

Teaching by Learning: Harnessing LLMs Collaboration for Adaptive Data Synthesis via Cognitive Load Feedback

Yue Luo¹, Feng Zhao^{1*}, Yu Yang², Guandong Xu²

¹Natural Language Processing and Knowledge Graph Lab

School of Computer Science and Technology

Huazhong University of Science and Technology, Wuhan, China

²Centre for Learning, Teaching and Technology

The Education University of Hong Kong, Hong Kong SAR, China

{luoyue1, zhaof}@hust.edu.cn

{yangyy, gdxu}@eduhk.hk

Abstract

Recent advancements in Large Language Models (LLMs) are increasingly driven by using LLMs themselves to synthesize large amounts of instruction-tuning data. To transcend the limitations of single model synthesis, recent research has shifted toward multiple model data synthesis, where distinct models assume specialized roles to synthesize data through interaction. However, existing feedback cannot distinguish truly high-quality instructions, those that promote complex reasoning, deviate from the learned distribution, and require multi-source knowledge, due to coarse granularity and signal dilution. In human cognition, truly high-quality instructions require greater mental effort—the **Cognitive Load**—to address. In this paper, we propose a paradigm shift in feedback. Instead of external, result-oriented evaluation, we propose an internal, process-aware feedback based on target models’ computational effort. The novel feedback, namely **Inference Stress Index (ISI)**, quantifies instruction quality in a fine-grained, dilution-resistant manner by focusing on high-uncertainty tokens. Building on this, we present the **Cognitive-Feedback Adaptive Generation (CoFAG)** framework, a self-evolving multiple model collaboration paradigm involving a synthesizer model, a fixer model, and a collaborator model. Extensive experiments demonstrate that CoFAG significantly outperforms other data synthesis techniques on complex reasoning benchmarks, validating the superiority of using internal cognitive feedback for generating high-quality data.

1 Introduction

In recent years, Large Language Models (LLMs) have achieved breakthrough advances in capabilities. They demonstrate remarkable performance on diverse tasks, including math reasoning, code

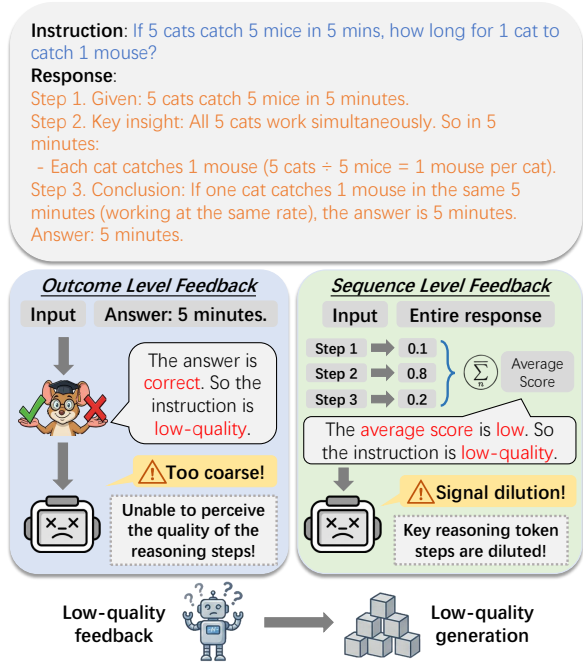


Figure 1: Summary of existing feedback mechanisms. Outcome-Level Feedback focuses solely on the final result. Sequence-Level Feedback averages over the entire token sequence, diluting the signal from critical tokens. Low-quality feedback led to low-quality data synthesis.

generation, and medical question answering (Yang et al., 2025; OpenAI, 2023; Dubey et al., 2024). This success is largely attributed to training LLMs on large-scale instruction datasets. However, manually constructing these datasets is prohibitively expensive and hard to scale. This challenge has prompted the academic community to adopt a new paradigm: leveraging LLMs to synthesize instruction datasets (Long et al., 2024; Wang et al., 2023). Early approaches relied on single-model synthesis, where a model attempts to self-improve via iterative prompting. However, these methods often suffer from mode collapse and self-delusion. A model rarely generates data that exceeds its own

*Corresponding author.

intrinsic knowledge boundaries (Tan et al., 2024).

To transcend these limitations, recent research has shifted towards multiple model collaboration (Li et al., 2025b; Lee et al., 2024). In this paradigm, distinct models assume specialized roles to synthesize data through interaction. A generator model creates instructions, and a target model attempts them. The core challenge lies in the design of the feedback mechanism: how should the target model signal the generator to synthesize better data? We argue that truly high-quality instructions share three key characteristics. They (1) promote complex reasoning, (2) deviate from the model’s learned distribution, and (3) require multi-source knowledge integration (Ye et al., 2025). As shown in Figure 1, existing feedback mechanisms fail to capture these characteristics. Outcome-level feedback methods (Li et al., 2025b; Lee et al., 2024) rely on binary correctness signals. These signals are too coarse-grained to perceive the quality of reasoning steps. Sequence-level feedback methods (Li et al., 2024a,b, 2025c) use metrics like perplexity, instruction-following difficulty, or influence functions. These metrics are typically aggregated by averaging over all tokens. While averaging ensures length-invariance, it dilutes the signal from critical reasoning tokens. A genuinely challenging instruction may receive a low score because high-impact tokens are overwhelmed by low score tokens.

These feedback signals fail to accurately pinpoint the truly crucial parts of the inference process, thus failing to properly guide the optimization of the synthesis strategy. This ultimately leads to low-quality data synthesis. To address this, we draw inspiration from human psychology. In cognitive load theory (Sweller, 1988; Sweller et al., 1998), **Cognitive Load** refers to the mental effort imposed on working memory during task execution. We observe a direct analogy to LLMs. When an instruction compels a model to perform complex logical inference, handle unfamiliar patterns, or integrate multi-source knowledge, its internal computational process becomes significantly more effortful.

Building on this insight, we propose a paradigm shift. Instead of external, result-oriented evaluation, we advocate for an internal, process-aware feedback mechanism. This mechanism is based on the target model’s computational effort. We formalize this as the **Inference Stress Index (ISI)**, a novel metric that quantifies the model’s cognitive load at a token-by-token level. Sequence-level

metrics suffer from averaging dilution. In contrast, ISI aggregates token-wise uncertainty in a dilution-resistant manner. It focuses on the top fraction of high-uncertainty tokens. This design preserves the signal from critical reasoning steps.

We operationalize this paradigm through the **Cognitive-Feedback Adaptive Generation (CoFAG)** framework. CoFAG is a self-evolving multiple model collaboration system. It involves three specialized roles: a synthesizer model, a fixer model, and a collaborator model. The synthesizer model generates candidate instructions. The fixer model provides cognitive feedback via ISI. The collaborator model supplies reference responses. CoFAG uses the fixer model’s cognitive load to classify newly generated instructions. It then optimizes the synthesizer model to learn the generation patterns of high-cognitive-load instructions. This iterative refinement compels the synthesizer to master the generation of high-quality instructions. As a result, the fixer model obtains an efficient learning trajectory.

Our contributions are summarized as follows:

- **A New Paradigm for Instruction Quality.** We are the first to propose and formalize the target model’s internal cognitive load as a fine-grained proxy for instruction quality in the context of multiple model collaborative data synthesis. We devise ISI to quantify cognitive load, shifting feedback from coarse, result-oriented signals to process-aware measures.
- **Cognitive-Feedback Adaptive Generation Framework.** We introduce CoFAG, a novel self-evolving framework that operationalizes our paradigm through a symbiotic three-model collaboration, eliminating the need for external binary feedback or manual curation.
- **Empirical Validation.** Extensive experiments demonstrate that CoFAG significantly enhances target model capabilities on complex reasoning benchmarks. On MATH, CoFAG boosts accuracy to 62.30%, outperforming the strongest baseline (61.32%).

