DICE: Structured Reasoning in LLMs through SLM-Guided Chain-of-Thought Correction

Yiqi Li[1], Yusheng Liao[1,2], Zhe Chen[1,2], Yanfeng Wang[1,2], Yu Wang[1,2]*

[1]Shanghai Jiao Tong University,
[2]Shanghai Artificial Intelligence Laboratory
{17-adamant,liao20160907,chenzhe2018,wangyanfeng,yuwangsjtu}@sjtu.edu.cn

Abstract

When performing reasoning tasks with user-specific requirements, such as strict output formats, large language models (LLMs) often prioritize reasoning over adherence to detailed instructions. Fine-tuning LLMs on supervised datasets to address this is impractical due to high computational costs and limited parameter access. To tackle this, we propose DICE, a lightweight framework that guides small language models (SLMs) to refine LLMs' outputs through chain-of-thought (CoT) correction. DICE decouples the process by first prompting LLMs to generate natural language responses, then using trained SLMs to analyze and refine these outputs to meet structured output specifications. This framework preserves LLMs' broad knowledge and reasoning capabilities while ensuring the outputs conform to user demands. Specifically, DICE first constructs structured CoT adaptation datasets via a two-stage method and subsequently applies a dual-tuning strategy to fine-tune SLMs for generating structured outputs in an analyze-then-answer pattern.[1] Experiments demonstrate that DICE improves the average format accuracy and content correctness of LLM outputs by 35.4% and 29.4%, respectively, achieving state-of-the-art (SOTA) performance over other competitive baselines.

1 Introduction

Large language models (LLMs) have demonstrated significant advancements across diverse natural language processing (NLP) tasks, exhibiting exceptional capabilities in language comprehension and reasoning (Guo et al., 2025; Yang et al., 2024; Grattafiori et al., 2024; Team et al., 2024; Chen et al., 2025b; Hurst et al., 2024). Their ability to follow general instructions is crucial in practical scenarios, such as complex decision-making, scientific research, and automated problem solving (Ouyang et al., 2022; Qin et al., 2024; Zhao et al., 2025). However, this instruction-following ability tends to degrade when LLMs are applied to challenging reasoning tasks (Tam et al., 2024; Shorten et al., 2024; Chen et al., 2025a), which restricts the broader application in reasoning-intensive tasks.

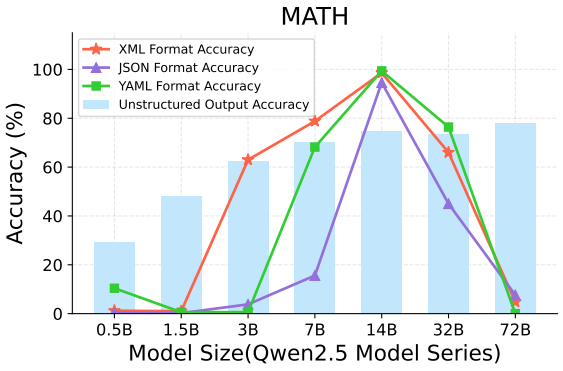


Figure 1: **Structured format accuracy and unstructured output accuracy across model sizes on *MATH*.** The models are required to generate structured output given 2-shot prompts. The bars represent content accuracy of unstructured natural language outputs, and the lines denote the format accuracy of structured outputs. More details about formats are in Appendix B.

While scaling up LLMs can enhance their reasoning capacity (Kaplan et al., 2020; Zhong et al., 2021; Naveed et al., 2024), we observe a counter-intuitive trade-off: larger models sometimes exhibit weaker adherence to user-specific instructions compared to smaller counterparts, even when their underlying reasoning is correct. Our preliminary experiments (Figure 1) provide empirical evidence of this trade-off, focusing on user instructions related to specific output formats. It is observed that format accuracy peaks at mid-scale models while declining in larger models, despite their superior reasoning performance. Specifically, large-scale

*Corresponding author
[1]The datasets and code will be available at https://github.com/1717Li/DICE

Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, pages 6951–6966
November 4-9, 2025

