Efficiently Selecting Response Generation Strategy by Self-Aligned Perplexity for Fine-Tuning LLMs

Anonymous ACL submission

Abstract

Fine-tuning large language models (LLMs) typically relies on producing large sets of input-output pairs. Yet for a given question, there can be many valid outputs. In practice, these outputs are often derived by distilling knowledge from teacher models, and they can vary depending on the specific teacher model or prompting strategy employed. Recent findings show that *how* these training outputs are generated can significantly affect the performance of the fine-tuned model, raising an important question: how do we pick the best *data generation method* from among numerous possibilities? Rather than exhaustively training and evaluating on each candidate, this paper proposes a scalable approximate method that assesses a *small* subset of generated data to estimate its suitability for a specific target LLM. Our central idea is that effective outputs should be *familiar* to the target LLM. While previous work measures familiarity with perplexity, we find that perplexity might be suboptimal in characterizing "familiarity" through empirical analyses and practical observations. To address this, we introduce *self-aligned perplexity*, a novel metric capturing how closely candidate outputs adhere to the target LLM's own style and reasoning patterns. In this way, we can identify the most effective generation strategy on a small sample, then apply it to produce the complete training set. We demonstrate that training on data generated by the chosen method yields significant improvements across diverse reasoning-focused benchmarks, particularly in cases where different candidate methods lead to highly divergent training outcomes.

1 Introduction

When instruction-tuning an LLM, training data consists of question-response pairs, where multiple valid responses can be generated for the same input. Previous studies (Anonymous, 2024) show that datasets with identical input questions but different responses can lead to varied learning outcomes, even when responses contain similar levels of detail. This raises a key question: *how can we construct responses that are most effective for the target LLM?*

Prior research has explored improving responses by adding details or rationales, such as structuring ground truth step by step (Hsieh et al., 2023; Ranaldi and Freitas, 2024), incorporating rationales, or enriching responses with additional information (Zhang et al., 2024; Kang et al., 2023; Li et al., 2022). However, recent studies (Anonymous, 2024; Yang et al., 2024) suggest that more details or converting responses to step by step style do not always improve performance and that alignment with the LLM's linguistic style is crucial.

In our experiment, we observe that no single response generation strategy works universally across tasks. Thus, we need to create a method to find out the most effective way to generate responses for each task, rather than a single method for all tasks.

Some works (Xu et al., 2024; Kim et al., 2024) attempt to predict the effectiveness of response generation methods by evaluating the entire training dataset. They generate full training datasets using each method and then estimate training effectiveness based on scores computed via algorithms or reward models. However, these approaches are computationally expensive and not scalable.

However, can we predict the effectiveness of each data generation methods efficiently? We observe an interesting phenomenon that each response generation method produces responses with a consistent style, meaning that a small subset of generated examples can effectively represent the entire dataset. Based on this assumption, we propose an efficient ranking pipeline that evaluates a limited number of samples (e.g., 50) to assess the performance of each response generation strategy. This approach uses an alignment estimation function to assign scores to each strategy, enabling us to identify the best-performing method without the need for a full-dataset evaluation.

Previous research (Anonymous, 2024) used perplexity to measure a model's familiarity with candidate question-answer pairs, proposing that lower-perplexity responses for the same input tend to yield better training performance. However, we found several cases where perplexity-based filtering was ineffective. For instance, responses structured in a step-by-step or redundant language style often have low perplexity but do not necessarily improve training outcomes. When examining the initial response from the target LLM, we note that on some tasks, the probability of the model producing a step-by-step or redundant response in its initial prediction is very low, even though these responses have low perplexity. These findings suggest that perplexity can be "hacked" by response style. Thus, traditional perplexity alone is insufficient for selecting the best response generation strategy.

To address this, we propose self-aligned perplexity, a refined metric for measuring a model's familiarity with target responses. The key idea is that a model is most familiar with the data it generates itself. Leveraging this, we modify perplexity computation by incorporating model-generated responses as in-context examples. Specifically, we first have the model produce initial responses, which is then appended to the question as an in-context example. A prompt enforce the model to pay attention to these examples when computing perplexity, thereby altering the probability estimation of the candidate response. If the target response deviates significantly from the model's own generated response—the one it is most familiar with—the model assigns it a lower probability, increasing its perplexity. Our experiments show that self-aligned perplexity outperforms traditional perplexity in selecting effective data generation strategies.

In our experiments, we observe a strong correlation between the proposed indicator and the ranking of training dataset performance. Furthermore, we construct a pool of answer generation strategies and demonstrate that applying our selection criterion leads to significant performance gains compared to the baselines—especially in scenarios where different data-generation methods produce highly divergent outcomes

2 Related Works

There has been extensive research into what types of data yield the best training outcomes for large language models (LLMs). Previous studies have identified several factors that positively influence model training, such as adding complexity (Xu et al., 2023), adding details (Zhang et al., 2024; Kang et al., 2023; Li et al., 2022), adding diversity (Luo et al., 2023), augmenting ground-truth answers in a step-by-step manner (Hsieh et al., 2023; Ho et al., 2022; Magister et al., 2023; Fu et al., 2023; Ranaldi and Freitas, 2024), and ensuring correctness (Trinh et al., 2024; Ranaldi and Freitas, 2024). However, in practice, these metrics are challenging to measure for a given dataset, making it difficult to determine the quality of training data based on these criteria. Anonymous (2024) found that familiarity, measured by perplexity, significantly impacts model training.

Perplexity has been widely used for different purpose in prior research. Perplexity has been used to select prompts (Gonen et al., 2022), showing that prompts with lower perplexity generally lead to better performance in question-answering tasks. It has also been used for selecting pretraining datasets (De la Rosa et al., 2022), detecting AI-generated content (Xu and Sheng, 2024; Hu et al., 2020), and selecting instruction-tuning data from a database (Mekala et al., 2024). Li et al. (2024) modify the perplexity score and propose "IDF" (Instruction Following Difficulty), which is used to select a small pool of challenging data from the original dataset for efficient training. Researchers hypothesize that higher perplexity indicates more challenging data, which can be beneficial for teaching LLMs new knowledge. In addition, perplexity or confidence-based curricula have been explored for NMT (Kocmi and Bojar, 2017) and general text generation (Platanios et al., 2019), where harder (high-perplexity) data are introduced progressively to improve sample efficiency. Unlike these studies, our focus is on identifying the best strategy to generate target responses (y) for a given input (x), rather than selecting difficult (x, y) pairs for training language models.

Recent efforts have begun to ask which teacher model produces the most useful synthetic targets. Xu et al. (2024) introduce a *Compatibility-Adjusted Reward* (CAR) and judge its quality by the Spearman correlation between CAR scores and downstream accuracy on two instruction-following datasets, each evaluated with a single meta-prompt. Kim et al. (2024) study nine datasets spanning mathematics, coding, and general instructions; they correlate several corpus statistics with training gains and combine them with principal-component analysis to rank teacher models. Our study differs in four key respects. First, we estimate a strategy's quality from only a small sample of its outputs, making synthetic data generation and evaluation far more affordable. Second, we target accuracy improvement, not just rank correlation. Third, we experiment on a much broader benchmark: 17 diverse tasks plus six Plan-Bench planning tasks. Fourth, we evaluate our method on datasets generated using diverse meta-prompts, explicitly accounting for prompt variability.

3 Method

This paper aims to efficiently select the most effective answer generation strategy for fine-tuning a target LLM. In what follows, we first present the problem setup, then detail our proposed *self-aligned perplexity* metric for scoring the outputs from each candidate strategy.

3.1 Problem Definition

Let $\mathcal{S} = \{S_1, \ldots, S_n\}$ be a set of candidate answer-generation strategies, where each strategy S_k produces a response $\hat{y}^k = S_k(x)$ for an input x. Our goal is to select the strategy S_ι that yields the most effective training data $\mathcal{D} = \{(x, \hat{y}^k)\}$ to fine-tune a target model M. Since generating the full dataset via the API for every strategy is costly, we evaluate a small subset $\mathcal{D}_s$ of size K ($K \ll |\mathcal{D}|$) to estimate how well each strategy's outputs align with M.

3.2 The Familiarity Hypothesis

The work in (Anonymous, 2024) suggests that if the model is more "familiar" with a given response, then the model can learn better with the given response. In their work, perplexity, which is correlated to the likelihood of generating a response with the model, is used to measure this familiarity score. In our study, we argue that perplexity is sub-optimal to measure familiarity. We suggest that familiarity can be more precisely measured by this equation:

$$F(\hat{y}) = \mathbb{E}_y \left[s(y, \hat{y})\right] = \int s(y, \hat{y}) P_M(y) dy, \quad (1)$$

where $s(y, \hat{y})$ is a semantic similarity measure between $\hat{y}$ and a sample response y drawn from the model M. In plain language, it quantifies how similar a candidate response is to the range of answers that the model might generate. It is straightforward to demonstrate that when $s(y, \hat{y}) = \delta(y, \hat{y})$, i.e., when $\delta(y, \hat{y}) = 1$ only if y is exactly identical to $\hat{y}$, the function F becomes equivalent to the likelihood $P_M(\hat{y})$, and hence equivalent to perplexity. using perplexity as a surrogate to measure familiarity fails to account for the variety of responses that may be semantically equivalent to a candidate response, thereby underestimating the familiarity. In practice, this results in assigning an excessively high perplexity to a good candidate response that the model might actually be familiar with, as evidenced by our empirical study in section 6.1.

3.3 Self-Aligned Perplexity

To address the issue above, we propose a new surrogate for measuring familiarity based on in-context learning. Instead of relying solely on raw likelihood, we treat the model's own prediction $y = M(x)$ as an in-context style example, and assess the likelihood of a candidate response conditioned on this example. This encourages the model to favor candidates that stylistically align with its own reasoning patterns.

Traditional Perplexity. For a candidate response $\hat{y}$ consisting of $|\hat{y}|$ tokens, perplexity is defined as:

$$\varphi_{\mathsf{PPL}}(M, x, \hat{y}) = \exp\left\{-\frac{1}{|\hat{y}|}\sum_{t=1}^{|\hat{y}|} \log P_M\left(\hat{y}_t \,\middle|\, x, \hat{y}_{<t}\right)\right\}. \quad (2)$$

Although lower perplexity suggests familiarity, it can misjudge responses that differ stylistically from what M typically produces (see When perplexity fails from the section 6.1).

Self-Aligned Perplexity. To address this, we use M's initial prediction, $y = M(x)$, as an in-context style example. For each input x_i in a subset $\mathcal{D}_s$:

1. Generate $y_i = M(x_i)$. $M(x_i)$ may generate incorrect responses, which can sometimes cause $\mathrm{Prompt}(y_i)$ to mislead the in-context perplexity calculation. To mitigate this, we employ a filtering mechanism to remove incorrect $M(x_i)$. After removing the incorrect samples using the evaluation metrics we used during testing, we get a group of correct y and place it into the collection S.

2. For candidate $\hat{y}_i$, include y_s (ensure s not equal to i) from the collection S in the prompt

$\mathcal{P}$ to provide style guidance. Please notice that we make sure the correct prediction y_s do not answer the question x_i, because we wish to evaluate response candidates, we only wish to provide a stylish guide rather than providing the answer.

3. Compute the perplexity of $\hat{y}_i$ conditioned on this prompt.

Formally,

$$\varphi_{\text{SPPL}}(M, x_i, \hat{y}_i) = \varphi_{\text{PPL}}(M, \mathcal{P}(y_s), \hat{y}_i) \quad \text{for } s \neq i\,. \tag{3}$$

This self-aligned measure penalizes responses that deviate from M's own style.

Selection Criterion. For each generation strategy S_k, we evaluate its effectiveness on a small subset $\mathcal{D}_s$ by computing the average self-aligned perplexity:

$$\pi_{\mathsf{SPPL}}(S_k) = \frac{1}{K} \sum_{i=1}^{K} \varphi_{\mathsf{SPPL}}(M, x_i, \hat{y}_i^k), \tag{4}$$

and select the optimal strategy via:

$$S_\iota = \arg\min_k \pi_{\mathsf{SPPL}}(S_k). \tag{5}$$

4 Benchmark Construction

In this section, we show how we use different strategies (distinct prompts and teacher LLMs) in generating high-quality responses with different styles.

4.1 Target LLMs and APIs

We use Mistral-7B-instruct-V2 (Jiang et al., 2023), Llama3-instruct (Dubey and Abhinav Jauhri, 2024) and Qwen-2.5-7B-Instruct(Qwen et al., 2025) as the target language models M. In this paper, we refer to Llama3-instruct, Mistral-7B-instruct-V2, and Qwen-2.5-7B-Instruct as Mistral7B, Llama3, and Qwen2.5, respectively. We use GPT-4o, MiniGPT-4o, and Claude 3.5 APIs as teacher models for response generation. Specifically, we use gpt-4o-mini-2024-07-18 and gpt-4o-2024-08-06 (OpenAI, 2023) from OpenAI, and claude-3-5-sonnet-20240620 (Anthropic, 2023) from Anthropic.