2 Related Work

2.1 Efficient LLM Alignment

The generation of high-quality instruction datasets aims at facilitating the efficient alignment of LLMs. A key insight in this field is the philosophy “less is

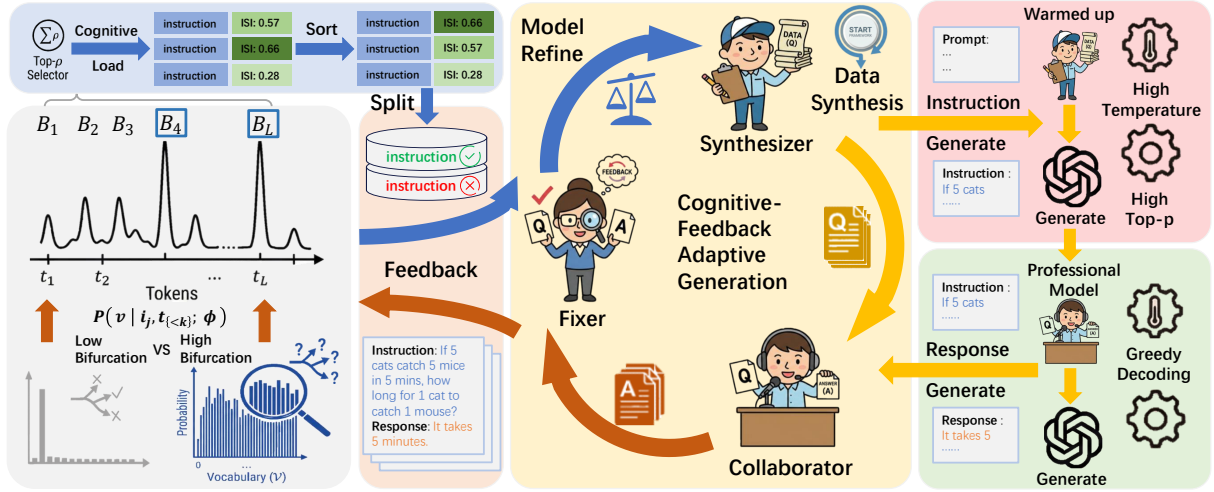


Figure 2: An overview of the Cognitive-Feedback Adaptive Generation (CoFAG) framework, which simulates a dynamic, closed-loop collaboration among a synthesizer model, a fixer model, and a collaborator model.

more”. LIMA (Zhou et al., 2023) demonstrates that fine-tuning a model on a carefully curated small dataset of 1,000 instructions is sufficient to exhibit strong performance. Similarly, LIMO (Ye et al., 2025) shows that even a small number of high-quality data can elicit complex mathematical reasoning abilities in models. These studies highlight the importance of data quality over quantity in efficient alignment and capability elicitation. Our work resonates with this principle, focusing on generating high-quality instruction data tailored to the learning frontiers of target models.

2.2 LLM-Driven Data Synthesis

The emergence of LLMs has sparked significant interest in exploring their potential for high-quality data synthesis (Tan et al., 2024; Schick and Schütze, 2021). Synthetic data is regarded as an effective data augmentation strategy to expand existing datasets and enrich data assets (Wang et al., 2024; Chang et al., 2024). Many methods leverage the in-context learning capabilities of LLMs, or adopt iterative self-generation techniques like Self-Instruct (Wang et al., 2023) and Evol-Instruct (Xu et al., 2024). However, single-model synthesis often suffers from mode collapse and self-delusion (Tan et al., 2024). Recent research has shifted towards multiple model collaboration, where distinct models assume specialized roles through interaction (Li et al., 2025b; Lee et al., 2024; Li et al., 2025c; Yuan et al., 2024). Existing feedback mechanisms either rely on coarse-grained outcome-level correctness signals or sequence-level metrics like perplexity, yet neither can precisely pinpoint the

truly critical parts within the reasoning process. Our work proposes a paradigm shift from these external evaluations to an internal, process-aware measure based on the target model’s cognitive load.

3 Methods

3.1 Task Formulations

Given a seed dataset $D_{\text{seed}} = \{(i_j, r_j)\}_{j=1}^N$ of size N , our goal is to synthesize a high-quality instruction dataset via multiple model collaboration and iteratively optimize the generator models through feedback. Here, i_j denotes the j -th instruction and r_j denotes its corresponding response.

In each $\text{round} \in [1, R]$, we construct a high-quality instruction dataset $D_{\text{high_quality}}^{(\text{round})}$ of size N' and accumulate it as:

$$D_{\text{accumulated}}^{(\text{round})} = \bigcup_{j=1}^{\text{round}} D_{\text{high_quality}}^{(j)}. \quad (1)$$

The target model at round , is obtained by fine-tuning the initial target model on $D_{\text{accumulated}}^{(\text{round})}$. The generator model at round is derived by optimizing the generator model from $\text{round} - 1$ via preference learning.

Across R rounds, we obtain an optimized generator model, an enhanced target model, and an accumulated synthetic dataset. This design promotes rapid improvement with a small number of rounds and modest synthetic data sizes.

3.2 Framework Overview

We propose **CoFAG**, which is illustrated in Figure 2. CoFAG is a collaborative data synthesis

framework that operationalizes a closed-loop multiple model collaboration process. We consider a collaborative setting with three specialized roles:

Synthesizer: an LLM that proposes candidate instructions. Critically, it dynamically adjusts its instruction generation strategy based on feedback from the fixer model.

Fixer: an LLM that executes instructions, exposes its cognitive state as feedback, and guides the refinement of the synthesizer’s generation.

Collaborator: an LLM that provides reference responses for the synthesized instructions.

CoFAG leverages the cognitive load of the fixer model to provide fine-grained signals for optimizing the synthesizer model’s generation strategy. CoFAG adopts token-level cognitive signals and aggregates them in a dilution-resistant manner. Concretely, we measure token-wise cognitive bifurcation in the fixer model’s decoding process and compute the Inference Stress Index (ISI) by focusing on the top fraction of high-bifurcation tokens.

Prior to entering the iterative loop, the **Model Preparation** step performs initial capability alignment among the Synthesizer and Fixer models via Supervised Fine-Tuning (SFT). Subsequently, the framework proceeds through three steps repeated over rounds: **Data Synthesis**, **Cognitive Feedback**, and **Model Refinement**, forming a self-evolving loop. The following subsections describe each step in detail.

3.3 Model Preparation

Before entering the dynamic collaboration loop, initial capability alignment among the models is essential. We employ Supervised Fine-Tuning (SFT) to achieve this goal.

Specifically, we first perform SFT on the synthesizer model with data pairs from the seed dataset $D_{\text{seed}} = \{(i_j, r_j)\}_{j=1}^N$. The prompts in the synthesizer model’s training data are fixed task-designed statements P , such as “Generate a mathematical problem”, with labels being the instructions i_j from the seed dataset. This process enables the synthesizer model to learn to generate instructions relevant to the target task domain, formalized as the optimization objective:

$$\max_{\theta} \sum_{(i) \in D_{\text{seed}}} \log P(i \mid P; \theta) \quad (2)$$

Subsequently, the preliminarily fine-tuned synthesizer model generates a small-scale instruction

dataset, which is used to perform SFT on the fixer model. This warm-up step equips the fixer model with initial capabilities to understand and respond to task-related instructions. Crucially, it also aligns the fixer model’s output style with the standard format. This ensures that the subsequent ISI calculation purely reflects the cognitive load of reasoning.