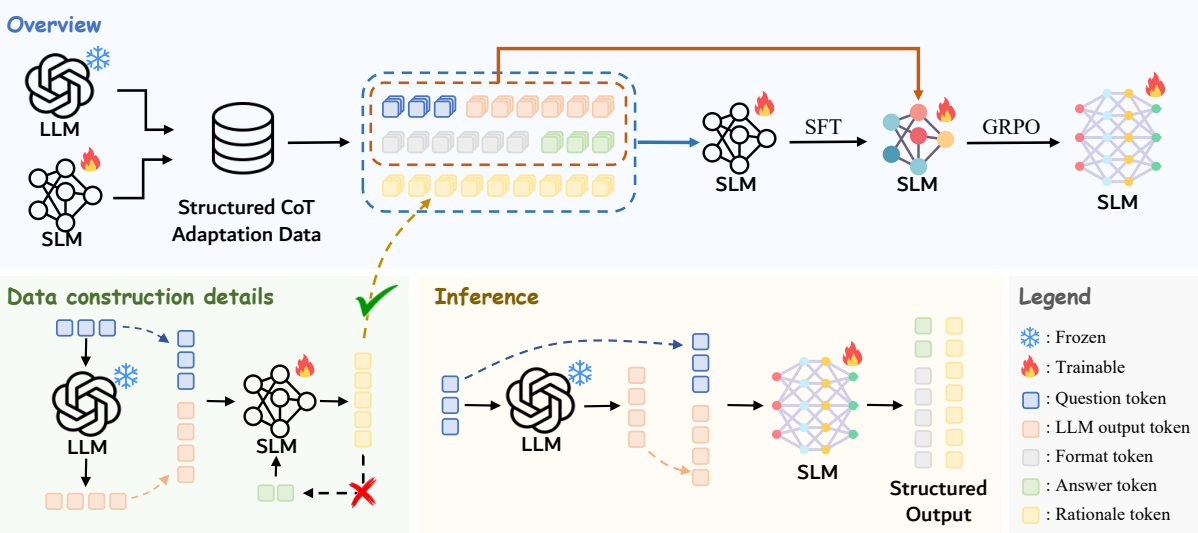


Figure 2: **Overview of DICE framework.** The training process comprises two sequential phases: DICE first employs a two-stage strategy to construct structured chain-of-thought data and subsequently implements a dual-tuning methodology to optimize the SLM to enforce rigorous format compliance. During inference, the trained SLM systematically analyzes and refines the natural language outputs from the LLM.

models (e.g., 32B and 72B parameters) optimized for diverse tasks tend to allocate more attention to solving difficult problems, but often at the expense of strict adherence to output formatting instructions. The fragility of structured outputs exacerbates this issue—minor deviations (e.g., a misplaced bracket in JSON) can lead to complete parsing failures, disproportionately penalizing larger models during evaluation.

A natural solution is to fine-tune LLMs on the supervised dataset, but it is associated with several critical challenges: (1) **Inefficiency**: fine-tuning LLMs typically requires prohibitive computational resources and extended training duration; (2) **Alignment Tax**: task-specific fine-tuning risks catastrophic forgetting, which can inadvertently lead to performance degradation (Ouyang et al., 2022; Bubeck et al., 2023; Jiang et al., 2024). (3) **Impracticality**: for many API-only LLMs such as GPT-4 (Achiam et al., 2023), fine-tuning is infeasible due to inaccessible of model parameters. Recent studies have investigated collaborative frameworks that utilize small language models (SLMs) to effectively adapt LLMs to domain-specific tasks. Some methods use the probability distribution shift of SLM during fine-tuning to calibrate the LLM outputs (Liu et al., 2024; Ormazabal et al., 2023), while others employ model collaboration techniques to facilitate multi-step reasoning generation and path search (Sun et al., 2024; Fan et al., 2025; Kim et al., 2025; Zheng et al., 2025). Additionally, several studies fine-tune SLMs to learn the correctional residuals between the ground-truth and the LLM-generated answers (Ji et al., 2024; Kim et al., 2024; Chen et al., 2024). However, these methods originally focus on enhancing the LLM's reasoning performance while overlooking its capability to follow instructions. Moreover, they fail to fully exploit the information embedded in the outputs of LLM, resulting in a high mis-correction rate. Thus, even when applied to structured reasoning tasks, these methods fail to adequately balance format output and reasoning performance.

To address these limitations, we propose a framework that adapts LLMs to structured reasoning tasks by guiDing SLMs to thInk with Chain-of-thought corrEction (DICE), as illustrated in Figure 2. The framework operates in two stages: first, the LLM is prompted to produce unstructured natural language responses, avoiding interference from complex formatting requirements that could degrade reasoning quality. Then, a trained SLM is deployed to refine these outputs into specific formats. To train the SLM, we employ a two-stage process to generate rationales and construct structured chain-of-thought (CoT) adaptation datasets, followed by a dual-tuning strategy that guides the SLMs to perform deep analysis on LLM outputs before generating final answers. The core innovation of DICE lies in our novel use of the model collaboration framework to enhance the instruction-following capabilities of LLMs and the analyze-

then-answer pattern used in SLMs generation. By utilizing chain-of-thought prompting to stimulate the reasoning ability of SLMs (Wei et al., 2023; Lyu et al., 2023; Srivastava et al., 2025), they are able to improve instruction-following ability without compromising the inherent reasoning performance of LLMs, thereby addressing the mis-correction issues observed in prior approaches.