4.2 Datasets

We use English reasoning datasets referenced in the technical reports of LLaMA3 (Dubey and Abhinav Jauhri, 2024), Mistral (Jiang et al., 2023), and Qwen-2.5 (Qwen et al., 2025) (the three target models M in our experiments). We select datasets with at least 650 examples that can be evaluated via accuracy. If a dataset lacks sufficient training data, we reconstruct it to contain at least 400 training, 50 validation, and 200 testing examples.

For datasets with subcategories (e.g., MATH, MMLU, MMLU_PRO, API_BANK, AGIEVAL), we choose the challenging subcategory (i.e., with the lowest reported accuracy). For example, we include moral scenarios from MMLU, Professional Law from MMLU_PRO, Level 3 problems from API_BANK, geometry from MATH, and LogicQA from AGIEVAL; we also incorporate the Algebra subcategory from MATH as in (Anonymous, 2024).

Following (Anonymous, 2024), we train and evaluate the first 1,000 training and testing examples, generating up to 1,000 training examples per data generation strategy.

Main-experiment corpus. In total, our datasets include: **Mathematics:** GSM8K (Cobbe et al., 2021), MATH (Algebra) and MATH (Geometry) (Hendrycks et al., 2021); **Commonsense reasoning:** PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), Hellaswag (Zellers et al., 2019), and ECQA (Aggarwal et al., 2021); **Reading comprehension:** BoolQ (Clark et al., 2019) and SQuAD (Rajpurkar et al., 2016); **Aggregated benchmarks:** MMLU (Moral Scenarios) (Hendrycks et al., 2020), MMLU_PRO (Professional Law) (Wang et al., 2024), and AGIEval (LogicQA) (Zhong et al., 2023); **Coding:** MBPP (Austin et al., 2021); **Reasoning:** DROP (Dua et al., 2019) and ARC-Challenge (Clark et al., 2018); and **Tool-using:** API-BANK (Lv 3 problems) (Li et al., 2023). More details are in Table 16 (Appendix).

PlanBench Extension. We further evaluate the most challenging subtasks of PlanBench (Valmeekam et al., 2023)—those on which GPT-3 attains an accuracy below 20%. The subtasks comprise *plan generation*, *plan optimization*, *plan verification*, *plan reuse*, *plan generalization*, and *replanning*. Although we experimented with various prompt formats, Qwen consistently failed to solve any *execution* problems. Since our method relies on generating correct responses for use as in-context examples, we exclude the execution task from our evaluation. The remaining six categories remain sufficiently challenging and are not part of our main training benchmarks. We use them solely to analyze performance variance when models are trained on tasks that are very challenging. How do different response generation strategies affect performance variance under such conditions?

4.3 Data Generation Strategies

Given 1,000 samples, we use different strategies to generate target responses. For a fair comparison, we use the same prompts from (Anonymous, 2024) to generate responses, including **GPT-4o Answer Directly**, **Claud Answer Directly**, **MiniGPT-4o Answer Directly**, **Step-by-Step** and **Rewrite Ground Truth**. Besides, we design two new prompts named **GPT-4o Examples** and **Human Examples** on our own. Please refer to Appendix A.5 for details on each response construction method.

We provide ground truth to the teacher models and allow up to three attempts for data generation. If the first result is incorrect, we regenerate; otherwise, we stop. The same evaluation script used during testing is applied to check correctness.

5 Experiment

In this section, we treat each generation strategy from Section 4.3, and response-selection metrics from the related work section, as baselines. We then compare the average training outcomes of our method against these baselines across all tasks.

There are two benchmark sets, detailed in Section 4.2. The first is the Main-experiment corpus, which covers a diverse range of tasks and serves as the primary benchmark for evaluating both the general ranking ability of our metric and the average performance gains achievable by our method.

Since our goal is efficient data selection, we evaluate each metric using only a small subset of the training data. For each method, we repeat the process three times, each time selecting a different subset of size K=50 from the training dataset, and report the average performance across these runs. For example, one run may use the first 50 samples, another the second 50, and so on. The final result is computed as the average of these three evaluations.

5.1 Hyperparameters

We utilize the identical hyperparameter settings as referenced in (Anonymous, 2024). Specifically, for model fine-tuning, a learning rate of 2e-5, a batch size of 32, and a warm-up phase encompassing 10% of the total training iterations are applied. A cosine annealing schedule is implemented for the learning rate, and only the Q and V matrices of the LoRA parameters are fine-tuned with a rank of 8. All models undergo training and evaluation using half-precision arithmetic.

5.2 Evaluation Metrics

Accuracy. For every {model, dataset} pair, we let each ranking metric select the top-ranked response-generation strategy, fine-tune the model on data produced by that strategy, and record the resulting test accuracy. We then report the *macro average* of these accuracies across all evaluated tasks. This score answers the practical question: *If I trust a metric to choose my training data, how well will my model perform on average?*

Weighted Spearman correlation. To measure how closely a metric's ranking matches the gold ranking, we compute a weighted Spearman coefficient in which each task is weighted by the standard deviation of accuracies obtained from all candidate strategies; tasks whose choice of strategy matters more thus contribute more. The exact formula and implementation details are provided in Appendix A.3.

5.3 Comparison with Baseline Response Generation Strategies

Table 1 summarizes the average test accuracy obtained when the target model is fine-tuned on data produced by each response-generation strategy. For datasets that provide chain-of-thought (CoT) groundtruth, we additionally evaluate the **Rewrite Ground Truth** strategy. As this strategy is only applicable to CoT datasets and some datasets do not have CoT groundtruth, its results are excluded from the table to avoid skewing the overall averages; nevertheless, they are included in every metric that ranks candidate strategies on a per-task basis.

Effect of task-specific variance. Table 10 shows that the performance gap among generation strategies is highly task-dependent: some tasks show differences of several percentage points, while others are nearly insensitive to the chosen strategy. To quantify how much our method helps when the choice of generation strategy matters most, we group every {model, dataset} pair by the standard deviation (SD) of accuracies across baselines. **All tasks** include all pairs without filtering. **High-variance tasks** retain only those with $SD > 2\%$. **Very-high-variance tasks** retain only those with $SD > 4\%$. In the whole Main-experiment corpus, our approach delivers the highest mean accuracy,

Methods	STD Range	num of recorded data	mistral	llama 3 instruct	qwen	Avg Acc
Upper bound	All Data	51	59.39%	64.44%	71.41%	65.08%
Step-by-step			56.25%	60.94%	69.87%	62.35%
GPT-4 ICL examples			57.29%	62.03%	69.90%	63.07%
Human examples			56.95%	61.91%	70.22%	63.03%
Mini-GPT-4			56.72%	61.36%	70.13%	62.74%
GPT-4			56.92%	62.83%	69.89%	63.21%
Claude			57.45%	62.93%	70.30%	63.56%
Ours			58.33%	63.63%	70.38%	64.11%
Ours - Best	All Data	51	+0.87%	+0.69%	+0.08%	+0.55%
Ours - Avg of Others			+1.40%	+1.63%	+0.33%	+1.12%
Ours - Best	$SD > 2.00\%$	27	+1.24%	+0.98%	-0.93%	+0.88%
Ours - Avg of Others			+1.80%	+2.23%	+1.18%	+1.92%
Ours - Best	$SD > 4.00\%$	14	+2.21%	+1.44%	-1.32%	+1.49%
Ours - Avg of Others			+2.63%	+5.88%	+0.76%	+3.29%

Table 1: Comparison of our method with other response generation strategies, averaged over three subsets. Experiments are conducted on datasets from the Main-experiment corpus, introduced in Section 4.2. In this benchmark, Claude emerges as the strongest competitor among the baseline methods.

Methods	STD Range	num of recorded data	mistral	llama 3 instruct	qwen	Avg Acc	Weighted Spearman Pho
Upper bound	All Data	51	59.39%	64.44%	71.41%	65.08%	
IDF			56.78%	61.18%	69.55%	62.51%	0.019
skywork			56.32%	61.64%	70.26%	62.74%	0.250
CAR			56.36%	61.80%	70.31%	62.82%	0.258
perplexity			57.48%	63.03%	70.37%	63.63%	0.236
Ours			58.33%	63.63%	70.38%	64.11%	0.267
Ours - Perplexity	All Data	51	+0.85%	+0.60%	+0.01%	+0.48%	+0.031
Ours - Perplexity	$STD > 2.00\%$	27	+1.22%	+0.84%	-0.03%	+0.91%	+0.039
Ours - Perplexity	$STD > 4.00\%$	14	+1.83%	+2.71%	+0.02%	+1.82%	+0.077

Table 2: We compare our method against IDF (Li et al., 2024), Skywork (Liu et al., 2024), CAR (Xu et al., 2024), and Perplexity(Anonymous, 2024). The experiments are conducted on datasets from the Main-experiment corpus, introduced in Section 4.2. In this benchmark, Perplexity emerges as the strongest competitor among the baselines.

exceeding the strongest single baseline (Claude) by 0.55% and the mean of all baselines by 1.12%.

When we restrict evaluation to high-variance tasks, the average gain of our method over Claude rises to 0.88%; under the very-high-variance filter, the gain further climbs to 1.49%. Relative to the mean of all baselines, the improvements reach 3.29% on the very-high-variance subset. These results confirm that *self-aligned perplexity* is especially valuable when candidate generation strategies lead to widely divergent training outcomes.

5.4 Comparison with Alternative Response Selection Metrics

Table 2 reports results obtained with the same setup as in Section 5.3, but swapping the ranking metric. Across the full Main-experiment corpus, *self-aligned perplexity* achieves the best mean accuracy and the highest weighted Spearman correlation; standard perplexity is the closest baseline. **All tasks:** Using every training run, our metric surpasses standard perplexity by 0.48% in accuracy and by 0.031 in weighted Spearman ρ. **High-variance tasks (SD $>$ 2%):** The margins widen to 0.91% in accuracy and 0.039 in weighted ρ. **Very-high-variance tasks (SD $>$ 4%):** Gains further increase to 1.82% in accuracy and 0.077 in weighted ρ. These results mirror the trend observed in Section 5.3: the larger the performance spread among candidate strategies, the more our metric outperforms conventional perplexity, underscoring its value for selecting high-quality training data.

5.5 Performance Differences among Response Generation Strategies Can be Very Large

Candidate response-generation strategies can yield significantly different results depending on the task. To illustrate this, we evaluate various strategies on the PLANBENCH benchmark introduced in Section 4.2, which is designed to be more difficult than standard instruction-following datasets due to its long-horizon, goal-conditioned reasoning requirements. As shown in Table 6 and Tabel 17(Appendix), training outcomes vary significantly across methods, underscoring the importance of selecting an appropriate generation strategy. Our self-aligned perplexity metric improves accuracy by an average of 1.77% over standard perplexity and 5.24% over the mean performance of all strategies. The results further demonstrate that,

Target Response style	Model	Task	PPL	$\mathrm{S_{sbs}PPL}$	$\mathrm{S_{cad}PPL}$	$\mathrm{S_rPPL}$
Step by Step(sbs)	Mistral7B	ECQA	**4.476**	**3.695**	4.85	4.329
GPT4 Answer Directly(cad)			5.551	4.116	**4.768**	4.456
Redundant(r)			4.944	4.334	5.615	**4.326**
Step by Step(sbs)	Mistral7B	PIQA	**4.290**	**3.816**	5.968	4.028
GPT4 Answer Directly(cad)			6.277	4.053	**5.962**	4.250
Redundant(r)			4.547	3.919	6.724	**4.027**

Table 3: Examples showing that in-context perplexity favors responses matching the style of the in-context example. PPL is standard perplexity; $\mathrm{S_{sbs}PPL}$, $\mathrm{S_{cad}PPL}$, and $\mathrm{S_rPPL}$ use step-by-step, GPT-4o Answer Directly, and redundant responses as context, respectively.

as the optimal model varies across datasets and continues to shift as APIs evolve, model-aware selection metrics like self-aligned perplexity remain critical.

6 Ablation Study

6.1 Why Self-Aligned Perplexity Outperforms Traditional Perplexity

Traditional perplexity is sensitive to surface-level stylistic cues, so a low score does not necessarily mean the response “feels” familiar to the model. We therefore anchor the metric on the model’s own zero-shot prediction: the closer a candidate lies to this anchor, the more familiar it should be. Injecting that prediction as a single in-context example reshapes the probability distribution, yielding a self-aligned perplexity that more faithfully reflects the response’s true familiarity.