3.4 Data Synthesis

In each iteration round, the synthesizer model generates candidate instructions based on the task-designed prompt P . Formally, given the current synthesizer model $\mathcal{S}^{(\text{round})}$, we sample a set of instructions:

$$I^{(\text{round})} = \{i_j\}_{j=1}^M, \quad i_j \sim P_{\mathcal{S}^{(\text{round})}}(\cdot \mid P). \quad (3)$$

Subsequently, the collaborator model provides reference responses for each synthesized instruction. For each instruction i_j , the collaborator model generates a corresponding response r_j , forming the test dataset:

$$D_{\text{test}}^{(\text{round})} = \{(i_j, r_j)\}_{j=1}^M \quad (4)$$

where each response r_j is a token sequence $(t_1, \dots, t_{L_j})$.

3.5 Cognitive Feedback

This step converts the fixer model’s cognitive state into supervision for optimizing the synthesizer model. Given the synthesized data $D_{\text{test}}^{(\text{round})}$, the fixer model processes each pair and exposes a token-level score. We aggregate the token scores as an instruction’s cognitive load.

Specifically, let $\mathcal{V}$ be the vocabulary. At decoding step k , given instruction i_j and context $t_{<k}$, the fixer model produces $P(\cdot \mid i_j, t_{<k}; \phi)$. We define the instantaneous **Cognitive Bifurcation** as the entropy of this distribution:

$$\mathcal{B}_k = - \sum_{v \in \mathcal{V}} P(v \mid i_j, t_{<k}; \phi) \log P(v \mid i_j, t_{<k}; \phi). \quad (5)$$

High $\mathcal{B}_k$ indicates that the fixer model faces multiple plausible reasoning continuations or must retrieve non-trivial knowledge.

Then, we aggregate these token-level signals into a sequence-level score, namely Inference Stress Index (ISI), which quantifies the overall cognitive load of a sample. To avoid diluting the impact of truly difficult reasoning tokens in long outputs, we adopt a Pareto-style aggregation: we compute ISI

using only the top ρ fraction of Cognitive Bifurcation values. Following the insight from 80/20 rule (Wang et al., 2025), we set $\rho = 0.2$ in all experiments. Formally, let $\{\mathcal{B}_k\}_{k=1}^{L_j}$ be the bifurcation sequence for sample j , and let π_j be a permutation that sorts the bifurcations in descending order. Define $m_j = \max(1, \lceil \rho L_j \rceil)$. We then define the ISI as:

$$\text{ISI}(i_j; \phi, \rho) = \frac{1}{m_j} \sum_{k=1}^{m_j} \mathcal{B}_{\pi_j(k)}. \quad (6)$$

Finally, we treat $\text{ISI}(i_j; \phi, \rho)$ as a reward signal for instruction quality (higher indicates more cognitively demanding instructions). We rank instructions by ISI and construct preference pairs by selecting the top 50% as positive examples i_w and the bottom 50% as negative examples i_l . Each pair (i_w, i_l) is stored in the preference dataset $D_{\text{pref}}^{(\text{round})}$.

3.6 Model Refinement

This step leverages the cognitive feedback to refine both the synthesizer model and the fixer model.

We adopt the Direct Preference Optimization (DPO) (Rafailov et al., 2023) algorithm to refine the instruction generation strategy of the synthesizer model using the preference data $D_{\text{pref}}^{(\text{round})}$. The loss function is defined as:

$$\mathcal{L}(S; S_{\text{ref}}) = -\mathbb{E}_{(i_w, i_l) \sim D_{\text{pref}}} \left[\log \sigma \left(\beta \log \frac{P_S(i_w)}{P_{S_{\text{ref}}}(i_w)} - \beta \log \frac{P_S(i_l)}{P_{S_{\text{ref}}}(i_l)} \right) \right] \quad (7)$$

where S is the synthesizer model to be optimized, S_{ref} is a copy of the pre-optimization synthesizer model, β is a hyperparameter controlling the degree of policy deviation, and σ denotes the Sigmoid function. The newly optimized synthesizer model is employed to generate the final high-quality training data $D_{\text{high_quality}}^{(\text{round})}$ for this iteration round.

The fixer model is trained on the accumulated data $D_{\text{accumulated}}^{(\text{round})} = \bigcup_{j=1}^{\text{round}} D_{\text{high_quality}}^{(j)}$, where the responses are provided by the collaborator model. This enables the fixer model to progressively improve its capabilities, providing more accurate cognitive feedback in subsequent rounds.

4 Experiments

4.1 Experimental Settings

Datasets. We selected five seed datasets from the mathematics, legal, and medical domains to evaluate the improvement of our method’s synthetic data for LLMs. We selected three distinct mathematics

datasets, each representing a different level of problem complexity. For a simple mathematical reasoning scenario, we utilized **GSM8K** (Cobbe et al., 2021), a primary school math problem dataset. To simulate a medium-difficulty scenario, we leveraged the mathematics-focused subset of **Infinity-Instruct** (Li et al., 2025a), a large-scale instruction tuning dataset. To test performance on complex and challenging problems, we employed **MATH** (Hendrycks et al., 2021), a dataset composed of competition-level mathematics problems designed to rigorously assess advanced problem-solving abilities. For the legal domain, we incorporated **Olaw** (Lab, 2024), a dataset comprising legal questions and case analyses to test legal reasoning and comprehension. For the medical field, we used **medical-o1-reasoning-SFT** (Chen et al., 2024), a collection of medical case studies and reasoning questions aimed at evaluating the model’s clinical reasoning and application of medical knowledge.

Evaluation Metrics. Across all experiments, we adopt **Accuracy** as the evaluation metric. We use an LLM judge (Qwen-plus-latest) to assess if the model’s generated response is semantically equivalent to the ground-truth answer. This approach eliminates the format mismatch problem between the ground-truth answers in test datasets and the response format learned by the fixer model. The accuracy is formally calculated as follows:

$$\text{Accuracy} = \frac{1}{N} \sum_{i=1}^N \mathbb{I}(\text{response}_i \equiv \text{answer}_i) \quad (8)$$

where $\mathbb{I}(\cdot)$ is the indicator function, which returns 1 if the judge deems the model’s response and the ground-truth answer equivalent, and 0 otherwise. The experimental demonstration will show the maximum value from multiple experiments.

Baselines. To benchmark the performance of our proposed **CoFAG** framework, we compare it with several data synthesis methods. These include methods that focus on generating challenging instructions, such as **Forewarned is Forearmed** (Li et al., 2025b), which optimizes a proposer model to specifically generate instructions that induce failure from the target model. Another approach, **LLM2LLM** (Lee et al., 2024), operates on a similar principle of error-based feedback, iteratively using the synthesizer model to generate new data for samples where the fixer model previously made