We conduct extensive experiments on five reasoning benchmarks to validate DICE's effectiveness in adapting LLMs to downstream structured tasks. Compared to LLM with a 2-shot prompt, DICE achieves significant improvements, demonstrating average gains of 35.4% in format accuracy and 29.4% in content accuracy. Moreover, DICE consistently outperforms other baselines across nearly all evaluated datasets. Our key contributions can be summarized as follows:

- To the best of our knowledge, we are the first to identify the negative correlation between instruction-following ability and model scale in reasoning tasks: while larger models exhibit stronger reasoning capabilities, their adherence to instructions tends to decline.
- We introduce DICE, a lightweight framework that leverages SLM to adapt LLMs to structured reasoning tasks. DICE operates without modifying the LLMs' parameters, thereby circumventing the "alignment tax" associated with fine-tuning and preserving the LLMs' general knowledge.
- Extensive experiments show that DICE outperforms other baselines in improving instruction-following capabilities in reasoning tasks without compromising reasoning performance. Furthermore, DICE demonstrates superior generalizability across datasets and models, making it applicable in a wider range of scenarios.

2 Related Work

2.1 Instruction-Following Ability of LLMs

The capability of LLMs to follow user instructions is critical for practical applications. In recent years, numerous studies on model instruction-following have emerged. IFEval (Zhou et al., 2023) and CIF-Bench (LI et al., 2024) contain various instructions for evaluating the general instruction-following proficiency. FOFO (Xia et al., 2024) and StructuredRAG (Shorten et al., 2024) specifically target format compliance evaluation. Specifically, for tasks requiring structured outputs, constrained decoding-based methods (Willard and Louf, 2023; Koo et al., 2024; Dong et al., 2025) have been proposed to enforce models to generate responses in specific formats. However, such methods suffer from poor flexibility and simultaneously degrade the quality of model-generated content (Tam et al., 2024). To solve these problems, we first propose reasoning tasks with specific output formats to simultaneously evaluate model instruction-following and reasoning abilities. Secondly, we introduce a model collaboration-based approach that enhances the model's instruction-following ability while improving its reasoning performance.

2.2 Collaboration of SLMs and LLMs

Due to prohibitively high computational costs and inaccessibility to model parameters, direct fine-tuning of LLMs remains infeasible for most researchers. This challenge has driven extensive exploration into LLM and SLM collaborative frameworks for task-specific adaptation. Distribution alignment approaches (Liu et al., 2024; Ormazabal et al., 2023) attempt to integrate the output distribution shifts during SLM fine-tuning with LLM output distributions, but their practical applicability is circumscribed by the inaccessibility of full vocabulary probability distributions of various LLMs. Routing-based mechanisms (Sun et al., 2024; Aggarwal et al., 2024; Fan et al., 2025) decompose tasks into multi-step reasoning processes, where SLM selects optimal paths on multiple responses generated by LLM or dynamically selects between SLM and LLM during inference. However, these methods lead to increased computational cost and latency due to repeated LLM invocations. SLM correction frameworks (Kim et al., 2024; Ji et al., 2024; Kim et al., 2025) train SLM to learn the residual between the LLM output and target answer, but existing methods suffer from high mis-correction rates due to insufficient analysis and utilization of LLM output. When applied to structured reasoning tasks, these challenges persistently undermine the model's ability to maintain structured output fidelity while ensuring reasoning accuracy in complex questions. To address these limitations, this work constructs structured chain-of-thought adaptation benchmarks and leverages the analyze-then-answer pattern to enhance the reasoning and correction capability of SLMs.

3 Method

In this section, we delve into the technical details of adapting LLMs to structured reasoning tasks by guiDing SLMs to thInk with Chain-of-thought corrEction (DICE). In Section 3.1, we present the two-stage construction process of the structured chain-of-thought adaptation dataset. Next, in Section 3.2, we introduce the dual-tuning strategy used to fine-tune the SLM. The overview of DICE is presented in Figure 2 and Algorithm 1.

Algorithm 1: Algorithm of DICE

Input: Pretrained LLM $\mathcal{M}$, SLM π_θ, original dataset $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$, format token y_f, learning rate for SFT η_S, learning rate for SFT η_G, training iteration for SFT T_S, training iteration for GRPO T_G