When perplexity fails. According to Table 3, On ECQA, a deliberately *redundant* answer(see Appendix A.5.1 for how we construct this dataset) scores 4.94 in raw perplexity, while the terser, higher-quality *GPT-4-direct* answer scores 5.55 (Table 3). A similar pattern appears in PIQA (4.55 vs. 6.28). Thus, lower perplexity can sometimes reflect wordiness rather than genuine familiarity with the model’s preferred style.

How self-aligned perplexity helps. According to Table 3, adding a single in-context example can realign the perplexity scores. For the GPT-4–style response on ECQA, the raw perplexity (PPL) is 5.551, which is higher than the redundant-style response (4.944). After prepending an in-context example drawn from another GPT-4–style answer, the GPT-4 response’s perplexity drops to 4.768. In contrast, when the same example is added to the redundant and step-by-step responses, their perplexities increase from 4.944 and 4.476 to 5.615 and 4.850, respectively.

Weighted Method	Model	Accuracy	Spearman’s ρ
Ours	Mistral7B	58.3%	0.301
TTT (lr=2e-5)		57.8%	0.103
TTT (lr=2e-4)		57.9%	0.545
Ours	Llama3	63.6%	0.147
TTT (lr=2e-5)		61.3%	0.083
TTT (lr=2e-4)		62.7%	0.372
Ours	Qwen2.5	70.4%	0.449
TTT (lr=2e-5)		70.2%	-0.040
TTT (lr=2e-4)		70.9%	0.256
Ours	Average	64.1%	0.267
TTT (lr=2e-5)		63.1%	0.077
TTT (lr=2e-4)		63.8%	0.297

Table 4: Ours (K=50, avg. of 3 subsets) vs. Train-then-Test (TTT) (K=100, 1 seed) on Main-experiment corpus.

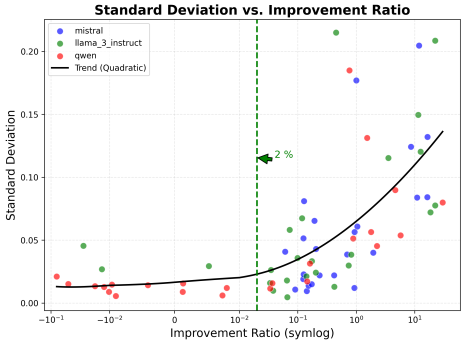

Figure 1: When the improvement ratio is high, the standard deviation of training outcomes across different response-generation strategies tends to be larger.

A similar effect occurs on PIQA: the GPT-4 response on Mistral has an initial perplexity of 6.277, higher than the redundant style (4.570). With a GPT-4 in-context example, its perplexity decreases to 5.962, while the redundant style’s perplexity rises to 6.724.

Across the two tasks, using the model’s own prediction as the in-context anchor consistently lowers the score for its native style by 0.6–1.5 points, restoring the correct ordering and yielding rankings that track downstream fine-tuning gains.

6.2 Why Do Some Datasets Show Greater Variance in Training Outcomes?

We observed a striking regularity across tasks: whenever enlarging the training set from 100 to 1 000 examples yields little or no accuracy gain, the choice of response-generation method matters equally little. Conversely, tasks that continue to improve with more data show pronounced performance gaps between generation strategies.

Let the **improvement ratio** be defined as $\mathrm{Acc}_{1000}/\mathrm{Acc}_{100} - 1$, representing the relative gain from increasing the training size ten-fold. Figure 1 plots log(improvement ratio) (x-axis) against the standard deviation of accuracies across generation

	0–10	10–20	20–30	0–30	30–60	60–90	0–50	50–100	100–150
Accuracy (%)	63.92	64.28	63.51	64.15	63.96	64.08	64.15	63.98	64.20
Weighted ρ	0.242	0.316	0.184	0.244	0.251	0.217	0.249	0.233	0.319
	0–100	100–200	200–300	0–200	0–300	-	-	-	-
Accuracy (%)	64.18	64.12	64.08	64.12	64.12	-	-	-	-
Weighted ρ	0.269	0.280	0.287	0.288	0.287	-	-	-	-

Table 5: Performance on different subsets when ranking with self-aligned perplexity. An interval such as *60–90* means starting at index 60 and using the next 30 instances (indices 60–89) for ranking calculation.

methods (y-axis). A clear positive trend emerges: once the improvement ratio exceeds roughly 2%, the variance among methods rises sharply; below this threshold, it is nearly zero. We plot Figure 1 using training results from all tasks in the Main-experiment corpus and PlanBench.

The results suggest that divergence across generation strategies is greatest exactly when the dataset still offers headroom for improvement. On such high-variance tasks, selecting the right response-generation method is critical, underscoring the value of our self-aligned perplexity criterion.

6.3 Our Method vs. Train-Then-Select

One natural (but computationally expensive) approach to select the optimal response generation strategy is to adopt a Train-Then-Select (TTS) procedure. In this way, we first generate a small dataset (e.g., 100 samples) using each candidate strategy. For each dataset, we train the target model and evaluate its performance. We then rank the strategies based on the results and choose the best-performing one to generate the remainder of the dataset.

When evaluating TTS, we train the target model on 100 samples under two settings: **1) Standard Training:** A learning rate of 2e-5 for 20 epochs (matching our main setup). The performance accuracies for each strategy under this setting is in the Table 14. **2) Intense Training:** A learning rate of 2e-4 for 40 epochs. The performance accuracies for each strategy under this setting is in Table 15.

After ranking the strategies using TTS, we compare their performance with ours. In Table 4, despite using less data and requiring no training, validation, or testing computations for strategy selection, our method achieves better average accuracy and comparable weighted Spearman correlation.

6.4 Stability of Our Method

As shown in Table 5, accuracy generally improves as the subset size grows, and the overall performance is consistent across ranges. Small subsets sometimes degrade accuracy (values highlighted in red); thus, we recommend using at least 30 samples or even 50 samples for the best performance. Specifically, "0–10", "10–20", and "20–30" denote the first, second, and third batches of ten training examples, respectively, while "0–50" and "0–100" correspond to the first 50 and 100 examples. When the subset size reaches 50 or more, average accuracy stabilises. The weighted Spearman correlation (ρ) also increases with larger subsets, but the gains taper off once the subset size exceeds 50.

6.5 Ground Truth vs. Synthetic Data

As shown in Table 10 (Appendix), when ground truth is provided in natural language (e.g., GSM8K, MATH, ECQA, MBPP), training on ground truth is less effective than on synthetic data. This is because LLMs are more familiar with LLM-generated data, as demonstrated by Anonymous (2024). However, when the ground truth is written as a gold label without a CoT inference process, training on the gold label can sometimes outperform training on CoT synthetic data within the same domain. However, in Table 8 (Appendix), training on gold labels harms cross-domain performance more than training on synthetic data. Besides, in real-life scenarios, training on natural language data is crucial, as users expect to see the rationale behind the final prediction made by LLMs.

7 Conclusions

In this paper, we present a novel and scalable approach for selecting the optimal response-generation strategy to train large language models. We introduce a new metric, self-aligned perplexity, which more effectively evaluates the alignment between a target model and its response options compared to traditional perplexity. We demonstrate that choosing the optimal generation strategy based on self-aligned perplexity leads to substantial improvements in model performance, particularly on tasks with high performance variance. We hope our work will inspire researchers who use perplexity as a downstream metric or who wish to build the most effective instruction tuning datasets.

8 Limitations

While recent open-source "O1" models support in-context chain-of-thought reasoning, our evaluations focus on larger teacher models. We used meta prompts to elicit reasoning steps but did not test on O1 models. We believe our style-alignment approach still applies, though further validation on smaller or differently pretrained models is needed. We leave broader scaling studies and extensions to other model families for future work.

References

Shourya Aggarwal, Divyanshu Mandowara, Vishwajeet Agrawal, Dinesh Khandelwal, Parag Singla, and Dinesh Garg. 2021. Explanations for commonsenseqa: New dataset and models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3050–3065.

Anonymous. 2024. I learn better if you speak my language: Understanding the superior performance of fine-tuning large language models with LLM-generated responses. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 10225–10245, Miami, Florida, USA. Association for Computational Linguistics.

Anthropic. 2023. Claude 3.5 api. https://docs.anthropic.com/claude. Accessed: Month Day, Year.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, and 1 others. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Javier De la Rosa, Eduardo G Ponferrada, Paulo Villegas, Pablo Gonzalez de Prado Salas, Manu Romero, and Marıa Grandury. 2022. Bertin: Efficient pre-training of a spanish language model using perplexity sampling. *arXiv preprint arXiv:2207.06814*.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *Preprint*, arXiv:1903.00161.

Abhimanyu Dubey and etc. Abhinav Jauhri. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. 2023. Specializing smaller language models towards multi-step reasoning. In *International Conference on Machine Learning*, pages 10421–10430. PMLR.

Hila Gonen, Srini Iyer, Terra Blevins, Noah A Smith, and Luke Zettlemoyer. 2022. Demystifying prompts in language models via perplexity estimation. *arXiv preprint arXiv:2212.04037*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*.

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. *arXiv preprint arXiv:2103.03874*.

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *Preprint*, arXiv:2305.02301.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger P Levy. 2020. A systematic assessment of syntactic generalization in neural language models. *arXiv preprint arXiv:2005.03692*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Minki Kang, Seanie Lee, Jinheon Baek, Kenji Kawaguchi, and Sung Ju Hwang. 2023. Knowledge-augmented reasoning distillation for small language models in knowledge-intensive tasks. *Preprint*, arXiv:2305.18395.

Seungone Kim, Juyoung Suk, Xiang Yue, Vijay Viswanathan, Seongyun Lee, Yizhong Wang, Kiril Gashteovski, Carolin Lawrence, Sean Welleck, and Graham Neubig. 2024. Evaluating language models as synthetic data generators. *Preprint*, arXiv:2412.03679.

Tom Kocmi and Ondrej Bojar. 2017. Curriculum learning and minibatch bucketing in neural machine translation. *arXiv preprint arXiv:1707.09533*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024. 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.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. Api-bank: A comprehensive benchmark for tool-augmented llms. *arXiv preprint arXiv:2304.08244*.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, Wenhu Chen, and Xifeng Yan. 2022. Explanations from large language models make small reasoners better. *Preprint*, arXiv:2210.06726.

Chris Yuhao Liu, Liang Zeng, Jiacai Liu, Rui Yan, Jujie He, Chaojie Wang, Shuicheng Yan, Yang Liu, and Yahui Zhou. 2024. Skywork-reward: Bag of tricks for reward modeling in llms. *Preprint*, arXiv:2410.18451.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *Preprint*, arXiv:2306.08568.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2023. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, Toronto, Canada. Association for Computational Linguistics.

Dheeraj Mekala, Alex Nguyen, and Jingbo Shang. 2024. Smaller language models are capable of selecting instruction-tuning training data for larger language models. *arXiv preprint arXiv:2402.10430*.

OpenAI. 2023. Gpt-4 api. `https://platform.openai.com/docs/models/gpt-4`. Accessed: Month Day, Year.

Emmanouil Antonios Platanios, Otilia Stretcu, Graham Neubig, Barnabas Poczos, and Tom M Mitchell. 2019. Competence-based curriculum learning for neural machine translation. *arXiv preprint arXiv:1903.09848*.

Qwen, :, 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, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Leonardo Ranaldi and Andre Freitas. 2024. Aligning large and small language models via chain-of-thought reasoning. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1812–1827, St. Julian's, Malta. Association for Computational Linguistics.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106.

Trieu H Trinh, Yuhuai Wu, Quoc V Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36:38975–38987.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, and 1 others. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, and Radha Poovendran. 2024. Stronger models are not stronger teachers for instruction tuning. *Preprint*, arXiv:2411.07133.

Zhenyu Xu and Victor S Sheng. 2024. Detecting ai-generated code assignments using perplexity of large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23155–23162.

Zhaorui Yang, Tianyu Pang, Haozhe Feng, Han Wang, Wei Chen, Minfeng Zhu, and Qian Liu. 2024. Self-distillation bridges distribution gap in language model fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1028–1043, Bangkok, Thailand. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Hanyu Zhang, Xiting Wang, Xiang Ao, and Qing He. 2024. Distillation with explanations from large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 5018–5028.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. 2023. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*.

A Appendix

A.1 Can we get performance gain if we simply put all of the response variants together?