Method	Math Reasoning Tasks				Domain Reasoning			Summary	
	GSM8K	MATH	Inf-Inst	Avg.	Olaw	Med	Avg.	Avg.	Impr. (Δ)
<i>Fixer Model: Qwen2.5-3B</i>									
Zero-shot	79.52	59.62	77.50	72.21	39.84	34.02	36.93	58.10	-
<i>Forewarned is Forearmed</i>									
Iteration 1	81.04	60.26	78.80	73.37	38.03	32.48	35.25	58.12	+0.02
Iteration 2	80.97	<u>61.32</u>	<u>79.30</u>	<u>73.86</u>	38.97	33.72	36.34	<u>58.86</u>	+0.76
<i>Montessori-Instruct</i>									
Iteration 1	80.74	60.36	78.20	73.10	38.34	32.80	35.57	58.09	-0.01
Iteration 2	80.81	60.94	78.60	73.45	39.41	34.06	36.73	58.76	+0.66
<i>LLM2LLM</i>									
Iteration 1	80.21	59.98	78.10	72.76	38.15	32.62	35.38	57.81	-0.29
Iteration 2	80.36	60.22	78.30	72.96	38.65	33.24	35.95	58.15	+0.05
<i>sRecycled</i>	80.66	60.80	78.60	73.35	39.53	<u>34.18</u>	36.86	58.75	+0.65
Ours									
Iteration 1	<u>81.12</u>	60.76	79.10	73.66	38.59	33.78	36.19	58.67	+0.57
Iteration 2	81.50 (+0.38)	62.30 (+0.98)	80.70 (+1.40)	74.83 (+0.97)	40.16 (+0.32)	34.90 (+0.72)	37.53 (+0.60)	59.91 (+1.05)	+1.81
<i>Fixer Model: Llama3.2-3B-Instruct</i>									
Zero-shot	78.08	51.70	75.60	68.46	37.03	32.14	34.59	54.91	-
<i>Forewarned is Forearmed</i>									
Iteration 1	79.22	<u>52.12</u>	76.10	69.15	36.65	31.88	34.27	55.19	+0.28
Iteration 2	<u>79.60</u>	51.84	76.30	69.25	37.59	32.90	35.25	55.65	+0.74
<i>Montessori-Instruct</i>									
Iteration 1	78.46	51.80	76.00	68.75	36.71	32.12	34.41	55.02	+0.11
Iteration 2	79.15	51.94	<u>76.60</u>	69.23	37.40	32.76	35.08	55.57	+0.66
<i>LLM2LLM</i>									
Iteration 1	78.39	51.48	75.70	68.52	36.40	31.62	34.01	54.72	-0.19
Iteration 2	78.62	51.76	76.00	68.79	37.15	32.30	34.72	55.17	+0.26
<i>sRecycled</i>	78.92	51.82	76.50	69.08	38.15 (+0.12)	33.48 (+0.22)	35.81 (+0.17)	<u>55.77</u>	+0.86
Ours									
Iteration 1	79.45	52.04	76.50	<u>69.33</u>	37.34	32.54	34.94	55.57	+0.66
Iteration 2	80.43 (+0.83)	52.68 (+0.56)	77.30 (+0.70)	70.14 (+0.81)	<u>38.03</u>	<u>33.26</u>	<u>35.64</u>	56.34 (+0.57)	+1.43

Table 1: The performance of different data synthesis methods on various datasets using different models as fixers.

errors. A more sophisticated method, **Montessori-Instruct** (Li et al., 2025c), using Influence Functions to assess the utility of synthetic data. We also include **sRecycled** (Li et al., 2024a), which introduces a selective reflection-tuning mechanism where the fixer model autonomously selects and recycles high-quality synthetic data.

Implementation Details. For our method and all baselines, we select Qwen2.5-7B-Instruct (Yang et al., 2024) as the synthesizer model, alongside Qwen2.5-3B (Yang et al., 2024) and Llama3.2-3B-Instruct (Dubey et al., 2024) as the fixer models. As the collaborator model, we utilized Qwen2.5-Math-7B-Instruct for mathematical tasks and Qwen2.5-7B-Instruct for other domains. All experiments follow the two-iteration setting established by the baseline methods (Li et al., 2025c). Detailed parameter settings are presented in Appendix B.

4.2 Main Results

Performance on Math Reasoning. To rigorously assess our framework, we evaluated it on challenging mathematical reasoning benchmarks. As evidenced in Table 1, when applying Qwen2.5-

3B as the fixer model, CoFAG demonstrates a significant performance advantage. Notably, the performance of our method after a single iteration is already comparable to, or even exceeds, the final performance of other methods. After the second iteration, CoFAG’s superiority is even more pronounced. This consistent performance margin validates the superiority of the CoFAG framework.

Cross-Architecture and Domain Generalization.

We replicated our experiments using Llama3.2-3B-Instruct as the fixer model. As shown in Table 1, the performance trends mirror those observed with the Qwen model. CoFAG delivers the highest scores across all benchmarks, improving the fixer model’s accuracy on GSM8K to 80.43% and on MATH to 52.68%. This demonstrates that the benefits of our framework are not contingent on a specific model family. And we extended our evaluation to knowledge-intensive domains beyond mathematics, including medicine and law. In Table 1, the data synthesized by CoFAG improved the fixer model’s accuracy by a notable margin over baselines. The results confirm that our framework is an architecturally agnostic and domain-versatile tool for en-

Model	Method	MATH	AIME	Avg.
Qwen2.5-3B	Base	59.62	10.00	34.81
	Forewarned is Forearmed	61.32	11.67	36.50
	CoFAG (Ours)	62.30	15.00	38.65
	Δ vs. Forewarned	+0.98	+3.33	+2.15
Qwen2.5-7B	Base	64.22	13.33	38.78
	Forewarned is Forearmed	65.38	13.33	39.36
	CoFAG (Ours)	66.16	16.67	41.42
	Δ vs. Forewarned	+0.78	+3.34	+2.06
Qwen2.5-14B	Base	67.86	15.00	41.43
	Forewarned is Forearmed	68.40	16.67	42.54
	CoFAG (Ours)	69.14	18.33	43.74
	Δ vs. Forewarned	+0.74	+1.66	+1.20
Average	Base	63.90	12.78	38.34
	Forewarned is Forearmed	65.03	13.89	39.46
	CoFAG (Ours)	65.87	16.67	41.27
	Δ vs. Forewarned	+0.83	+2.78	+1.80

Table 2: Performance comparison of different data synthesis methods across Qwen2.5 model scales on MATH and AIME.

hancing diverse LLMs.

Scalability and Computational Cost. To further evaluate the scalability and robustness of CoFAG, we conduct experiments with fixer models of different scales and extend the evaluation to more challenging mathematical benchmarks. Specifically, we use Qwen2.5-3B, 7B, and 14B as fixer models. In addition to MATH, we further evaluate on AIME by combining the 2024 and 2025 datasets as the test set. As shown in Table 2, CoFAG consistently outperforms both the base model and the strongest baseline across all model scales. On AIME, CoFAG achieves 15.00%, 16.67%, and 18.33%, bringing clear improvements over the corresponding base models. The cognitive-feedback mechanism remains effective as model scale increases and continues to provide useful guidance on more challenging reasoning tasks.

Compared with standard SFT, CoFAG introduces offline computational overhead during the training phase. This overhead mainly comes from two components: the fixer model performs an additional forward pass to compute ISI, and the synthesizer model is updated by DPO on preference data. As detailed in Appendix F, the additional training-side overhead is on the same order as one SFT training iteration. However, CoFAG improves data efficiency, substantially reducing the scale of SFT dataset required. The instruction generation process avoids the costly manual or model-assisted construction of SFT dataset.

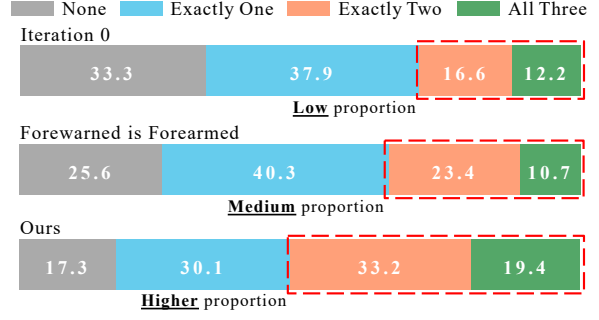


Figure 3: Distribution of instructions' characteristics before and after different methods (N=1,000).

Analysis of Instruction Quality. To quantitatively measure the impact on instruction quality of our framework and baseline, we conducted an analysis of the generated instructions (with Infinity-Instruct as the seed dataset). We sampled 1,000 instructions from the synthesizer model before any iterative refinement and after frameworks had converged. We then utilized Qwen-plus-latest to classify each instruction, identifying how many high-quality characteristics it exhibited: promoting complex reasoning, deviating from the learned distribution, and requiring multi-source knowledge. The results are summarized in Figure 3. CoFAG significantly enhanced the synthesizer model's capacity to create high-quality instructions. The proportion of instructions featuring exactly two and three characteristics soared from 28.8% to 52.6%, significantly surpassing 34.1% of Forewarned is Forearmed. CoFAG specifically enhanced the synthesizer model's ability to synthesize instructions with multiple layers of cognitive complexity. The more detailed case analysis of the instructions is presented in Appendix D and Appendix E.