// Structured $\mathcal{Q}$ construction
$\mathcal{Q} \leftarrow \emptyset$;
for $x^i, y^i \in \mathcal{D}$ **do**
  $y_o^i \sim \mathcal{M}(\cdot|x^i)$
  $r_1^i, y_{s,1}^i \sim \pi_0(\cdot|x^i, y_o^i)$
  if $y_{s,1}^i = y^i$ **then**
    $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x^i, y_o^i, y_f, r_1^i, y^i)\}$
  else
    $r_2^i, y_{s,2}^i \sim \pi_0(\cdot|x^i, y_o^i, y^i)$
    if $y_{s,2}^i = y^i$ **then**
      $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(x^i, y_o^i, y_f, r_2^i, y^i)\}$
    end
  end
end
// Fine-tuning the SLM
for $t = 1$ *to* T_S **do**
  $\mathcal{L}_{SFT}(\theta, \mathcal{Q}) \leftarrow \mathbb{E}_{\mathcal{Q}}[\mathcal{L}_{SFT}(\theta)]$ (Eq. 5)
  $\theta \leftarrow \theta - \eta_S \nabla_\theta \mathcal{L}_{SFT}(\theta, \mathcal{Q})$
end
for $t = 1$ *to* T_G **do**
  $\mathcal{L}_{GRPO}(\theta, \mathcal{Q}) \leftarrow \mathbb{E}_{\mathcal{Q}}[\mathcal{L}_{GRPO}(\theta)]$ (Eq. 9)
  $\theta \leftarrow \theta - \eta_S \nabla_\theta \mathcal{L}_{GRPO}(\theta, \mathcal{Q})$
end
Output: Fine-tuned SLM π_θ

3.1 Structured Chain-of-Thought Adaptation Dataset Construction

Given a pre-trained LLM $\mathcal{M}$, pre-trained SLM π_0, and question-answer pairs $\mathcal{D} = \{(x^i, y^i)\}_{i=1}^N$ from the original training set (N is the size of the benchmark), our approach begins by instructing the LLM $\mathcal{M}$ directly to obtain the natural language outputs:

$$y_o^i \sim \mathcal{M}(\cdot|x^i) \tag{1}$$

To address the problem of high mis-correction rate in prior approaches, we construct analytical data based on y_o, which can guide the SLM to reason before generating the final answers. To reduce computational cost and generate rationales more suitable for SLM to learn, we propose a two-stage methodology, akin to STaR (Zelikman et al., 2022), to instruct the pre-trained SLM π_0 to reason the LLM output and provide the predicted answer. Considering the limited instruction-following capability of π_0, we first generate two demonstrations with the assistance of LLM in the following two steps. In the first stage, the rationale r_1^i and predicted answer $y_{s,1}^i$ are formulated as:

$$(r_1^i, y_{s,1}^i) \sim \pi_0(\cdot|x^i, y_o^i) \tag{2}$$

Based on the assumption that rationales leading to correct predicted answers possess positive utility, we filter the generated outputs, retaining only those associated with accurate answers ($y_{s,1}^i = y^i$). In the subsequent stage, for the filtered-out samples, where π_0 alone struggles to generate meaningful rationales, we append the answer label y^i as a contextual hint to the original input of these challenging samples, then regenerate rationales and answers:

$$(r_2^j, y_{s,2}^j) \sim \pi_0(\cdot|x^j, y_o^j, y^j), \text{for } y_{s,1}^j \neq y^j \tag{3}$$

After generation, we repeat the aforementioned filtering procedure. Empirical results from our experiments demonstrate that, following this two-stage approach, the SLM successfully generates rationales yielding correct predictions for over 90% of the original training split. Consequently, we discard the filter-out samples in the second stage. The final rationale set can be formulated as $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 = \{r_1^i | y_{s,1}^i = y^i\}_{i=1}^{N_1} \cup \{r_2^j | y_{s,2}^j = y^j\}_{j=2}^{N_2}$, where N_1 and N_2 denote the number of samples retaining after each filtering procedure. Ultimately, we embed the rationales and answers into the required format (presented as y_f), obtaining the new training target $t^k = (y_f, r^k, y^k)$. The final structured chain-of-thought adaptation dataset is $\mathcal{Q} = \{(x^k, y_o^k, t^k)\}_{k=1}^{N_1+N_2}$.

3.2 Guiding SLMs to Think Through a Dual-Tuning Strategy

The most straightforward fine-tuning approach is SFT, wherein the π_0 is trained to predict all target tokens with equal emphasis. The training objective is to minimize the cross-entropy loss:

$$\mathcal{L}_{SFT}(\theta) = -\log \pi_\theta(t|x, y_o) \tag{4}$$

where $(x, y_o, t) \sim \mathcal{Q}$. The target t consists of three components, allowing us to decompose the

loss function and gradient into three corresponding terms:

$$\begin{aligned}\mathcal{L}_{SFT}(\theta) &= -\log \pi_\theta(y_f, r, y|x, y_o) \\ &= \mathcal{L}_f(\theta) + \mathcal{L}_r(\theta) + \mathcal{L}_y(\theta)\end{aligned} \tag{5}$$

$$\nabla_\theta \mathcal{L}_{SFT}(\theta) = \nabla_\theta \mathcal{L}_f(\theta) + \nabla_\theta \mathcal{L}_r(\theta) + \nabla_\theta \mathcal{L}_y(\theta) \tag{6}$$