Selecting the optimal data generation strategy remains essential, even when resources or funding are unlimited. As shown in Table 7, simply combining six types of synthetic data (Total $n_{\text{train}} = 6000$) does not guarantee a performance gain over selecting the best synthetic training data. For example, after training the Llama3 model on API-Bank using all six types of synthetic data, the evaluation accuracy is only 49%, much lower than when selecting the Claude Answer Directly data (54.7%). Indeed, according to Table 7, if we combine the mixture of the top three data generation strategies (Mixture of good $n_{\text{train}} = 3000$), the performance is almost always better than if we simply combine all of the data together (Total $n_{\text{train}} = 6000$). This underscores the importance of selecting data generation strategies, even if we can afford large-scale synthetic data generation and training.

A.2 The Impact of Accuracy of the Synthetic Data on Training Outcomes

In our experiment, we aim to ensure the correctness of generated answers by validating them against ground truth answers. Our research seeks to identify the best strategy for generating the optimal version of an answer. In other words, we can adjust data generation strategies to ensure correctness.

In our experiments, we use ground truth answers to guide the generated answers for nearly all datasets, with the only exceptions being mathematical problems. This follows the setting of the paper to maintain consistency with previous work (Anonymous, 2024). This approach might be acceptable since closed-source APIs tend to generate accurate answers. For GSM8K and Math Algebra, GPT-4o, Claude, and MiniGPT-4o achieve accuracies of 90% or above.

To evaluate the impact of accuracy on training outcomes, we conducted the following experiment. As shown in Table 9, we tested three approaches: training on the full dataset, using only correct predictions, and replacing incorrect predictions with rewritten ground truth. These approaches showed less than a 2% improvement overall. Note that in this experiment, GPT-4 refers to the gpt-4-1106-preview API, rather than the gpt-4o-2024-08-06 API, which was used in all other experiments in the paper. The mathematical capabilities of GPT-4o, GPT-4-Mini, and Claude are similar on Math Algebra tasks. Therefore, we used the gpt-4-1106-preview API, which has a weaker ability to solve Math Algebra problems. The benifit of using it is that it makes more mistakes on GSM8K so that we can better evaluate the influence of accuracy. We used this API once to generate the data and train the model from there.

According to the table, the overall benefit of replacing incorrect examples with rewritten ground truth or removing incorrect examples has minimal impact on the overall training outcomes.

A.3 Weighted Spearman's Rank Correlation Coefficient

Spearman's rank correlation (ρ) measures how well two orderings agree, ignoring absolute values. Because some of our $\{\text{model}, \text{dataset}\}$ pairs exhibit far larger performance gaps among response-generation strategies than others, we assign higher importance to pairs whose choice of strategy matters more. We therefore adopt a *weighted* variant of Spearman's correlation in which each item is given a non-negative weight w_i.

Definition. Let $R_{1,i}$ and $R_{2,i}$ be the ranks of the i-th item under two orderings and let w_i be its weight.

Methods	STD Range	num of recorded data	mistral	llama 3 instruct	qwen	Avg Acc	Weighted Spearman Pho
Upper bound	All Data	18.0	52.88%	54.87%	41.87%	49.88%	
Step-by-step			37.56%	44.48%	31.06%	37.70%	
GPT-4 ICL examples			40.86%	45.84%	36.86%	41.19%	
Human examples			45.01%	41.82%	29.89%	38.91%	
Mini-GPT-4			38.68%	40.29%	30.51%	36.49%	
GPT-4			39.33%	41.86%	31.08%	37.42%	
Claude			51.89%	50.09%	37.27%	46.42%	
Ours			45.29%	50.63%	41.26%	45.73%	
Ours - Claude			-6.60%	+0.55%	+3.98%	-0.69%	
Ours - Avg of Others			+3.07%	+6.57%	+8.48%	+6.04%	
Upper bound	All Data	18.0	52.88%	54.87%	41.87%	49.88%	
IDF			43.29%	46.17%	34.91%	41.46%	0.331
skywork			43.66%	46.01%	36.12%	41.93%	0.226
CAR			41.87%	44.99%	35.00%	40.62%	0.239
perplexity			42.69%	48.50%	36.44%	42.54%	0.226
Ours			45.29%	50.63%	41.26%	45.73%	0.239
Ours - Perplexity			+2.59%	+2.14%	+4.81%	+3.18%	+0.012

Table 6: Comparison of our method with other metrics or response generation methods on 6 subsets from the PlanBench dataset as introduced by PlanBench Extension, introduced in Section 4.2. We compare our method against IDF (Li et al., 2024), Skywork (Liu et al., 2024), CAR (Xu et al., 2024), and Perplexity(Anonymous, 2024).

Denote the weighted means

$$\bar{R}_1 = \frac{\sum_{i=1}^{n} w_i R_{1,i}}{\sum_{i=1}^{n} w_i}, \qquad \bar{R}_2 = \frac{\sum_{i=1}^{n} w_i R_{2,i}}{\sum_{i=1}^{n} w_i}.$$

The **weighted Spearman correlation** is then the weighted Pearson correlation between the rank vectors:

$$\rho_w = \frac{\sum_{i=1}^{n} w_i \left(R_{1,i} - \bar{R}_1\right)\left(R_{2,i} - \bar{R}_2\right)}{\sqrt{\sum_{i=1}^{n} w_i \left(R_{1,i} - \bar{R}_1\right)^2} \sqrt{\sum_{i=1}^{n} w_i \left(R_{2,i} - \bar{R}_2\right)^2}}.$$

Choice of weights. For each $\{\text{model}, \text{dataset}\}$ pair, we first train the target model on data produced by every candidate response-generation method and record the resulting accuracies. The weight w_i is set to the *standard deviation* of these accuracies. Intuitively, tasks in which the strategies yield very different outcomes (w_i large) are more informative when judging a ranking metric, so they contribute more to ρ_w.

Interpretation. A value of $\rho_w \approx 1$ indicates that the metric produces a ranking almost identical to the gold ranking, with higher-variance tasks influencing the score most strongly. Conversely, $\rho_w \approx 0$ implies no weighted monotonic relationship, and $\rho_w \approx -1$ signals an inverse agreement.

Throughout the main text and Appendix, all reported "Spearman" results actually correspond to this weighted formulation.

A.4 Data Selection Rationale for the Benchmark

The datasets included in our benchmark, drawn from the Mistral, Llama, and Qwen benchmarks, were selected according to a specific set of rules designed to ensure relevance and suitability. These rules are as follows:

1. Sufficient Dataset Size: We only included datasets where the combined size of the training, validation, and testing sets exceeded 650 samples. This threshold was chosen to ensure sufficient data for robust model evaluation.

2. Accuracy as Evaluation Metric: A key requirement was that the dataset could be evaluated using accuracy as the primary metric. This allows for a clear and quantifiable assessment of model performance.

3. English Question-Answering Format: All selected datasets are in an English question-and-answer format to maintain consistency and focus on English language reasoning abilities.

4. Focus on Reasoning Tasks: The underlying task presented by each dataset must involve reasoning skills. This ensures that the benchmark effectively assesses the models' ability to reason and infer.

A detailed justification for the inclusion or exclusion of each dataset can be found in Table 16.

A.5 Correctness Filter

Without supervised fine-tuning (SFT), $M(x)$ may generate incorrect responses, making cosine similarity calculations between $M(x)$and $\hat{y}$ unreliable.

Method	Model	DROP	Hellaswag	API-Bank
Best $n_{\text{train}} = 1000$	Mistral7B	0.743	0.675	0.559
Avg $n_{\text{train}} = 1000$		0.726	0.646	0.446
Total $n_{\text{train}} = 6000$		0.740	0.738	0.555
Mixture of good $n_{\text{train}} = 3000$		0.770	0.731	0.555
Mixture of good $n_{\text{train}} = 1000$		0.744	0.686	0.535
Average of all $n_{\text{train}} = 1000$		0.711	0.686	0.433
Best $n_{\text{train}} = 1000$	Llama3	0.805	0.718	0.547
Avg $n_{\text{train}} = 1000$		0.778	0.711	0.392
Total $n_{\text{train}} = 6000$		0.810	0.738	0.490
Mixture of good $n_{\text{train}} = 3000$		0.812	0.745	0.527
Mixture of good $n_{\text{train}} = 1000$		0.804	0.728	0.490
Average of all $n_{\text{train}} = 1000$		0.771	0.705	0.457
Best $n_{\text{train}} = 1000$	Qwen2.5	0.814	0.739	0.461
Avg $n_{\text{train}} = 1000$		0.804	0.719	0.413
Total $n_{\text{train}} = 6000$		0.798	0.748	0.584
Mixture of good $n_{\text{train}} = 3000$		0.824	0.738	0.584
Mixture of good $n_{\text{train}} = 1000$		0.818	0.742	0.490
Average of all $n_{\text{train}} = 1000$		0.778	0.712	0.412

Table 7: **Best** represents the best data generation strategy for the task with the target model. **Total** combines all strategies, yielding $n_{\text{train}} = 6000$. **Mixture of good** ($n_{\text{train}} = 3000$) includes the top three strategies with 1000 samples each, while **Mixture of good** ($n_{\text{train}} = 1000$) has about 333 samples per strategy.

To alleviate this, we introduce a filtering mechanism to filter out the incorrect $M(x)$. We notice that for mathematical problems, the correct final answer typically appears as the last number in $M(x)$. Therefore, for Math-related tasks, we use regular expressions (regex) to extract the last number from the prediction and compare it directly with the ground truth. For other types of problem, we use the Qwen2.5-Instruct 7b model to extract the predicted label from the model output. We then compare this extracted label with the true gold label; if they match, we consider the prediction correct by default. The code we used to extract labels are detailed in Appendix A.6

Response Construction Details

Ground Truth: This strategy uses the original ground-truth responses from the datasets as target outputs. Since our focus is on selecting effective chain-of-thought (CoT) target responses, we apply this method to datasets that include human-annotated CoT reasoning steps, such as GSM8K, MATH, ECQA, MBPP. When human-annotated CoT is unavailable, we use the gold label as ground truth.

GPT-4o Answer Directly, **Claud Answer Directly**, and **MiniGPT-4o Answer Directly** generate responses based on questions and the ground truth using GPT-4o, Claude 3.5 and Mini-GPT4, respectively. **Rewrite Ground Truth:** Direct GPT-4o to restyle the ground truth in its own language. This method is only applicable to GSM8K, MATH Algebra, ECQA. The other tasks's ground truth consists of target labels without any human-annotated chain-of-thought (CoT) reasoning, making rewriting infeasible. **Step-by-Step:** instructs GPT-4o to generate step-by-step responses based on questions and ground truth. **GPT-4o Examples:** To facilitate problem-solving, we provide GPT-4o with two high-quality, expert-selected in-context examples of its own responses. GPT-4o is then tasked with generating new responses based on these examples. **Human Examples:** To aid GPT-4o in understanding problem-solving for these datasets, we provide two carefully chosen human-written examples as context. GPT-4o then uses these examples to generate new responses. We put more details in Section A.5.2 in Appendix.

A.5.1 Prompt for Self-Aligned Perplexity

Redundant Prompt

We construct redundant prompts to demonstrate that the perplexity of the redundant target responses is lower than that of GPT-4's answers. Perplexity primarily reflects how fluent the language is and how well the language style aligns with the model,

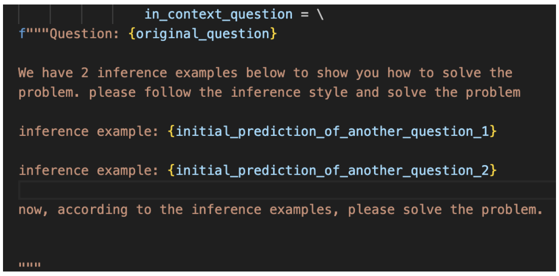

Figure 2: Prompt that we used for self-aligned perplexity

but it places less emphasis on semantic meaning.

```
f"""We have the question and the groundtruth.
    ↪ Given on the groundtruth, please
    ↪ reformat the groundtruth so that it
    ↪ answer the question in a step by step
    ↪ redundant manner. Be as repetitive and
    ↪ step by step and redundant as possible.


Question: {question}
Groundtruth: {groundtruth}


1. We wish you to reformat a new groundtruth.
    ↪ The new groundtruth are reformated a
    ↪ new groundtruth which solve the
    ↪ problem as steo by step and redundant as
    ↪  possible.
2. You will pretend as you do not know the
    ↪ groundtruth, because we will use your
    ↪ step by step redundant answer as target
    ↪ responses to train our model.
3. (important format) You must generate the
    ↪ groundtruth with the step by step
    ↪ redundant inference process directly.
    ↪ Please not saying anything like 'sure I
    ↪ can help you with' or 'sure, i will not
    ↪ mention the gold label'
4. (important format) You will inference first then
    ↪  put the Final Answer: {gold_label}

at the end like this

INFERENCE HERE
Final Answer: {gold_label}
"""
```