4.3 Ablation Studies

Effect of Warm-up. To dissect the contribution of our model preparation step, we conducted separate ablation experiments by individually removing the warm-up process for the synthesizer model and the fixer model. As shown in Table 3, removing either warm-up component degrades performance. Removing the initial SFT for the synthesizer model results in the most significant performance drop. This is because the unaligned synthesizer model introduces semantic noise into the system. Forgoing the fixer model's warm-up step leads to a score of 61.40%. This configuration introduces feedback noise. The unwarmed fixer model lacks the baseline competency to properly assess the instructions.

Method	MATH	GSM8k
Ours	62.30	81.50
<i>Effect of Warm-up</i>		
W/o synthesizer's warm-up	60.76 (-1.54)	79.37 (-2.13)
W/o fixer's warm-up	61.40 (-0.90)	80.97 (-0.53)
<i>Effect of Cognitive Feedback</i>		
Forewarned is Forearmed	61.32 (-0.98)	81.04 (-0.46)
Montessori-Instruct	60.94 (-1.36)	80.81 (-0.69)
IFD	61.24 (-1.06)	80.89 (-0.61)
<i>Effect of Synthesizer Optimization</i>		
Original synthesizer	59.88 (-2.42)	79.68 (-1.82)
LLM2LLM	60.22 (-2.08)	80.36 (-1.14)
Filtering	61.08 (-1.22)	80.66 (-0.84)

Table 3: Ablation study of different components on MATH and GSM8K datasets with Qwen2.5-3B as the fixer model.

Effect of Cognitive Feedback. To evaluate different methods for collecting the fixer model’s feedback, we compared our cognitive feedback with three baselines: (1) Forewarned is Forearmed, which takes instructions that the fixer model failed to perform correctly as positive samples; (2) Montessori-Instruct, which selects instructions based on the reduction of the fixer model’s loss on the seed test set; and (3) IFD (Li et al., 2024b), which utilizes the Instruction Following Difficulty metric to score and select instructions. The results in Table 3 clearly show that our method achieves a superior accuracy of 62.30%, consistently outperforming Forewarned is Forearmed (61.32%), Montessori-Instruct (60.94%), and IFD (61.24%). This outcome highlights a fundamental conclusion: our process-aware cognitive feedback is a far more effective signal for instruction quality than result-oriented correctness, training benefit, or sequence-level difficulty metrics.

Effect of Synthesizer Optimization. To analyze the effectiveness of the synthesizer model’s optimization, we compare our method with the following baselines: (1) Original synthesizer model: This method solely uses the warm-up synthesizer model to generate training data in one go, with the same scale as that produced by other methods. (2) LLM2LLM: This method generates training data by rewriting instructions that the fixer model failed to handle correctly. (3) Filtering: This method uses the warm-up synthesizer model to generate a large number of instructions in one go, then selects the top 50% of the instructions with the highest ISI from this pool as training data. As illustrated in Table 3, our method consistently shows superior

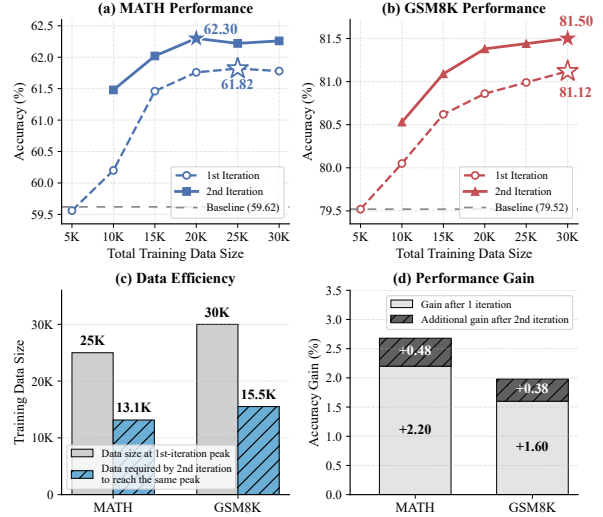


Figure 4: Ablation study of global iterative process on MATH and GSM8K datasets.

performance and data efficiency across all baselines. It achieves the highest accuracy of 62.30%, while other methods lag significantly. The conclusion is that the optimization process itself is critical. Merely filtering for high ISI instructions or rewriting failed ones is insufficient.

Effect of Global Iterative Process. We analyze how the number of iterations impacts the fixer model’s performance. Figure 4 plots the fixer model’s accuracy on the MATH and GSM8K dataset against the total training data size, comparing the results of a single iteration versus two iterations. The results show a clear benefit to the iterative process. The second iteration demonstrates significantly greater data efficiency. The fixer model achieves an accuracy of approximately 62.0% with only 15K training samples at the second iteration in MATH. Additionally, the iterative process successfully raises the fixer model’s absolute performance ceiling. The fixer model’s peak accuracy improves from 61.82% in the first iteration to 62.3% in the second. The performance gain across iterations provides strong evidence that a dynamic and iterative process yields a more efficient learning trajectory.

5 Conclusion

This paper introduces a novel data synthesis framework, CoFAG, designed to generate higher-quality training data by measuring fixer models’ internal processing effort. Central to this method is a self-evolving loop where a synthesizer model is optimized to generate instructions that induce high Cognitive Load, a process-aware quality signal we

quantify using ISI. Empirical results demonstrate that, CoFAG produces a curriculum with verifiably higher quality. The performance in all aspects validates the superiority of using internal cognitive feedback for high-quality instruction generation.

Limitations

Although CoFAG achieves consistent improvements, several limitations remain. First, its three-model collaborative framework introduces additional offline cost for response generation, ISI computation, and DPO-based synthesizer refinement, while the final fixer model retains the same inference cost. Second, ISI relies on a reasonably aligned fixer model; when seed data are limited, insufficient warm-up may introduce noisy cognitive feedback. Third, while CoFAG is evaluated across different architectures, its behavior on frontier-scale models and the calibration of ISI across model scales remain underexplored. Finally, CoFAG mainly optimizes instruction generation, while the quality of reference responses still depends on the collaborator model. Noisy or incomplete responses may therefore limit the quality of the final SFT data.

Acknowledgments

This work was supported in part by the National Key R&D Program of China under Grant 2023YFF0905503, the National Natural Science Foundation of China under Grant No. 62472188, the Education University of Hong Kong Start-up Funding under Grant No. R4400, and the HK UGC Teaching Development and Language Enhancement Grant IICA Project under Grant No. T5024 and No. T5016.