In practice, these three components are intrinsically interwoven and vary in length across actual outputs, rendering separate computation infeasible. Moreover, the rationale r typically constitutes the longest segment; for example, in the MATH dataset requiring XML output, the average token ratio between y_f, r, and y is approximately $25:135:1$. Consequently, during gradient computation, $|\nabla_\theta \mathcal{L}_r(\theta)|$ can significantly exceed $|\nabla_\theta \mathcal{L}_f(\theta)|$ and $|\nabla_\theta \mathcal{L}_y(\theta)|$. This imbalance causes π_0 to prioritize minimizing $\mathcal{L}_r(\theta)$ during optimization, resulting in insufficient learning of y_f and y. Meanwhile, since r is generated by π_0 itself, it primarily contains in-domain knowledge that provides limited additional information. In contrast, novel knowledge such as user instruction and task-specific information is primarily contained in y_f and y, which necessitate more focused learning.

To address this challenge, we propose a dual-tuning strategy to progressively optimize the SLM. First, we conduct SFT utilizing Low-Rank Adaptation (LoRA (Hu et al., 2022)) on π_0 to rapidly acquire format specifications and the analyze-then-answer generation pattern, obtaining π_{SFT}. Subsequently, we employ a more granular fine-tuning method to further optimize π_{SFT} using the GRPO (Shao et al., 2024) algorithm (detailed in Appendix C). For model output $\hat{t} = (\hat{y_f}, \hat{r}, \hat{y})$, we design reward functions that assign rewards solely based on $\hat{y_f}$ and $\hat{y}$, neglecting $\hat{r}$. The total reward is calculated as follows:

$$\text{reward} = \begin{cases} 2, & \text{both } \hat{y_f} \text{ and } \hat{y} \text{ are correct} \\ 1, & \text{one of } \hat{y_f} \text{ and } \hat{y} \text{ is correct} \\ 0, & \text{both } \hat{y_f} \text{ and } \hat{y} \text{ are incorrect} \end{cases} \tag{7}$$

4 Experiments

4.1 Experimental Setup

Datasets and Metrics We evaluate the effectiveness of DICE from four dimensions: mathematical reasoning (GSM8K (Cobbe et al., 2021), MATH (Hendrycks et al., 2021)), commonsense reasoning (CommonsenseQA (Talmor et al., 2018), CSQA for short), domain-specific reasoning (MedQA-zh (Jin et al., 2021)), and implicit reasoning (StrategyQA (Geva et al., 2021)). To comprehensively evaluate the model's reasoning and instruction-following ability, we restructure the output into a more sophisticated XML format. We employ two metrics to evaluate the quality of the outputs: *Format Accuracy* (F-Acc) and *Content Accuracy* (C-Acc). Format Accuracy assesses adherence to structural elements and keywords prescribed by the template. Content accuracy is derived from the Exact Match (EM) score calculated for the final answer extracted from outputs. Notably, format compliance is a necessary condition for content accuracy. More details are in Appendix A.

Baselines We compare DICE against three methodological categories: (1) *Training-free method*: This category utilizes pre-trained large and small language models for response generation through 0-shot prompt and In-Context Learning (ICL). We also leverage the reflection baseline that feeds format-violated outputs from LLM ICL to the LLM and instruct it to reflect and regenerate the answer. (2) *Supervised Fine-Tuning (SFT) method*: In this category, SLMs are directly fine-tuned on the supervised training data. (3) *Model collaboration method*: Aligner (Ji et al., 2024), BBox-Adapter (Sun et al., 2024), and CoBB (Kim et al., 2024). Aligner leverages SLM to learn the mapping between LLM output and the ground-truth answer. BBox-Adapter scores iterative LLM generations via a trained evaluator, then applies beam search for optimal reasoning path selection. CoBB first constructs contrastive examples, then deploys SLM to learn from the pair-wise preference data through the ORPO (Hong et al., 2024) algorithm. **It should be noted that we apply all methods to the structured reasoning tasks, even though their original works primarily focus on content refinement.**

Implementation Details We select Qwen2.5-72B-Instruct-GPTQ-Int4 (Yang et al., 2024) as the LLM. The initial SLMs are derived from the instruction-tuned models within the Qwen2.5 series. For ICL, models generate responses through a 2-shot prompt. For the BBox-Adapter, we utilize a single generation step and classification mode. For CoBB, we generate one positive and one negative reasoning for each question. For both SFT and Aligner, the SLMs are trained for 3 epochs. For our proposed DICE, we initially fine-tune the SLMs using SFT for 2 epochs, followed by 1 epoch of