Declaration of Independence

This is the part of Declaration of Independence that we use in the experiment in Table 4.

```
f"""The Unanimous Declaration of the Thirteen
    ↪ United States of America . When, in the
    ↪ course of human events, it becomes
    ↪ necessary for one people to dissolve the
    ↪ political bonds which have connected
    ↪ them with another, and to assume among
    ↪  the powers of the earth, the separate and
    ↪ equal station to which the laws of nature
    ↪ and of nature\''s God entitle them, a
    ↪ decent respect to the opinions of
    ↪ mankind requires that they should
    ↪ declare the causes which impel them to
    ↪ the separation

We hold these truths to be self–evident, that all
    ↪ men are created equal, that they are
    ↪ endowed by their Creator with certain
    ↪ unalienable rights, that among these are
    ↪ life, liberty and the pursuit of happiness.
    ↪ That to secure these rights, governments
    ↪ are instituted among men, deriving their
    ↪ just powers from the consent of the
    ↪ governed. That whenever any form of
    ↪ government becomes destructive to these
    ↪  ends, it is the right of the people to alter
    ↪ or to abolish it, and to institute new
    ↪ government, laying its foundation on
    ↪ such principles and organizing its powers
    ↪  in such form, as to them shall seem most
    ↪  likely to effect their safety and
    ↪ happiness."""
```

Self-Aligned In-Context Prompt for Perplexity Calculation This prompt shows how we add self - generated initial predictions from other questions as in - context examples for perplexity calculation.

```
in_context_question = \
f"""Question: {original_question}

We have an inference example below to show you
    ↪  how to solve the problem. please follow
    ↪ the inference style and solve the problem

inference example: {
    ↪ initial_prediction_of_another_question}


now, according to the inference example, please
    ↪ solve the problem.

IMPORTANT FORMAT REQUIREMENT:
```

```
When you solve the problem, you need to make the problem solving process and  language as similar to the inference example above as possible. If the inference process does not follow at the prediction before, you have to correct your style at anytime when you notice the style is not following the inference example. this is the most important requirement. please follow it.

"""
```

A.5.2 Data Generation Strategies

We instruct GPT-4, Claude 3.5, Mini-GPT4 to generate different of target responses using different target reponse generation strategies.

GPT-4/Claude 3.5/Mini-GPT4 Answer Directly: This prompt is from (Anonymous, 2024). For tasks involving mathematics and coding, we submit the problems from our training dataset directly to GPT-4 or Claude 3.5 to obtain their solutions. In the case of classification tasks, we provide these models with the input questions alongside the correct labels (excluding any human-generated explanations) and utilize their outputs. These generated answers are then paired with the original questions to form the GPT-4/Claude 3.5 Direct Answer Training Dataset.

To ensure that the models develop their own problem-solving and analytical capabilities, we deliberately exclude any solutions or rationales related to math, coding, or classification tasks. This approach prevents the models from simply mimicking the ground truth processes, which could otherwise result in some of GPT-4's predictions lacking its unique reasoning style. Such mimicry would undermine the reliability of our perplexity measurements, which are designed to evaluate how effectively a language model handles outputs from other models.

The prompt below is designed to guide GPT-4/Claude 3.5 in generating responses without relying on the ground truth solutions:

```
"""We have the {question}


1. We wish you to answer the question.
2. You must answer the question (with inference process) directly without say anything else. Please not saying anything 'like sure I can help you with' or 'sure, i will not mention the gold label'
3. You will inference first then put the Final Answer (NUMBER_HERE) at the end of the prediction like this

INFERENCE HERE
Final Answer: NUMBER_HERE"""
```

Rewrite Ground Truth: This prompt is from (Anonymous, 2024). In this approach, we provide GPT-4 and Claude 3.5 with the ground truth data, which includes human-annotated rationales and detailed problem-solving steps. The goal is to have GPT-4 and Claude 3.5 rephrase the ground truth content using their own linguistic styles.

The subsequent prompt guides GPT-4 and Claude 3.5 to generate the GPT-4/Claude 3.5 Response (Rewrite GT) output.

```
"""Given the question: {question}
and the groundtruth: {groundtruth}

Please states the prediction in your own words. The groundtruth is 100% correct. You should not change the problem solving logic of the groundtruth. just restates it in  your own words.

1. You will pretend as you do not know the groundtruth, because we will use your prediction as target labels to train our model.
2. (important format) You must generate the groundtruth directly. Please not saying anything like 'sure I can help you with' or 'sure, i will not mention the gold label'

3. (important format) Please make sure the Final Answer: {gold_label} is placed at the end of the modified prediction."""
```

Step-by-step: This prompt is from (Anonymous, 2024). We instruct GPT-4 and Claude 3.5 to methodically address each problem by breaking it down into sequential steps. For tasks involving mathematics and coding, we present the problems directly from our training dataset to these models to obtain their solutions. In classification tasks, we provide GPT-4 and Claude 3.5 with the correct labels (excluding any human-generated explanations) along with the input questions, and then utilize their detailed, step-by-step responses. These generated answers are subsequently paired with the original questions to form the GPT-4/Claude 3.5 Step-by-Step Response (No GT) Dataset.

To ensure that the models develop their own unique problem-solving and analytical approaches, we intentionally exclude the solutions or rationales for the mathematics, coding, or classification tasks. This prevents the models from simply mimicking the problem-solving and analytical methods found in the ground truth data. Including such processes could result in some of GPT-4's and Claude 3.5's outputs not reflecting their inherent reasoning styles, thereby compromising the accuracy of our perplexity measurements. These measurements are designed to assess how effectively a language model can handle outputs generated by other language models.

The following prompt directs GPT-4 and Claude 3.5 to generate the GPT-4/Claude 3.5 Step-by-Step Response (No GT) responses.

```
"""
We have the question and the groundtruth. Please reformat the groundtruth in step by step manner with details.

Question: {question}
Groundtruth: {groundtruth}

1. We wish you to regenerate a new groundtruth. The new groundtruth solve the problem step by step. If you believe the groundtruth is not detail enough, you could add details.
2. You will pretend as you do not know the groundtruth, because we will use your prediction as target labels to train our model.
3. (important format) You must generate the groundtruth with the step by step inference process directly. Please not saying anything like 'sure I can help you with' or 'sure, i will not mention the gold label'
4. (important format) You will inference first then put the Final Answer: {gold_label}

at the end like this

INFERENCE HERE
Final Answer: {gold_label}
"""
```

GPT-4o with GPT-4o Examples: We developed this prompt specifically for the API-Bank and Plan-Bench datasets. This prompt utilizes GPT-4's own accurate generations as examples to help GPT-4 not only better understand the task but also demonstrate how to solve the problems effectively. The prompt below is an example that we used to generate target responses for the API-Bank dataset.

```
"""
We have the {question} and the groundtruth {gold_label}


1. We wish you to answer the question. We will use your answer to train our model, thus you will answer and pretend as not knowing the gold_label.
2. You must answer the question (with inference process) directly without say anything else. Please not saying anything 'like sure I can help you with' or 'sure, i will not mention the gold label'
3. You will inference first then put the Final Answer ({gold_label}) at the end of the prediction like this

INFERENCE HERE
Final Answer: {gold_label}

Example 1:

Question : {q1}

groundtruth: API-Request: [ToolSearcher(keywords='healthcare provider appointment availability checker')]

Inference: The user is requesting to find a healthcare provider (specifically a cardiologist) in Los Angeles for a check-up appointment. The available API description indicates that the ToolSearcher API can be used to search for relevant tools based on the provided keywords. Therefore, the first step is to search for a tool that can help find a healthcare provider appointment availability checker.

Final Answer: API-Request: [ToolSearcher(keywords='healthcare provider appointment availability checker')]
```

```
Example 2:

question: {q2}

groundtruth: API−Request: [HealthcareProviderAppointmentChecker(specialty='cardiologist', location='Los Angeles')]

Inference: The first API request was successfully made to find a tool for checking healthcare provider appointment availability. The HealthcareProviderAppointmentChecker API was identified, which requires specialty and location as input parameters to search for available appointment slots. Based on the user's request to find a cardiologist in Los Angeles for a check−up appointment, the next API call should use this information.

Final Answer: API−Request: [HealthcareProviderAppointmentChecker(specialty='cardiologist', location='Los Angeles')]


Example 3:

question: {q3}

groundtruth: API−Request: [ToolSearcher(keywords='healthcare provider appointment scheduler')]

Inference: The user initially searched for an availability checker and found available appointment slots for a cardiologist in Los Angeles. Now, the user needs to schedule an appointment, so the next step is to find a tool for scheduling healthcare provider appointments using the ToolSearcher API with relevant keywords.

Final Answer: API−Request: [ToolSearcher(keywords='healthcare provider appointment scheduler')]

We have the {question} and the groundtruth {gold_label}


1. We wish you to answer the question. We will use your answer to train our model, thus you will answer and pretend as not knowing the gold_label.
2. You must answer the question (with inference process) directly without say anything else. Please not saying anything 'like sure I can help you with' or 'sure, i will not mention the gold label'
3. You will inference first then put the Final Answer ({gold_label}) at the end of the prediction like this

INFERENCE HERE
Final Answer: {gold_label}

"""
```

GPT-4 with Human Written Examples: We developed this prompt specifically for the API-Bank and Plan-Bench datasets. This prompt utilizes human written examples to help GPT-4 not only better understand the task but also demonstrate how to solve the problems effectively. The prompt below is an example that we used to generate target responses for the API-Bank dataset.

```
"""
We have the {question} and the groundtruth {gold_label}


1. We wish you to answer the question. We will use your answer to train our model, thus you will answer and pretend as not knowing the gold_label.
2. You must answer the question (with inference process) directly without say anything else. Please not saying anything 'like sure I can help you with' or 'sure, i will not mention the gold label'
3. You will inference first then put the Final Answer ({gold_label}) at the end of the prediction like this

INFERENCE HERE
Final Answer: {gold_label}
```

Example 1:

Question : {q1}

groundtruth: API–Request: [ToolSearcher(keywords='healthcare provider appointment availability checker')]

Inference: The user is requesting to find a healthcare provider (specifically a cardiologist) in Los Angeles for a check–up appointment. The first step should be to search for a tool that can help find a healthcare provider appointment availability checker. To accomplish this, we choose the ToolSearcher API from the available APIs. The ToolSearcher API is used to search for relevant tools based on the provided keywords according to the description. We need to fill out the keywords according to the description. The keywords could be 'healthcare provider appointment availability checker.' Therefore, the next step (which is also the first step) is:

Final Answer: API–Request: [ToolSearcher(keywords='healthcare provider appointment availability checker')]

Example 2:

question: {q2}

groundtruth: API–Request: [HealthcareProviderAppointmentChecker(specialty='cardiologist', location='Los Angeles')]

Inference: According to the API call history, the user has called the ToolSearcher API and found the HealthcareProviderAppointmentChecker API. The next step is to fill out the input parameters for HealthcareProviderAppointmentChecker and use it to find healthcare provider appointment availability. The input parameters are specialty and location. The user wants to find a cardiologist in Los Angeles for a check–up appointment. Therefore, the next API request should be:

Final Answer: API–Request: [HealthcareProviderAppointmentChecker(specialty='cardiologist', location='Los Angeles')]

Example 3:

question: {q3}

groundtruth: API–Request: [ToolSearcher(keywords='healthcare provider appointment scheduler')]

Inference: The user previously called the HealthcareProviderAppointmentChecker API and found three appointment times, which are '2034–04–18 14:30:00', '2034–04–19 11:00:00', and '2034–04–20 09:45:00'. The next step is to find the scheduler for the appointment. Since there is no available tool, the user needs to search for a tool that can schedule healthcare provider appointments. The ToolSearcher API can be used to search for relevant tools based on the keywords according to the description. The keywords should be 'healthcare provider appointment scheduler'. Therefore, the answer is:

Final Answer: API–Request: [ToolSearcher(keywords='healthcare provider appointment scheduler')]

We have a question and a groundtruth

question: {question}

groundtruth: {gold_label}

1. We wish you to answer the question. We will use your answer to train our model, thus you will answer and pretend as not

```
        ↪ knowing the gold_label.
2. You must answer the question (with inference
        ↪ process) directly without say anything
        ↪ else. Please not saying anything 'like
        ↪ sure I can help you with' or 'sure, i will
        ↪ not mention the gold label'
3. You will inference first then put the Final
        ↪ Answer ({gold_label}) at the end of the
        ↪ prediction like this

INFERENCE HERE
Final Answer: {gold_label}

"""
```

A.6 Prompt Used to Extract Labels from Predictions

The code below shows how we use Qwen-2.5-Instruct to extract the predicted labels from the predictions.

```
if 'arc_challenge' in task_name or 'mmlu' in
        ↪ task_name or 'agieval' in task_name:
            gold_label_type = 'A/B/C/D'
        elif 'piqa' in task_name or 'winogrande' in
            ↪ task_name:
            gold_label_type = '1/2'
        elif 'squad' in task_name:
            gold_label_type = 'text_span'
        elif 'gsm8k' in task_name or 'math' in
            ↪ task_name:
            gold_label_type = 'number'
        elif 'ecqa' in task_name:
            gold_label_type = '1/2/3/4/5'
        elif 'esnli' in task_name:
            gold_label_type = 'Entailment/Neutral/
                ↪ Contradiction'
        elif 'boolq' in task_name:
            gold_label_type = 'True/False'
        elif 'mmlu_pro' in task_name:
            gold_label_type = 'A/B/C/D/E/F/G/H/I/
                ↪ J'
        elif 'hellaswag' in task_name:
            gold_label_type = '1/2/3/4'
        elif 'drop' in task_name:
            gold_label_type = '
                ↪ number_or_text_span'
        elif 'api_bank' in task_name:
            gold_label_type = 'API-request'
        elif 'plan_bench' in task_name:
            gold_label_type = '[PLAN]
                ↪ SOME_PLAN_HERE[PLAN
                ↪ END]'
    else:
        a = 1

    for i in range(len(question_list)):
        question = \
f"""Given the prediction, what is the final answer
    ↪ by the prediction?

The prediction is "{predict_list[i]}"

Directly output {gold_label_type} without saying
    ↪ anything else.
"""
```

A.7 AI Assistant

We used GPT-4o as a writing assistant and programming aid for editing purposes.