References

- Hsin-Yu Chang, Pei-Yu Chen, Tun-Hsiang Chou, Chang-Sheng Kao, Hsuan-Yun Yu, Yen-Ting Lin, and Yun-Nung Chen. 2024. [A survey of data synthesis approaches](#). *CoRR*, abs/2407.03672.
- Junying Chen, Zhenyang Cai, Ke Ji, Xidong Wang, Wanlong Liu, Rongsheng Wang, Jianye Hou, and Benyou Wang. 2024. [Huatuogpt-o1, towards medical complex reasoning with llms](#). *Preprint*, arXiv:2412.18925.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. [Training verifiers to solve math word problems](#). *CoRR*, abs/2110.14168.
- Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 82 others. 2024. [The llama 3 herd of models](#). *CoRR*, abs/2407.21783.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. [Measuring mathematical problem solving with the MATH dataset](#). In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.
- HKAIR Lab. 2024. [Hk-o1aw models: Leveraging o1 slow thinking in the development of hong kong legal large language models](#). <https://github.com/HKAIR-Lab/HK-O1aw>.
- Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael W. Mahoney, Kurt Keutzer, and Amir Gholami. 2024. [LLM2LLM: boosting llms with novel iterative data enhancement](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 6498–6526. Association for Computational Linguistics.
- Jijie Li, Li Du, Hanyu Zhao, Bo wen Zhang, Liangdong Wang, Boyan Gao, Guang Liu, and Yonghua Lin. 2025a. [Infinity instruct: Scaling instruction selection and synthesis to enhance language models](#). *Preprint*, arXiv:2506.11116.
- Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024a. [Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16189–16211, Bangkok, Thailand. Association for Computational Linguistics.
- Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. [From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7602–7635, Mexico City, Mexico. Association for Computational Linguistics.
- Qintong Li, Jiahui Gao, Sheng Wang, Renjie Pi, Xueliang Zhao, Chuan Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025b. [Forewarned is forearmed: Harnessing llms for data synthesis via failure-induced exploration](#). In *The Thirteenth International*

- Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025. OpenReview.net.
- Xiaochuan Li, Zichun Yu, and Chenyan Xiong. 2025c. [Montessori-instruct: Generate influential training data tailored for student learning](#). In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.
- Lin Long, Rui Wang, Ruixuan Xiao, Junbo Zhao, Xiao Ding, Gang Chen, and Haobo Wang. 2024. [On llms-driven synthetic data generation, curation, and evaluation: A survey](#). In *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 11065–11082. Association for Computational Linguistics.
- OpenAI. 2023. [GPT-4 technical report](#). *CoRR*, abs/2303.08774.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.
- Timo Schick and Hinrich Schütze. 2021. [Generating datasets with pretrained language models](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6943–6951. Association for Computational Linguistics.
- John Sweller. 1988. [Cognitive load during problem solving: Effects on learning](#). *Cogn. Sci.*, 12(2):257–285.
- John Sweller, Jeroen J. G. van Merriënboer, and Fred G. W. C. Paas. 1998. [Cognitive Architecture and Instructional Design](#). *Educational Psychology Review*, 10(3):251–296.
- Zhen Tan, Dawei Li, Song Wang, Alimohammad Beigi, Bohan Jiang, Amrita Bhattacharjee, Mansoor Karami, Jundong Li, Lu Cheng, and Huan Liu. 2024. [Large language models for data annotation and synthesis: A survey](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 930–957. Association for Computational Linguistics.
- Ke Wang, Jiahui Zhu, Minjie Ren, Zeming Liu, Shiwei Li, Zongye Zhang, Chenkai Zhang, Xiaoyu Wu, Qiqi Zhan, Qingjie Liu, and Yunhong Wang. 2024. [A survey on data synthesis and augmentation for large language models](#). *CoRR*, abs/2410.12896.
- Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xiong-Hui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. 2025. [Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning](#). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.
- Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. [Self-instruct: Aligning language models with self-generated instructions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.
- Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhao Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. [Wizardlm: Empowering large pre-trained language models to follow complex instructions](#). In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. [Qwen3 technical report](#). *CoRR*, abs/2505.09388.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. [Qwen2.5 technical report](#). *CoRR*, abs/2412.15115.
- Yixin Ye, Zhen Huang, Yang Xiao, Ethan Chern, Shijie Xia, and Pengfei Liu. 2025. [LIMO: less is more for reasoning](#). *CoRR*, abs/2502.03387.
- Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. [Self-rewarding language models](#). In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.
- Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. [LIMA: less is more for alignment](#). In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Algorithm 1 CoFAG for synthetic data construction

Require: Initial seed instructions D_{seed} , synthesizer model S with parameters θ , fixer model F with parameters ϕ , Number of iterations R .

- 1: Warm up S by maximizing $\mathbb{E}_{i \sim D_{seed}}[\log P(i|p; \theta)]$, where p is a fixed prompt. Let model be $S^{(0)}$. {Step 1: Model Preparation}
 - 2: Generate warm-up set $D_{warmup} \sim S^{(0)}(p)$; fine-tune F on D_{warmup} to get $F^{(0)}$.
 - 3: $D_{final} \leftarrow \emptyset$
 - 4: **for** $round = 1$ to R **do**
 - 5: Generate candidate instructions $D_{test}^{(round)} \sim S^{(round-1)}(p)$.
 - 6: For each $i_j \in D_{test}^{(round)}$, calculate cognitive load $ISI(i_j; \phi^{(round-1)})$. {Step 2: Cognitive Feedback}
 - 7: Form preference dataset $D_{pref}^{(round)} = \{(i_w, i_l)\}$ by pairing high-load instructions (i_w) with low-load ones (i_l).
 - 8: Minimize $\mathcal{L}_{DPO}$ using pairs from $D_{pref}^{(round)}$ to produce the optimized synthesizer $S^{(round)}$. {Step 3: Strategy Refinement}
 - 9: Generate a new dataset $D_{high_quality}^{(round)} \sim S^{(round)}(p)$.
 - 10: $D_{final} \leftarrow D_{final} \cup D_{high_quality}^{(round)}$.
 - 11: Fine-tune initial model F_0 with D_{final} to produce the enhanced fixer model $F^{(round)}$.
 - 12: **end for**
 - 13: **return** Final fixer model $F^{(R)}$ and dataset D_{final} .
-

A Algorithm Pseudocode

This section presents the pseudocode for the Cognitive-Feedback Adaptive Generation (CoFAG) framework. As shown in Algorithm 1, the algorithm formalizes the iterative synthesizer-fixer loop described in the main paper, outlining the key steps from the initial warm-up phase to the dynamic process of cognitive feedback and strategy refinement. This provides a high-level guide for implementing the CoFAG workflow.

B Detailed Experimental Setup

This section outlines the specific configurations used to conduct our experiments, ensuring they can be precisely replicated. We detail the dataset sizes for each stage, the software and hardware environ-

ments, and the complete set of hyperparameters for all model training and optimization phases.

B.1 Dataset and Iteration Sizes

The volume of data generated at each step was calibrated based on the domain.

For mathematical reasoning datasets (GSM8K, MATH, Infinity-Instruct):

- **Synthesizer Warm-up:** A maximum of 10,000 instructions from the original training set were used.
- **Fixer Warm-up:** The warm-up synthesizer generated 5,000 instructions.
- **Cognitive Feedback Loop (per iteration):**
 - The synthesizer generated a test dataset of 10,000 instructions.
 - We selected the top 50% (5,000) as positive samples and the bottom 50% (5,000) as negative samples to create 5,000 preference pairs for DPO.
- **Fixer Optimization (per iteration):** The refined synthesizer generated 10,000 new instructions for training the fixer model.

For legal (Olaw) and medical (medical-o1-reasoning-SFT) datasets:

- The process was identical to the math setup, with one exception.
- **Fixer Optimization (per iteration):** The refined synthesizer generated 15,000 instructions. This larger size was chosen to help the fixer model achieve better convergence in these knowledge-intensive domains, especially since our framework focuses on optimizing instruction generation, not the ground-truth responses.

B.2 Implementation Details

- **Models:** As stated in the paper, the primary synthesizer was Qwen2.5-7B-Instruct, and fixers included Qwen2.5-3B and Llama3.2-3B-Instruct. Ground-truth responses were generated by Qwen2.5-Math-7B-Instruct (for math) and Qwen2.5-7B-Instruct (for other domains). The automated judge was Qwen-plus-latest.