SLM	LLM	Method	GSM8K		MATH		CSQA		MedQA-zh		StrategyQA		Average	
			F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc
	72B	0-shot	0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
✗	72B	ICL	64.2	61.8	4.8	4.2	97.4	82.6	77.0	67.2	95.2	74.2	67.7	58.0
	72B	Reflection	64.4	62.0	4.8	4.2	97.4	82.6	77.2	67.4	95.2	74.2	67.8	58.1
	✗	0-shot	0.0	0.0	0.0	0.0	0.0	0.0	8.8	0.2	0.0	0.0	1.8	0.0
	✗	ICL	10.8	2.6	1.2	0.4	76.2	14.4	57.8	8.2	84.3	47.6	46.1	14.6
	✗	SFT	98.6	25.6	94.6	12.4	**100.0**	57.6	**100.0**	45.2	96.9	59.0	98.0	40.0
0.5B	72B	Aligner	**99.6**	46.2	96.2	48.6	**100.0**	78.6	**100.0**	86.8	96.9	69.4	98.5	65.9
	72B	BBox-Adapter	91.0	83.8	11.2	9.4	99.4	85.2	94.8	83.8	95.2	**79.0**	78.3	68.2
	72B	CoBB	98.4	94.2	96.6	76.6	97.4	79.6	99.8	84.0	99.6	71.2	98.4	81.1
	72B	DICE (Ours)	**99.6**	**95.2**	**99.4**	**79.0**	**100.0**	**85.8**	**100.0**	**88.0**	**100.0**	78.2	**99.8**	**85.2**
	✗	0-shot	0.0	0.0	0.4	0.2	7.4	0.0	13.8	0.0	0.0	0.0	4.3	0.0
	✗	ICL	77.4	46.0	1.0	0.6	81.2	31.0	99.0	31.4	98.3	53.7	71.4	32.5
	✗	SFT	98.8	46.4	98.0	25.8	**100.0**	70.8	**100.0**	71.2	98.3	65.9	99.0	56.0
1.5B	72B	Aligner	98.6	51.0	98.2	38.6	**100.0**	79.2	**100.0**	80.6	**100.0**	73.8	99.4	64.6
	72B	BBox-Adapter	93.4	87.0	11.4	9.6	99.0	85.0	98.4	**88.2**	94.8	78.6	79.4	69.7
	72B	CoBB	97.4	92.8	96.0	74.8	98.6	82.4	99.0	83.6	**100.0**	74.7	98.2	81.7
	72B	DICE (Ours)	**99.8**	**95.6**	**99.6**	**79.8**	**100.0**	**85.6**	**100.0**	87.8	**100.0**	**79.5**	**99.9**	**85.7**
	✗	0-shot	82.0	63.8	60.6	38.6	89.6	1.4	54.2	0.0	84.3	55.9	74.1	31.9
	✗	ICL	92.6	75.0	63.0	37.0	96.6	65.0	94.2	52.0	98.3	62.0	88.9	58.2
	✗	SFT	99.4	63.4	99.0	32.0	**100.0**	77.6	**100.0**	73.6	**100.0**	69.9	99.7	63.3
3B	72B	Aligner	**99.8**	61.8	98.8	50.6	**100.0**	80.0	**100.0**	83.6	99.1	74.2	99.5	70.0
	72B	BBox-Adapter	93.2	85.8	10.6	8.8	99.8	**85.6**	98.8	**88.6**	95.2	78.6	79.5	69.5
	72B	CoBB	97.0	92.2	96.4	74.6	98.4	83.4	99.6	86.4	99.6	73.4	98.2	82.0
	72B	DICE (Ours)	99.6	**94.2**	**99.6**	**77.8**	**100.0**	84.4	**100.0**	88.0	99.6	**80.4**	**99.8**	**85.0**

Table 1: **Performance comparison on five downstream reasoning tasks with XML output requirement.** All models originate from the Qwen2.5 model series. The base LLM is Qwen2.5-72B-Instruct-GPTQ-Int4. For each model size of SLM, the highest and second-highest scores are highlighted in **bold** and underlined, respectively.