A.8 Required Compute Resources

Each individual training run reported in this paper requires approximately 5–48 GPU hours when using a 40GB A100 GPU. We do not recommend you to reproduce every training run, as there are too many experiments. Instead, we strongly recommend directly using the reported training outcomes from each table as the final results. You can then compute your ranking metrics to evaluate how well your metric aligns with the training outcomes. Calculating metrics such as perplexity on a small subset of all of the dataset takes only about 2 hours on a single 40GB A100 GPU.

A.9 License of the Dtasets

All dataset we use are publicly available dataset for research purpose. API-BANK (Lv 3 problems) (Li et al., 2023): CC-BY-SA GSM8K (Cobbe et al., 2021): MIT license PIQA (Bisk et al., 2020): unkown BoolQ (Clark et al., 2019):CC BY-SA 3.0 MBPP (Austin et al., 2021):CC BY 4.0 DROP (Dua et al., 2019): CC BY-SA 4.0 ARC-Challenge (Clark et al., 2018):CC BY-SA 4.0 PlanBench (Valmeekam et al., 2023): MIT license MATH (Algebra) and MATH (Geometry) (Hendrycks et al., 2021): MIT license SQuAD (Rajpurkar et al., 2016):SA 4.0 license MMLU(Hendrycks et al., 2020): MIT license WinoGrande (Sakaguchi et al., 2021): Apache-2.0 license Hellaswag (Zellers et al., 2019): MIT license ECQA (Aggarwal et al., 2021): Apache-2.0 license MMLU_PRO (Wang et al., 2024): Apache-

Method	Model Type	training task	GSM8K	Math Algebra	ECQA	SQUAD	DROP	WINOGRANDE
Gold Label	Mistral	ECQA	0.383	0.181	0.722	0.251	0.084	0.562
GPT-4o Answer Directly			0.484	0.218	0.707	0.175	0.016	0.638
Gold Label	Mistral	SQUAD	0.082	0.0931	0.633	0.74	0.208	0.566
GPT-4o Answer Directly			0.512	0.234	0.594	0.748	0.268	0.628
Gold Label	Mistral	DROP	0.076	0.097	0.621	0.561	0.628	0.578
GPT-4o Answer Directly			0.542	0.241	0.602	0.546	0.736	0.638
Gold Label	Mistral	WINOGRANDE	0.381	0.172	0.625	0.166	0.042	0.742
GPT-4o Answer Directly			0.477	0.219	0.569	0.106	0.016	0.713
Gold Label	LLAMA3	ECQA	0.798	0.416	0.734	0.193	0.1	0.637
GPT-4o Answer Directly			0.778	0.469	0.723	0.389	0.284	0.638
Gold Label	LLAMA3	SQUAD	0.584	0.366	0.712	0.758	0.49	0.639
GPT-4o Answer Directly			0.791	0.457	0.726	0.759	0.368	0.651
Gold Label	LLAMA3	DROP	0.144	0.169	0.674	0.574	0.738	0.582
GPT-4o Answer Directly			0.776	0.507	0.703	0.555	0.786	0.626
Gold Label	LLAMA3	WINOGRANDE	0.776	0.445	0.717	0.226	0.162	0.766
GPT-4o Answer Directly			0.775	0.485	0.721	0.305	0.238	0.695
Gold Label	Qwen	ECQA	0.914	0.903	0.814	0.662	0.008	0.675
GPT-4o Answer Directly			0.903	0.888	0.793	0.668	0.016	0.716
Gold Label	Qwen	SQUAD	0.899	0.892	0.784	0.768	0.056	0.693
GPT-4o Answer Directly			0.896	0.911	0.789	0.756	0.074	0.712
Gold Label	Qwen	DROP	0.788	0.904	0.799	0.701	0.664	0.711
GPT-4o Answer Directly			0.911	0.903	0.792	0.741	0.806	0.701
Gold Label	Qwen	WINOGRANDE	0.893	0.904	0.78	0.651	0.004	0.725
GPT-4o Answer Directly			0.902	0.896	0.798	0.68	0.022	0.721

Table 8: The training data size is 1000. This table compares the in-domain and cross-domain performance when training on gold-label vs. GPT-4 generated synthetic data. As can be seen from the table, the in-domain performance of the model is typically higher when training with gold-label data. However, the cross-domain performance when training on GPT-4 generated data is significantly higher than when training with only gold-label data. The grey area represents the in-domain performance.

Dataset	**Method**	**Accuracy and N train**	**Mistral**	**Llama3-8B-Chat**
MATH Algebra	GPT4 preview	82.5%, 1000	0.301	0.504
	GPT4 only correct	100%, 825	0.293	0.501
	GPT4 only correct + rewritten ground truth	100%, 1000	0.293	0.500
MATH Algebra	Claude	90.1%, 1000	0.265	0.508
	Claude only correct	100%, 901	0.277	0.487
	Claude only correct + rewritten ground truth	100%, 1000	0.286	0.492
MATH Algebra	Mini GPT4	91.6% , 1000	0.313	0.523
	Mini GPT4 only correct	100%, 916	0.311	o.523
	Mini GPT4 only correct + rewritten ground truth	100%, 1000	0.326	0.539
GSM8K	GPT4 preview	92.1%, 1000	0.597	0.799
	GPT4 only correct	100%, 921	0.587	0.791
	GPT4 only correct + rewritten ground truth	100%, 1000	0.588	0.808
GSM8K	Claude	95.6%, 1000	0.578	0.796
	Claude only correct	100%, 956	0.580	0.797
	Claude only correct + rewritten ground truth	100%, 1000	0.588	0.798
GSM8K	Mini GPT4	89.8% , 1000	0.623	0.795
	Mini GPT4 only correct	100%, 898	0.606	0.793
	Mini GPT4 only correct + rewritten ground truth	100%, 1000	0.607	0.790

Table 9: The table shows that the accuracy of the generated data has a marginal effect on the training outcome. In this table, we use the API with different math abilities. The rank of their math problem-solving abilities is: Claude >MiniGPT-4 >GPT-4 preview. GPT-4 preview represents the data generated using the GPT-4 preview model, rather than the GPT-4o model.

2.0 license AGIEval(Zhong et al., 2023): MIT license

Data Generation Strategy	Model Type	gsm8k	math algebra	math geometry	ecqa	boolq	winogrande	piqa	agieval	squad	arc challenge	drop	mbpp	api bank	hellaswag	mmlu pro law	mmlu moral scenarios
gold label	mistral				0.717	0.997	0.736	0.854	0.44	0.741	0.747	0.645		0.452	0.772	0.263	0.679
groundtruth		0.442	0.194	0.125	0.684								0.325				
gpt4		0.62	0.324	0.146	0.703	0.87	0.717	0.864	0.41	0.732	0.631	0.723	0.362	0.515	0.659	0.238	0.691
claude		0.582	0.278	0.136	0.735	0.885	0.724	0.848	0.445	0.736	0.753	0.729	0.379	0.579	0.553	0.248	0.751
mini gpt4		0.619	0.306	0.151	0.708	0.882	0.695	0.868	0.427	0.732	0.772	0.735	0.348	0.43	0.663	0.205	0.659
step by step		0.626	0.314	0.137	0.706	0.874	0.693	0.862	0.445	0.749	0.71	0.696	0.333	0.377	0.644	0.249	0.714
openai human written examples		0.621	0.303	0.163	0.708	0.891	0.721	0.859	0.413	0.76	0.692	0.741	0.345	0.411	0.674	0.233	0.71
gpt4 style in context examples		0.61	0.254	0.158	0.726	0.884	0.727	0.868	0.44	0.761	0.697	0.735	0.378	0.416	0.672	0.225	0.728
rewrite groundtruth in own words		0.502	0.238	0.127	0.703								0.306				
gold label	llama 3 instruct				0.737	0.979	0.761	0.852	0.432	0.756	0.766	0.742		0.507	0.772	0.332	0.639
groundtruth		0.678	0.404	0.239	0.701								0.445				
gpt4		0.816	0.559	0.301	0.74	0.87	0.697	0.866	0.448	0.759	0.806	0.793	0.482	0.477	0.712	0.247	0.659
claude		0.803	0.5	0.254	0.756	0.865	0.72	0.86	0.445	0.765	0.801	0.757	0.471	0.547	0.709	0.259	0.737
mini gpt4		0.805	0.551	0.28	0.721	0.864	0.677	0.868	0.437	0.747	0.816	0.783	0.491	0.384	0.719	0.225	0.645
step by step		0.797	0.562	0.26	0.731	0.869	0.72	0.853	0.433	0.779	0.792	0.78	0.455	0.227	0.71	0.242	0.684
openai human written examples		0.81	0.547	0.283	0.735	0.893	0.717	0.867	0.44	0.766	0.804	0.807	0.477	0.347	0.706	0.229	0.667
gpt4 style in context examples		0.796	0.494	0.285	0.736	0.885	0.719	0.87	0.447	0.752	0.811	0.794	0.47	0.368	0.721	0.259	0.681
rewrite groundtruth in own words		0.742	0.444	0.241	0.727								0.437				
gold label	qwen				0.816	0.892	0.732	0.867	0.48	0.77	0.855	0.663		0.515	0.74	0.303	0.605
groundtruth		0.899	0.894	0.667	0.793								0.59				
gpt4		0.897	0.916	0.679	0.794	0.858	0.709	0.878	0.552	0.76	0.886	0.798	0.591	0.436	0.722	0.3	0.656
claude		0.895	0.904	0.648	0.788	0.862	0.72	0.88	0.553	0.766	0.874	0.793	0.607	0.462	0.72	0.309	0.66
mini gpt4		0.904	0.904	0.654	0.787	0.87	0.712	0.882	0.555	0.763	0.891	0.821	0.642	0.379	0.701	0.308	0.664
step by step		0.899	0.907	0.642	0.792	0.859	0.716	0.88	0.548	0.767	0.881	0.806	0.623	0.417	0.713	0.287	0.662
openai human written examples		0.905	0.909	0.647	0.787	0.871	0.697	0.883	0.547	0.794	0.883	0.82	0.628	0.458	0.724	0.283	0.608
gpt4 style in context examples		0.899	0.903	0.657	0.803	0.88	0.731	0.878	0.57	0.785	0.868	0.809	0.631	0.309	0.742	0.3	0.642
rewrite groundtruth in own words		0.902	0.904	0.654	0.787								0.589				