Stage	Fine-tuning Type	LoRA Rank	Batch Size	Gradient Steps	LR Scheduler	Learning Rate	DPO Parameters		
							β	FTX	Loss Function
Synthesizer Warmup	LoRA	16	1	2	Cosine	2.0×10^{-6}	—	—	—
Fixer Warmup	Full+ZeRO-3	—	1	2	Cosine	1.0×10^{-5}	—	—	—
Synthesizer Optimization	DPO+LoRA	16	1	1	Cosine	5.0×10^{-7}	0.1	0.2	Sigmoid
Fixer Optimization	Full+ZeRO-3	—	1	2	Cosine	1.0×10^{-6}	—	—	—

Table 4: Hyperparameter Settings for Model Training and Optimization

- **Hardware:** All experiments were conducted on a server equipped with **4 NVIDIA GeForce RTX 3090 GPUs** (24GB VRAM each).
- **Software:** Model fine-tuning and optimization were managed using **LLaMA-Factory**. Data generation was accelerated with **VLLM**. Key libraries included PyTorch, Transformers, and DeepSpeed for ZeRO-3 optimization.
- **Generation Parameters:** For tasks using VLLM, the temperature for creative instruction generation was set to **1.2** with top-p sampling at **0.9**. To ensure deterministic and factual ground-truth responses, the temperature was set to **0**.

B.3 Hyperparameter Settings

All supervised fine-tuning (SFT) and Direct Preference Optimization (DPO) stages were performed using LLaMA-Factory. The key hyperparameters are detailed in Table 4. Parameters not specified here were left at their default values within the LLaMA-Factory framework.

C Prompting Template

This section provides the exact templates used for interacting with the large language models at various stages of our framework. Precise and consistent prompting is critical for reproducibility, as minor changes in wording can lead to significant differences in model behavior. We present the prompts for instruction generation, standard response generation, and automated answer evaluation.

C.1 The Prompt of Instruction Generation

In math domain:

Prompt
Generate a mathematical problem that is clear, solvable, challenging, and suitable for students to practice in class.

In legal domain (the English version is shown for clarity; the Chinese version was used in experiments to match the OLaw dataset.):

Prompt
Generate 1 legal question. The problem needs to meet the following requirements: 1. Complexity requirements: - The answer to the question needs to be concluded through in-depth understanding and analysis of reference legal materials - It should be an open question, requiring reasoning and explanation - Can't be simple questions or factual questions 2. Problem form: - Can be a case analysis question - Can be a legal application - It can be a legal concept explanation question 3. Problem structure: - Clear statement of problem - No answer required 4. Difficulty level: - Medium difficulty, suitable for law students or practitioners to think and discuss Please output the question directly.

In medical domain:

Prompt
Generate a medical problem.

C.2 The Prompt of Standard Response Generation

In math domain:

Prompt
Please reason step by step, and put your final answer within <code>\boxed{ }</code> .

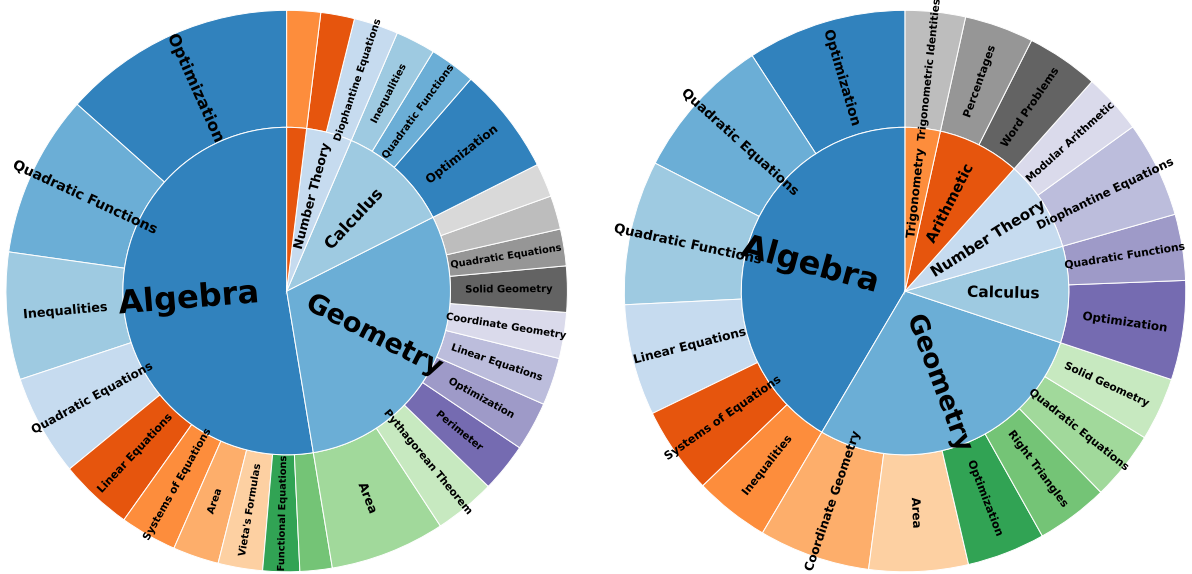


Figure 5: Distribution of instruction categories before and after synthesizer optimization. Inner rings represent major categories; outer rings represent subcategories.

<instruction>

In legal domain (The English version of the prompt is displayed, but in order to keep the language setting consistent with the original dataset, the Chinese version of the prompt is used in the experiment) :

Prompt

You are an expert legal advisor, able to reason and answer legal questions step by step. You will be given a legal question. Your task is to answer it by reasoning step by step. Requirements:

1. Use multiple reasoning steps
2. Explore multiple approaches to reach an answer
3. Verify the given answer through different methods
4. Consider potential alternative answers and explain why they were rejected
5. Thoroughly test all other possibilities. You may be wrong
6. After reasoning, produce a clear conclusion

<instruction>

In medical domain:

Prompt

You are an expert medical consultant who can reason and answer medical questions step by step. You will be given a medical question. Your task is to answer the question by reasoning step by step. Requirements:

1. Use multiple reasoning steps
2. Explore multiple methods to reach the answer
3. Verify the given answer through different methods
4. Consider potential alternative answers and explain why they were rejected
5. Fully test all other possibilities. You may be wrong
6. After reasoning, output a clear conclusion

<instruction>

C.3 The Prompt of Answer Judge

All areas use the same prompt template:

Prompt

The following is the student's answer (predict) and standard answer (label) on a math/legal/medical problem. You need to determine whether the student's answer is correct. You just need to answer 'YES' or

[Instruction] Calculate 111111×22222222 [Response] To calculate the product of 111111 and 22222222 1. Write down the numbers one below the other: 2. Multiply each digit of the first number by each digit of the second number: 3. Move one position to the left and repeat the process. 4. Continue this process for all digits in both numbers. 5. Add up all the results. 6. find that the product of 111111 and 22222222 is [Complex reasoning paths] ✗ [Deviation from training] ✗ [Multisource knowledge] ✗ [ISI] <u>1.01</u>	[Instruction] How many times does the number '9' appear in all the integers from 1 to 911? [Response] To determine how many times 1. Units Place: The digit '9' appears in 9, 19, 29, ..., 899, 909. 2. Tens Place: The digit '9' appears in 90-99, 190-199, ..., 890-899. 3. Hundreds Place: The digit '9' appears in 900-911. 4. Therefore, the final count is the sum of all occurrences. [Complex reasoning paths] ✓ (Decomposition by place value) [Deviation from training] ✗ [Multisource knowledge] ✗ [ISI] <u>1.36</u>	[Instruction] There are three colors of balls in a box: red, blue, and green. Except for two that are not red balls, except for two that are not blue balls, except for two that are not green balls. How many balls are there in the box? [Response] This problem requires translating natural language constraints into a formal system. 1. Blue + Green = 2 2. Red + Green = 2 3. Red + Blue = 2 4. Solving this system of linear equations reveals that Red = 1, Blue = 1, and Green = 1. [Complex reasoning paths] ✗ [Deviation from training] ✗ [Multisource knowledge] ✓ (Natural language & Algebraic modeling) [ISI] <u>1.48</u>	[Instruction] A spider is at point A, a corner of the bottom surface of a rectangular room that is 30 meters long, 12 meters wide, and 14 meters high. It wants to go to point B, the opposite vertex (on the top surface). What is the shortest distance the spider can crawl on the wall and the ground? [Response] To find the shortest surface distance, we must unfold the 3D room into a 2D plane. There are multiple ways to unfold it. 1. Combine width and height..... 2. Combine length and height..... 3. Combine length and width..... 4. So the shortest path is [Complex reasoning paths] ✓ (Multi-step conceptual transformation) [Deviation from training] ✓ (Non-standard geometric application) [Multisource knowledge] ✓ (3D reasoning & 2D path optimization) [ISI] <u>2.54</u>
--	--	---	---