Method	MATH JSON		MATH YAML	
	F-Acc	C-Acc	F-Acc	C-Acc
LLM (ICL)	10.4	7.6	0.0	0.0
SLM (SFT)	98.0	23.8	98.8	29.0
LLM + Aligner	97.2	46.6	98.6	44.6
LLM + BBox-Adapter	18.4	12.4	3.2	2.2
LLM + CoBB	99.0	76.2	99.6	76.0
LLM + DICE(Ours)	**99.8**	**80.0**	**100.0**	**79.6**

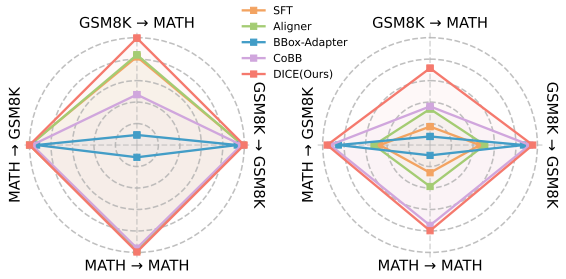
Table 2: **Performance comparison on MATH specifying JSON and YAML output format.** The model size of the LLM and SLM utilized is 72B and 1.5B.

GRPO fine-tuning. Further details on the implementation can be found in Appendix D.

4.2 Main Result

Table 1 presents the performance of DICE and other baselines on the five selected reasoning tasks with specialized XML output requirements. First, we observe that without any demonstrations, the Qwen2.5-72B-Instruct-GPTQ-Int4 largely fails to generate responses adhering to the specific formatting requirements specified in user instructions. When employing ICL, it exhibits substantial instability in format accuracy across diverse datasets. For instance, on more challenging benchmarks such as MATH, it tends to allocate excessive attention to problem-solving processes, consequently neglecting the formatting constraints outlined in instructions, results in substantially diminished formatting accuracy (below 5%). Furthermore, in the reflection baseline, even after feeding the incorrect responses along with feedback back into the LLM for regeneration, the format accuracy shows little improvement. This indicates that relying solely on LLMs cannot simultaneously balance output format and reasoning performance for these questions. However, after utilizing our proposed DICE framework with model size less than 3 billion parameters, near-perfect format accuracy (approaching 100%) is achieved consistently across all evaluated datasets, while simultaneously improving the content accuracy by an average of 29.4% compared to LLM using ICL. Moreover, under identical SLM size, DICE achieves either the best or second-best performance across all datasets, and obtains the highest average scores in both F-Acc and C-Acc. It significantly outperforms the collaboration-based baselines, including Aligner, BBox-Adapter, and CoBB, with average content accuracy gains of 18.4%, 16.1%, and 3.6%, respectively.

For more structures such as JSON and YAML, we also conduct experiments with 1.5B SLM on the MATH dataset, with results summarized in Table 2.



(a) Format Accuracy Performance (b) Content Accuracy Performance

Figure 3: **Cross-dataset generalization ability of different methods.** The 1.5B SLMs trained on GSM8K and MATH through different methods are evaluated on test sets of both benchmarks. "A→B" represents models that are trained on A and tested on B.

It is observed that our proposed DICE consistently achieves the highest F-Acc and C-Acc for both JSON and YAML formats, outperforming other baseline approaches. These results indicate that our proposed DICE framework not only adheres to user-specified output format constraints but also effectively harnesses the respective reasoning capabilities of both large and small language models, leading to high content accuracy.

4.3 Generalizability Analysis

Cross-Model Generalizability Our DICE framework solely requires the outputs from the LLMs, making it applicable to diverse LLMs in a plug-and-play manner. To systematically evaluate the cross-model generalizability, we employ the 1.5B SLM (trained to adapt Qwen2.5-72B-Instruct-GPTQ-Int4 in Table 1) to adapt other distinct LLMs: Qwen2.5-7B-Instruct (Yang et al., 2024), Meta-Llama-3-8B-Instruct (Grattafiori et al., 2024), and GPT-4.1-mini (OpenAI, 2025) to tasks with XML format requirements. As presented in Table 3, the results demonstrate that DICE can successfully adapt various large models to all XML-constrained reasoning tasks, achieving an average F-Acc exceeding 99.5%. Notably, DICE presents significant improvements in C-Acc compared to ICL baselines, with average gains of 16.3% (Qwen2.5-7B-Instruct), 5.8% (Meta-Llama-3-8B-Instruct), and 0.6% (GPT-4.1-min). Furthermore, compared to other baselines, DICE achieves superior performance in both F-Acc and C-Acc, underscoring its strong capability for cross-model generalization.

Cross-Dataset Generalizability To further investigate the generalization performance of our method across datasets, we conduct cross-dataset validation experiments using the 1.5B SLM in Table 1 on GSM8K and MATH datasets. Specifically, apart from the consistent train and test datasets, we assess both cross-dataset generalization scenarios: (1) evaluating MATH test performance of models trained on GSM8K, and (2) evaluating GSM8K test performance of models trained on MATH. The experimental results are visualized in Figure 3. It is observed that our DICE framework consistently achieves SOTA performance across all four evaluation dimensions compared to other baselines, which indicates that DICE not only enables SLM to effectively acquire domain-specific knowledge but also maintains strong performance across diverse datasets. Notably, when applying models trained on GSM8K to the more challenging MATH test set, DICE demonstrates substantial improvement in C-Acc, exceeding all baselines by at least 35%. This significant performance gap underscores DICE's capability to effectively leverage LLM outputs for generating more accurate responses, even when confronted with test data that exhibits greater complexity than the training samples. These findings collectively establish that the DICE framework attains exceptional cross-dataset generalizability relative to existing baseline methods.