Table 10: average of seed 0,1,2 train datasize 1000 lr 2e-05 epoch num 20

Data Generation Strategy	Model Type	gsm8k	math algebra	math geometry	ecqa	boolq	winogrande	piqa	agieval	squad	arc challenge	drop	mbpp	api bank	hellaswag	mmlu pro law	mmlu moral scenarios
gold label	mistral				0.722	0.996	0.742	0.852	0.440	0.748	0.759	0.628		0.465	0.771	0.252	0.650
groundtruth		0.440	0.201	0.110	0.672								0.370				
gpt4		0.625	0.319	0.177	0.700	0.867	0.713	0.869	0.400	0.732	0.611	0.746	0.347	0.510	0.654	0.229	0.713
claude		0.583	0.279	0.160	0.720	0.886	0.709	0.849	0.425	0.728	0.732	0.726	0.403	0.584	0.549	0.219	0.760
mini gpt4		0.627	0.291	0.148	0.710	0.873	0.688	0.877	0.420	0.740	0.775	0.726	0.363	0.433	0.663	0.183	0.643
step by step		0.639	0.323	0.127	0.705	0.885	0.687	0.861	0.445	0.752	0.708	0.676	0.340	0.478	0.639	0.196	0.723
openai human written examples		0.604	0.306	0.160	0.709	0.897	0.718	0.869	0.420	0.756	0.685	0.742	0.350	0.400	0.664	0.196	0.717
gpt4 style in context examples		0.619	0.231	0.169	0.725	0.887	0.732	0.879	0.430	0.764	0.678	0.732	0.373	0.433	0.687	0.223	0.710
rewrite groundtruth in own words		0.511	0.231	0.127	0.709								0.323				
gold label	llama 3 instruct				0.734	0.978	0.766	0.855	0.435	0.761	0.764	0.738		0.502	0.777	0.312	0.630
groundtruth		0.681	0.396	0.215	0.691								0.450				
gpt4		0.814	0.562	0.278	0.723	0.880	0.695	0.865	0.435	0.752	0.801	0.796	0.480	0.494	0.722	0.276	0.677
claude		0.816	0.493	0.253	0.748	0.879	0.728	0.864	0.455	0.763	0.808	0.746	0.500	0.547	0.710	0.286	0.757
mini gpt4		0.795	0.557	0.278	0.725	0.867	0.702	0.863	0.450	0.739	0.826	0.730	0.500	0.384	0.703	0.223	0.670
step by step		0.798	0.564	0.308	0.728	0.874	0.718	0.866	0.460	0.783	0.792	0.780	0.450	0.216	0.715	0.229	0.657
openai human written examples		0.811	0.547	0.266	0.736	0.891	0.719	0.864	0.450	0.770	0.809	0.808	0.457	0.355	0.699	0.269	0.640
gpt4 style in context examples		0.792	0.515	0.274	0.742	0.875	0.717	0.854	0.460	0.755	0.809	0.798	0.483	0.273	0.718	0.219	0.683
rewrite groundtruth in own words		0.729	0.443	0.241	0.715								0.417				
gold label	qwen				0.814	0.880	0.725	0.868	0.500	0.769	0.856	0.652		0.518	0.747	0.296	0.590
groundtruth		0.906	0.898	0.675	0.784								0.610				
gpt4		0.889	0.916	0.658	0.793	0.865	0.721	0.879	0.545	0.762	0.890	0.794	0.607	0.433	0.706	0.276	0.633
claude		0.884	0.906	0.662	0.796	0.873	0.716	0.885	0.550	0.767	0.867	0.798	0.600	0.457	0.717	0.322	0.667
mini gpt4		0.905	0.904	0.654	0.782	0.865	0.704	0.881	0.535	0.760	0.891	0.818	0.633	0.396	0.704	0.299	0.657
step by step		0.899	0.908	0.624	0.795	0.846	0.703	0.874	0.545	0.752	0.882	0.766	0.630	0.412	0.717	0.276	0.653
openai human written examples		0.907	0.910	0.658	0.790	0.876	0.699	0.884	0.540	0.808	0.881	0.816	0.617	0.445	0.731	0.286	0.583
gpt4 style in context examples		0.896	0.902	0.654	0.799	0.883	0.734	0.871	0.540	0.782	0.863	0.800	0.607	0.339	0.742	0.302	0.653
rewrite groundtruth in own words		0.911	0.899	0.654	0.791								0.587				

Table 11: seed 0 train datasize 1000 lr 2e-05 epoch num 20

Data Generation Strategy	Model Type	gsm8k	math algebra	math geometry	ecqa	boolq	winogrande	piqa	agieval	squad	arc challenge	drop	mbpp	api bank	hellaswag	mmlu pro law	mmlu moral scenarios
gold label	mistral				0.714	0.997	0.733	0.855		0.738	0.741	0.654		0.445	0.772	0.269	0.693
groundtruth		0.443	0.191	0.131	0.690								0.303				
gpt4		0.617	0.327	0.148	0.704	0.872	0.719	0.861	0.415	0.732	0.641	0.712	0.370	0.518	0.662	0.243	0.680
claude		0.581	0.277	0.143	0.742	0.885	0.731	0.847	0.455	0.740	0.764	0.730	0.367	0.576	0.555	0.262	0.747
mini gpt4		0.615	0.314	0.148	0.707	0.886	0.698	0.863	0.430	0.728	0.771	0.740	0.340	0.429	0.663	0.216	0.667
step by step		0.619	0.309	0.131	0.707	0.868	0.696	0.862	0.445	0.748	0.711	0.706	0.330	0.327	0.646	0.276	0.710
openai human written examples		0.630	0.302	0.165	0.707	0.888	0.723	0.854	0.410	0.762	0.695	0.740	0.343	0.416	0.679	0.252	0.707
gpt4 style in context examples		0.605	0.265	0.152	0.726	0.882	0.724	0.862	0.445	0.760	0.706	0.736	0.380	0.408	0.665	0.226	0.737
rewrite groundtruth in own words		0.497	0.241	0.139	0.700								0.297				
gold label	llama 3 instruct				0.738	0.979	0.759	0.850	0.430	0.754	0.767	0.744		0.510	0.769	0.342	0.643
groundtruth		0.677	0.408	0.241	0.706								0.443				
gpt4		0.817	0.557	0.312	0.748	0.865	0.698	0.866	0.455	0.762	0.808	0.792	0.483	0.469	0.707	0.233	0.650
claude		0.796	0.504	0.253	0.760	0.858	0.716	0.858	0.440	0.766	0.797	0.762	0.457	0.547	0.709	0.246	0.727
mini gpt4		0.810	0.548	0.274	0.719	0.863	0.664	0.871	0.430	0.751	0.811	0.810	0.487	0.384	0.727	0.226	0.633
step by step		0.796	0.561	0.266	0.733	0.867	0.721	0.846	0.420	0.777	0.792	0.780	0.457	0.233	0.708	0.249	0.697
openai human written examples		0.809	0.547	0.291	0.735	0.894	0.716	0.868	0.435	0.764	0.801	0.806	0.487	0.343	0.709	0.209	0.680
gpt4 style in context examples		0.798	0.484	0.291	0.733	0.890	0.720	0.878	0.440	0.751	0.812	0.792	0.463	0.416	0.723	0.279	0.680
rewrite groundtruth in own words		0.749	0.445	0.253	0.733								0.447				
gold label	qwen				0.817	0.898	0.735	0.867	0.470	0.771	0.854	0.668		0.514	0.737	0.306	0.613
groundtruth		0.896	0.892	0.658	0.798								0.580				
gpt4		0.901	0.904	0.692	0.794	0.855	0.703	0.878	0.555	0.759	0.884	0.800	0.583	0.437	0.730	0.312	0.667
claude		0.901	0.903	0.654	0.784	0.857	0.722	0.878	0.555	0.765	0.877	0.790	0.610	0.465	0.721	0.302	0.657
mini gpt4		0.903	0.904	0.662	0.789	0.872	0.716	0.882	0.565	0.765	0.891	0.822	0.647	0.371	0.700	0.312	0.667
step by step		0.899	0.907	0.646	0.790	0.866	0.723	0.883	0.550	0.775	0.881	0.826	0.620	0.420	0.711	0.292	0.667
openai human written examples		0.904	0.908	0.641	0.786	0.868	0.696	0.883	0.550	0.787	0.884	0.822	0.633	0.465	0.720	0.282	0.620
gpt4 style in context examples		0.900	0.903	0.658	0.805	0.878	0.730	0.882	0.585	0.787	0.870	0.814	0.643	0.294	0.742	0.299	0.637
rewrite groundtruth in own words		0.897	0.907	0.692	0.785								0.590				

Table 12: seed 1 train datasize 1000 lr 2e-05 epoch num 20

Data Generation Strategy	Model Type	gsm8k	math algebra	math geometry	ecqa	boolq	winogrande	piqa	agieval	squad	arc challenge	drop	mbpp	api bank	hellaswag	mmlu pro law	mmlu moral scenarios
gold label	mistral				0.681	0.996	0.743	0.838	0.450	0.741	0.734	0.656		0.449	0.776	0.269	0.663
groundtruth		0.441	0.211	0.101	0.679								0.350				
gpt4		0.617	0.315	0.169	0.708	0.868	0.700	0.870	0.415	0.739	0.661	0.720	0.343	0.482	0.641	0.276	0.703
claude		0.612	0.277	0.148	0.742	0.883	0.716	0.856	0.410	0.744	0.743	0.726	0.367	0.445	0.570	0.246	0.713
mini gpt4		0.622	0.320	0.177	0.703	0.865	0.688	0.855	0.435	0.740	0.768	0.708	0.353	0.429	0.670	0.219	0.697
step by step		0.622	0.322	0.139	0.709	0.866	0.697	0.843	0.430	0.763	0.700	0.714	0.360	0.298	0.661	0.219	0.720
openai human written examples		0.614	0.323	0.156	0.701	0.900	0.718	0.855	0.405	0.754	0.663	0.748	0.363	0.408	0.679	0.246	0.720
gpt4 style in context examples		0.606	0.251	0.165	0.712	0.884	0.724	0.860	0.420	0.771	0.711	0.748	0.373	0.449	0.673	0.266	0.737
rewrite groundtruth in own words		0.506	0.222	0.135	0.703								0.327				
gold label	llama 3 instruct				0.735	0.980	0.760	0.865	0.445	0.757	0.762	0.740		0.465	0.784	0.329	0.663
groundtruth		0.696	0.415	0.228	0.690								0.413				
gpt4		0.806	0.553	0.278	0.733	0.864	0.697	0.865	0.450	0.742	0.824	0.748	0.487	0.445	0.725	0.233	0.683
claude		0.789	0.489	0.257	0.734	0.866	0.685	0.846	0.450	0.759	0.800	0.770	0.507	0.547	0.716	0.243	0.743
mini gpt4		0.795	0.536	0.287	0.733	0.866	0.690	0.869	0.450	0.754	0.796	0.686	0.467	0.367	0.709	0.246	0.640
step by step		0.800	0.551	0.245	0.719	0.884	0.707	0.865	0.460	0.767	0.782	0.792	0.467	0.245	0.697	0.269	0.653
openai human written examples		0.796	0.529	0.287	0.736	0.884	0.714	0.863	0.450	0.757	0.808	0.800	0.460	0.367	0.689	0.223	0.680
gpt4 style in context examples		0.800	0.500	0.283	0.729	0.876	0.709	0.856	0.440	0.767	0.809	0.816	0.480	0.433	0.708	0.252	0.683
rewrite groundtruth in own words		0.754	0.431	0.291	0.715								0.457				
gold label	qwen				0.818	0.887	0.724	0.867	0.495	0.774	0.861	0.652		0.539	0.740	0.302	0.590
groundtruth		0.901	0.910	0.675	0.758								0.623				
gpt4		0.897	0.892	0.654	0.791	0.858	0.710	0.883	0.540	0.777	0.882	0.788	0.603	0.433	0.703	0.306	0.607
claude		0.881	0.916	0.641	0.785	0.859	0.735	0.877	0.540	0.762	0.872	0.798	0.597	0.461	0.732	0.292	0.690
mini gpt4		0.902	0.904	0.658	0.778	0.875	0.711	0.880	0.555	0.760	0.890	0.798	0.613	0.396	0.696	0.309	0.677
step by step		0.886	0.907	0.679	0.770	0.859	0.715	0.869	0.560	0.767	0.867	0.792	0.597	0.404	0.711	0.332	0.677
openai human written examples		0.900	0.887	0.646	0.794	0.881	0.707	0.872	0.540	0.802	0.895	0.804	0.603	0.449	0.724	0.306	0.640
gpt4 style in context examples		0.911	0.908	0.650	0.792	0.875	0.728	0.891	0.535	0.790	0.866	0.824	0.577	0.318	0.734	0.316	0.643
rewrite groundtruth in own words		0.905	0.899	0.637	0.799								0.600				

