falter in open-ended domains that lack clear, verifiable reward signals. (Right) We propose a third path for teaching deep reasoning, REverse-Engineered Reasoning (REER). REER works "backwards", recovering plausible human-like thought process from known-good outputs in open-source Question-Answer (QA) pairs

5.2 Technical Implementation: Searching in Latent Reasoning Space

The process is formalized as a gradient-free optimization problem:

1. ***Mathematical Formulation:*** *A trajectory* $z^* = \arg\min_{z \in Z} PPL(y|x,z)$ *is sought, minimizing the conditional perplexity of the reference answer* y.

2. ***Iterative Refinement:*** *Local search, which starts with a draft plan and refines it segment by segment, choosing variants that lower the PPL of the standard.*

3. ***Context Engineering:*** *Introducing human thinking patterns ("Hmm...", "Alternatively...") into prompts.*

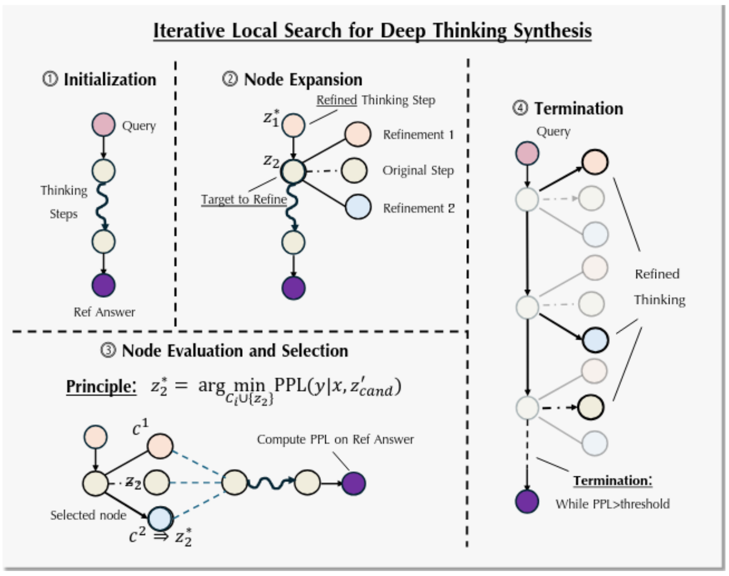


Figure 11: Method Overview: Iterative Local Search for deep reasoning Synthesis

5.3 Metrics and Results: DeepWriter-8B

- ***Dataset:*** *DeepWriting-20K — 20,000 trajectories for creative tasks.*

- ***Performance:*** *The **DeepWriter-8B** model (based on Qwen3) surpassed specialized open-source solutions and showed results comparable to **GPT-4o** and **Claude 3.5** in LongBench-Write and HelloBench tasks.*

- ***Qualitative Profile:*** *Anomalously high indicators in problem deconstruction and logical connectivity when generating long texts.*

5.4 Key Insights and Analysis

- ***PPL as an Ideal Proxy:*** *Perplexity reliably predicts the "explanatory power" of reasoning for creative tasks.*
- ***Trajectory Length:*** *Removing long reasoning significantly reduces quality in professional writing but is less critical for pure creativity.*
- ***"Backward" Advantage:*** *The method allows using any existing texts (e.g., literature) as training data, generating ideal CoT traces for them.*

Note: *I don't quite understand why perplexity was chosen as a proxy. After all, the initial problem statement was exactly about the model's creative reflections, while minimizing PPL means making y (answer) predictable for a specific generator LLM, without necessarily making z (trajectory):*

- *causally plausible reasoning.*
- *universally useful planning*
- *and/or "humanly correct" explanation.*

Yes, the authors intuitively interpret this as "good z makes good y logical."

But I would like to see correlation validation: to what extent does PPL reduction actually correlate with human assessment of "thought" quality and quality after SFT? The article has qualitative graphs (Figure 4), but no, for example, strict correlation table "Δ PPL vs win-rate / judge-score".

Also, I have suspicions of leakage / peeking at the answer risk, as the full reference y is used in the search when generating segment edits (they explicitly pass y into the segment-editing prompt). The authors say that the prompt "prevents copying" y. But without a quantitative metric (e.g., overlap / rouge-L / n-gram match between z and y), there remains a risk that part of z becomes a paraphrased summary of the answer, rather than a "plan".