Figure 6: Case studies illustrating the correlation between task complexity and the Inference Stress Index (ISI). We present four distinct instruction types ranging from rote calculation to complex spatial reasoning. The ISI score accurately reflects the cognitive load imposed on the model, with higher scores indicating tasks that require multi-step conceptual transformations and deviation from standard training distributions.

'NO'. Do not have any additional output:
predict: <Tested Answer>
label: <Real Answer>

D Case Study.

We present qualitative examples to illustrate how CoFAG adapts instruction difficulty across training iterations. For detailed examples of ISI calculation, see Appendix E. To intuitively understand how CoFAG reshapes the data distribution to target the fixer model's weaknesses, we use Qwen-plus-latest to classify instructions and removed classes that appeared less than 1% of the time. We visualize the topic composition of the synthesized instructions from the MATH dataset across two iterations in Figure 5. The inner ring represents broad mathematical disciplines and the outer ring details fine-grained subtopics. In Iteration 1, the generated data is heavily dominated by Algebra, accounting for half of the distribution. This aligns with the fixer model's initial state, where standard algebraic manipulation yields higher cognitive load. However, a shift occurs in Iteration 2. The frequency of occurrence of Algebra began to decrease, while the frequency of occurrence of Geometry gradually increased. Moreover, the outer ring in Iteration 2 exhibits a richer spectrum of subtopics, indicating that CoFAG not only increases quality but also significantly keeps the diversity of the dataset. This evolution confirms that our method effectively acts

as an adaptive instruction scheduler, dynamically steering generation away from the model's comfort zone toward its knowledge frontiers.

E ISI Calculation Examples

To provide a concrete understanding of how the Inference Stress Index (ISI) quantifies cognitive load, we present a qualitative analysis of four distinct instructions and their corresponding ISI scores in Figure 6. These cases illustrate the correlation between the complexity of the required reasoning path and the resulting ISI metric.

Case 1: Rote Calculation (ISI = 1.01). The first case involves a large number multiplication task (111111×22222222). While computationally intensive for a human, for an LLM, this task follows a strict algorithmic pattern. The reasoning path is linear and relies heavily on learned arithmetic distributions. Consequently, the model exhibits low uncertainty across tokens, resulting in a baseline ISI of 1.01.

Case 2: Decomposition and Counting (ISI = 1.36). The second case requires counting the occurrence of the digit '9' in a range of integers. This forces the model to decompose the problem by place value (units, tens, hundreds). The cognitive load increases as the model must maintain a running count and apply logic rather than simple pattern completion. The ISI rises to 1.36, reflecting this moderate increase in reasoning demand.

Operation	Scale	Estimated forward passes
ISI scoring	10,000 data points	10,000
DPO optimization	5,000 pairs	40,000
CoFAG extra cost	–	50,000
SFT baseline	10,000 samples	30,000

Table 5: Estimated forward-pass cost of the additional training-side operations in one CoFAG iteration compared with one standard SFT run on the same data scale.

Case 3: Logical Modeling (ISI = 1.48). The third case presents a logic puzzle involving colored balls ("Except for two that are not red..."). To solve this, the model must translate natural language constraints into a system of linear equations (Algebraic modeling). This translation process induces higher cognitive bifurcation (entropy) as the model bridges the gap between linguistic ambiguity and formal logic, yielding an ISI of 1.48.

Case 4: Spatial and Geometric Reasoning (ISI = 2.54). The final case is the most complex, involving a "spider in a room" shortest path problem. This requires a multi-step conceptual transformation: the model must unfold a 3D rectangular prism into a 2D plane to calculate the Euclidean distance. This task deviates significantly from standard textual training data and requires integrating spatial reasoning with geometric optimization. The fixer model exhibits high uncertainty at key reasoning junctions (such as deciding how to unfold the room), resulting in a high ISI of 2.54.

These examples validate that ISI is not merely measuring sequence length or vocabulary rarity, but is effectively capturing the *process-aware* difficulty of the underlying reasoning steps.

F Computational Cost Analysis

We provide a first-order calculation of the additional computational cost introduced by one CoFAG iteration in Table 5. This calculation focuses on the training-side overhead required by CoFAG beyond the standard SFT stage. Specifically, we count the cost of ISI-based feedback computation and DPO-based synthesizer optimization. The cost of sample generation is not included in this comparison, since generation serves as an offline data acquisition process and substantially reduces the human labor required to construct the training dataset.

In one CoFAG iteration, ISI is computed over a pool of 10,000 data points. These data points are ranked by ISI, and the top 5,000 and bottom 5,000

samples are paired to form 5,000 positive-negative preference pairs for DPO. The corresponding SFT baseline trains the fixer model on the full set of 10,000 data points.

We use forward passes as the accounting unit. For SFT, each sample requires one forward pass and one backward pass. Following the common approximation that a backward pass costs about twice a forward pass, the cost of SFT on 10,000 samples is:

$$\begin{aligned} C_{\text{SFT}} &= 10,000 \times (1 + 2) \\ &= 30,000. \end{aligned} \quad (9)$$

For ISI computation, the fixer model scores each data point with teacher forcing. Since the full response sequence can be processed in parallel, we count one forward pass per data point:

$$\begin{aligned} C_{\text{ISI}} &= 10,000 \times 1 \\ &= 10,000. \end{aligned} \quad (10)$$

For DPO, each preference pair contains one positive sample and one negative sample. For each sample, the trainable policy requires one forward pass and one backward pass, while the frozen reference model requires one additional forward pass. Thus, the cost of one DPO pair is:

$$\begin{aligned} C_{\text{DPO-pair}} &= 2 \times ((1 + 2) + 1) \\ &= 8. \end{aligned} \quad (11)$$

Since CoFAG uses 5,000 preference pairs, the DPO cost is:

$$\begin{aligned} C_{\text{DPO}} &= 5,000 \times 8 \\ &= 40,000. \end{aligned} \quad (12)$$

Therefore, the additional training-side overhead of one CoFAG iteration is:

$$\begin{aligned} C_{\text{CoFAG-extra}} &= C_{\text{ISI}} + C_{\text{DPO}} \\ &= 10,000 + 40,000 \\ &= 50,000. \end{aligned} \quad (13)$$

The relative overhead compared with one SFT run is:

$$\frac{C_{\text{CoFAG-extra}}}{C_{\text{SFT}}} = \frac{50,000}{30,000} \quad (14)$$

$$\approx 1.67.$$

This calculation shows that the additional training-side overhead of one CoFAG iteration is on the same order as one standard SFT run on the same data scale. The main overhead comes from DPO-based synthesizer optimization, while ISI scoring adds only one forward pass per data point. In addition, all these costs are incurred offline during data construction and synthesizer refinement. Once the fixer model is trained, CoFAG introduces no additional inference-time cost.