Table 13: seed 2 train datasize 1000 lr 2e-05 epoch num 20

Data Generation Strategy	Model Type	gsm8k	math algebra	math geometry	ecqa	boolq	winogrande	piqa	agieval	squad	arc challenge	drop	mbpp	api bank	hellaswag	mmlu pro law	mmlu moral scenarios
gold label	mistral				0.627	0.869	0.608	0.814	0.430	0.582	0.704	0.482		0.220	0.625	0.153	0.420
groundtruth		0.420	0.205	0.101	0.591								0.267				
gpt4		0.513	0.231	0.101	0.596	0.837	0.636	0.790	0.345	0.333	0.624	0.244	0.317	0.249	0.269	0.166	0.380
claude		0.505	0.215	0.110	0.634	0.837	0.627	0.804	0.400	0.290	0.630	0.250	0.340	0.257	0.284	0.179	0.413
mini gpt4		0.511	0.223	0.097	0.619	0.845	0.644	0.782	0.360	0.404	0.633	0.210	0.337	0.253	0.223	0.183	0.343
step by step		0.494	0.247	0.080	0.593	0.845	0.636	0.765	0.355	0.314	0.618	0.092	0.317	0.265	0.254	0.183	0.403
openai human written examples		0.504	0.230	0.118	0.611	0.853	0.639	0.811	0.355	0.467	0.578	0.280	0.317	0.257	0.316	0.166	0.517
gpt4 style in context examples		0.500	0.245	0.114	0.560	0.845	0.649	0.789	0.340	0.312	0.611	0.124	0.337	0.208	0.295	0.183	0.423
rewrite groundtruth in own words		0.450	0.214	0.110	0.603								0.317				
gold label	llama 3 instruct				0.710	0.852	0.636	0.789	0.395	0.680	0.764	0.620		0.082	0.610	0.196	0.200
groundtruth		0.794	0.460	0.249	0.691								0.407				
gpt4		0.791	0.491	0.266	0.686	0.802	0.634	0.801	0.430	0.504	0.760	0.410	0.480	0.082	0.592	0.223	0.387
claude		0.804	0.492	0.266	0.699	0.806	0.640	0.821	0.450	0.495	0.739	0.420	0.483	0.082	0.608	0.233	0.430
mini gpt4		0.797	0.477	0.257	0.710	0.800	0.621	0.823	0.425	0.509	0.751	0.400	0.497	0.086	0.589	0.229	0.373
step by step		0.808	0.496	0.219	0.702	0.818	0.626	0.799	0.435	0.565	0.743	0.488	0.477	0.110	0.608	0.199	0.407
openai human written examples		0.809	0.472	0.257	0.720	0.810	0.630	0.815	0.440	0.564	0.747	0.414	0.500	0.078	0.600	0.236	0.387
gpt4 style in context examples		0.800	0.434	0.266	0.695	0.793	0.638	0.808	0.455	0.429	0.762	0.344	0.497	0.090	0.582	0.229	0.407
rewrite groundtruth in own words		0.813	0.480	0.262	0.718								0.447				
gold label	qwen				0.791	0.843	0.677	0.875	0.465	0.703	0.877	0.334		0.220	0.702	0.306	0.387
groundtruth		0.913	0.918	0.679	0.792								0.603				
gpt4		0.908	0.898	0.692	0.802	0.831	0.711	0.863	0.560	0.661	0.891	0.092	0.637	0.237	0.697	0.326	0.580
claude		0.911	0.912	0.679	0.788	0.837	0.718	0.876	0.550	0.652	0.894	0.114	0.640	0.237	0.691	0.282	0.577
mini gpt4		0.902	0.914	0.667	0.791	0.852	0.720	0.884	0.550	0.660	0.891	0.068	0.600	0.237	0.702	0.296	0.583
step by step		0.909	0.919	0.599	0.802	0.848	0.708	0.870	0.545	0.682	0.879	0.062	0.617	0.224	0.690	0.309	0.563
openai human written examples		0.900	0.918	0.671	0.789	0.842	0.718	0.864	0.545	0.681	0.887	0.090	0.597	0.237	0.706	0.322	0.563
gpt4 style in context examples		0.918	0.914	0.679	0.798	0.836	0.710	0.865	0.535	0.678	0.888	0.050	0.627	0.196	0.696	0.326	0.567
rewrite groundtruth in own words		0.910	0.916	0.667	0.782								0.590				

Table 14: seed 0 train datasize 100 lr 2e-05 epoch num 20

Data Generation Strategy	Model Type	gsm8k	math algebra	math geometry	ecqa	boolq	winogrande	piqa	agieval	squad	arc challenge	drop	mbpp	api bank	hellaswag	mmlu pro law	mmlu moral scenarios
gold label	mistral				0.681	0.870	0.694	0.830	0.420	0.730	0.726	0.620		0.486	0.737	0.236	0.677
groundtruth		0.409	0.186	0.093	0.638								0.293				
gpt4		0.586	0.270	0.152	0.672	0.864	0.686	0.821	0.455	0.649	0.736	0.670	0.340	0.404	0.623	0.233	0.687
claude		0.554	0.237	0.122	0.663	0.858	0.701	0.855	0.405	0.690	0.760	0.662	0.360	0.400	0.619	0.243	0.710
mini gpt4		0.514	0.266	0.152	0.705	0.850	0.674	0.847	0.425	0.670	0.739	0.666	0.357	0.359	0.651	0.233	0.623
step by step		0.575	0.235	0.131	0.662	0.853	0.667	0.842	0.415	0.691	0.746	0.646	0.327	0.286	0.575	0.233	0.593
openai human written examples		0.536	0.278	0.156	0.674	0.874	0.665	0.850	0.435	0.700	0.764	0.698	0.340	0.302	0.628	0.229	0.677
gpt4 style in context examples		0.548	0.222	0.156	0.658	0.879	0.681	0.864	0.430	0.676	0.741	0.650	0.333	0.343	0.628	0.203	0.687
rewrite groundtruth in own words		0.443	0.202	0.101									0.330				
gold label	llama 3 instruct				0.705	0.866	0.675	0.847	0.430	0.727	0.773	0.684		0.494	0.682	0.299	0.633
groundtruth		0.683	0.404	0.211	0.679								0.430				
gpt4		0.798	0.529	0.257	0.731	0.864	0.679	0.845	0.440	0.729	0.815	0.734	0.470	0.424	0.711	0.246	0.683
claude		0.805	0.495	0.224	0.712	0.834	0.694	0.857	0.420	0.744	0.789	0.742	0.467		0.677	0.226	0.693
mini gpt4		0.807	0.504	0.278	0.719	0.852	0.674	0.858	0.445	0.746	0.795	0.744	0.473	0.335	0.676	0.266	0.630
step by step		0.779	0.528	0.198	0.690	0.874	0.683	0.863	0.435	0.736	0.797	0.708	0.457	0.253	0.688	0.233	0.620
openai human written examples		0.772	0.483	0.249	0.712	0.873	0.678	0.853	0.405	0.726	0.789	0.772	0.473	0.302	0.674	0.262	0.640
gpt4 style in context examples		0.794	0.488	0.283	0.712	0.861	0.690	0.859	0.440	0.729	0.770	0.754	0.473	0.380	0.702	0.259	0.693
rewrite groundtruth in own words		0.693	0.415	0.232									0.430				
gold label	qwen				0.820	0.883	0.704	0.858	0.480	0.747	0.849	0.642		0.457	0.725	0.339	0.563
groundtruth		0.867	0.896	0.637	0.823								0.523				
gpt4		0.897	0.890	0.620	0.787	0.859	0.709	0.881	0.545	0.743	0.882	0.808	0.617	0.388	0.687	0.362	0.690
claude		0.882	0.890	0.616	0.790	0.869	0.738	0.867	0.555	0.766	0.871	0.810	0.603	0.527	0.702	0.316	0.750
mini gpt4		0.889	0.912	0.624	0.794	0.867	0.719	0.887	0.530	0.750	0.891	0.772	0.580	0.429	0.707	0.289	0.640
step by step		0.902	0.899	0.586	0.788	0.868	0.731	0.878	0.545	0.737	0.881	0.788	0.630	0.339	0.715	0.309	0.677
openai human written examples		0.892	0.899	0.616	0.783	0.874	0.727	0.883	0.565	0.776	0.875	0.824	0.590	0.392	0.694	0.233	0.643
gpt4 style in context examples		0.896	0.899	0.637	0.782	0.864	0.720	0.881	0.550	0.764	0.868	0.832	0.620	0.335	0.752	0.302	0.677
rewrite groundtruth in own words		0.899	0.892	0.646									0.583				

Table 15: seed 0 train datasize 100 lr 0.0002 epoch num 40

Benchmark Name	Data Name	Chosen/Not Chosen	Why not chosen
Mistral 7B	Winogrande	✓	
	PIQA	✓	
	GSM8K	✓	
	MATH	✓	
	MBPP	✓	
	MMLU	✓	
	AGIEVAL	✓	
	ARC Challenge	✓	
	BoolQ	✓	
	Hellaswag	✓	
	CommonsenseQA	×	not a reasoning task
	BBH	×	In github, it says this dataset can never used in training.
	SIQA	×	not a reasoning task
	OpenbookQA	×	not a reasoning task
	ARC Easy	×	We already choose ARC Challenge
	NaturalQuestions	×	It evaluates world knowledge instead of reasoning ability
	TriviaQA	×	It evaluates world knowledge instead of reasoning ability
	QuAC	×	this is a multiturn, muti context qa dataset. evaluation is too hard
Llama 3	MMLU	✓	
	MMLU_Pro	✓	
	GSM8K	✓	
	MATH	✓	
	AGIEVAL	✓	
	ARC CHALLENGE	✓	
	DROP	✓	
	API-BANK	✓	
	IFEval	×	less than 650 data
	HumanEval+	×	less than 650 data
	BFCL	×	(subcategory) less than 650 data
	Nexus	×	Unable to find the dataset
	GPQA	×	less than 650 data
	HumanEval	×	less than 650 data
	ZeroSCROLLS/QuALITY	×	This dataset evaluating model's long context QA ability. The input is too long thus is hard to train.
	InfiniteBench/En.MC	×	This dataset evaluating model's long context QA ability. The input is too long thus is hard to train.
	NIH/Multi-needle	×	Long context QA task. The input is too long thus is hard to train. Llama already achieves 98.8% accuracy with zero-shot setting.
Qwen2.5	MMLU	✓	
	MMLU Pro	✓	
	MBPP	✓	
	ARC CHALLENGE	✓	
	GSM8K	✓	
	MATH	✓	
	WindoGrande	✓	
	HellaSwag	✓	
	MMLU stem	×	(subcategory) less than 650 data
	TruthfulQA	×	not reasoning task
	GPQA	×	less than 650 data
	TheoremQA	×	the data set is tooooo challenging for GPT-4o. it does not have the ability to be a teacher for this task.
	HumanEval	×	less than 650 data
	HumanEval+	×	less than 650 data
	MMLU redux	×	(subcategory) less than 650 data
	BBH	×	In github, it says this dataset can never used in training.
	MBPP+	×	less than 650 data
	MultiPL-E	×	(subcategory) less than 650 data

Table 16: This table explains which data from the Mistral, LLaMA3, and Qwen benchmarks were chosen and why some data were not selected. Multi-lingual dataset is not listed in this Table since our experiment only covers English-only datasets. API-BANK is in Table 16 from Llama 3 technical report.

Data Generation Strategy	Model Type	plan bench generation	plan bench optimality	plan bench generalization	plan bench replaning	plan bench reuse	plan bench verification
gold label	mistral	0.607	0.448	0.807	0.815	0.945	0.318
groundtruth							
gpt4		0.458	0.37	0.215	0.2	0.61	0.518
claude		0.505	0.45	0.362	0.445	0.755	0.586
mini gpt4		0.445	0.38	0.282	0.235	0.51	0.5
step by step		0.448	0.253	0.253	0.177	0.63	0.492
openai human written examples		0.485	0.43	0.302	0.28	0.73	0.474
gpt4 style in context examples		0.37	0.232	0.43	0.207	0.665	0.56
gold label	llama 3 instruct	0.478	0.38	0.853	0.87	0.87	0.482
groundtruth							
gpt4		0.463	0.36	0.265	0.225	0.72	0.49
claude		0.545	0.485	0.407	0.425	0.637	0.532
mini gpt4		0.415	0.375	0.25	0.215	0.5	0.674
step by step		0.475	0.39	0.287	0.232	0.732	0.564
openai human written examples		0.458	0.372	0.307	0.25	0.603	0.516
gpt4 style in context examples		0.45	0.435	0.438	0.223	0.75	0.468
gold label	qwen	0.32	0.285	0.38	0.732	0.845	0.516
groundtruth							
gpt4		0.237	0.253	0.145	0.2	0.515	0.538
claude		0.325	0.265	0.277	0.46	0.393	0.534
mini gpt4		0.253	0.255	0.168	0.188	0.42	0.55
step by step		0.265	0.182	0.135	0.17	0.577	0.536
openai human written examples		0.24	0.19	0.222	0.155	0.47	0.52
gpt4 style in context examples		0.302	0.268	0.338	0.2	0.597	0.534

Table 17: seed average of seed_0,1 train datasize 1000 lr 2e-05 epoch num 40