4.4 Effectiveness Analysis

To further investigate the insights underlying the effectiveness of our proposed DICE framework, we conduct a comprehensive comparative analysis of consistency between natural language LLM outputs (y_o) and the structured outputs from SLM ($\hat{t}$) across different generative model collaboration strategies (Aligner, CoBB, and DICE). We propose four metrics to evaluate the consistency: (1) **Consistent Correct Rate (CCR)**: The proportion of samples where both y_o and $\hat{t}$ are correct. (2) **Correction Rate (ECR)**: The proportion of samples where y_o is incorrect but is corrected by the SLM. (3) **Mis-correction Rate (CER)**: The proportion of samples where y_o is correct but becomes incorrect after adaptation. (4) **Consistent Error Rate (EER)**: The proportion of samples where both y_o and $\hat{t}$ are incorrect. The consistency performance of the generative methods is presented in Figure 4.

The results demonstrate that DICE substantially outperforms both Aligner and CoBB by achieving significantly higher CCR and lower CER. Notably, DICE maintains a CER below 2% across all datasets, indicating its strong ability to preserve the correctness of LLM outputs. This suggests that the analyze-then-answer paradigm adopted in

Method	GSM8K		MATH		CSQA		MedQA-zh		StrategyQA		Average	
	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc	F-Acc	C-Acc
Qwen2.5-7B-Instruct (ICL)	96.2	85.0	78.8	52.6	55.8	41.4	93.8	72.0	83.8	61.6	81.7	62.5
Qwen2.5-7B-Instruct + Aligner	99.4	54.8	97.2	40.8	**100.0**	77.2	**100.0**	76.6	**99.6**	66.7	99.2	63.2
Qwen2.5-7B-Instruct + BBox-Adapter	95.4	84.0	73.8	43.8	49.4	37.4	91.6	72.2	83.0	60.7	78.6	59.6
Qwen2.5-7B-Instruct + CoBB	98.0	91.0	95.4	66.6	99.2	78.0	98.4	73.0	97.8	69.0	97.8	75.5
Qwen2.5-7B-Instruct + DICE	**99.8**	**92.2**	**99.4**	**71.0**	**100.0**	**78.6**	**100.0**	**81.4**	**99.6**	**70.7**	**99.8**	**78.8**
Meta-Llama-3-8B-Instruct (ICL)	98.8	76.4	82.4	25.8	97.4	63.6	97.0	39.8	96.5	66.4	94.4	54.4
Meta-Llama-3-8B-Instruct + Aligner	99.6	52.0	98.0	27.2	**100.0**	**68.4**	**100.0**	**65.0**	**100.0**	64.2	**99.5**	55.4
Meta-Llama-3-8B-Instruct + BBox-Adapter	99.0	75.4	80.4	20.4	86.8	54.8	80.2	36.8	92.1	68.6	87.7	51.2
Meta-Llama-3-8B-Instruct + CoBB	98.6	80.6	91.2	31.4	96.4	64.4	98.2	46.0	98.3	63.8	96.5	57.2
Meta-Llama-3-8B-Instruct + DICE	**100.0**	**81.6**	**99.6**	**33.2**	**100.0**	**68.4**	**100.0**	48.6	97.8	**69.0**	**99.5**	**60.2**
GPT-4.1-mini (ICL)	**100.0**	96.2	78.6	67.4	**100.0**	**82.8**	97.8	76.2	**100.0**	**83.0**	95.3	81.1
GPT-4.1-mini + Aligner	99.8	55.2	96.2	35.2	**100.0**	81.4	**100.0**	77.0	98.3	75.1	98.9	64.8
GPT-4.1-mini + BBox-Adapter	**100.0**	92.0	81.4	65.6	98.2	81.0	96.6	74.0	99.6	82.1	95.2	78.9
GPT-4.1-mini + CoBB	96.2	90.4	81.8	59.0	97.6	80.4	98.6	72.8	93.7	73.8	93.6	75.3
GPT-4.1-mini + DICE	**100.0**	**97.0**	**99.4**	**69.4**	**100.0**	**82.8**	**100.0**	**78.2**	**100.0**	81.2	**99.9**	**81.7**

Table 3: **Cross-model generalization ability of different methods.** The 1.5B SLMs of all methods are trained to adapt Qwen2.5-72B-Instruct-GPTQ-Int4 to reasoning tasks with XML format requirements.

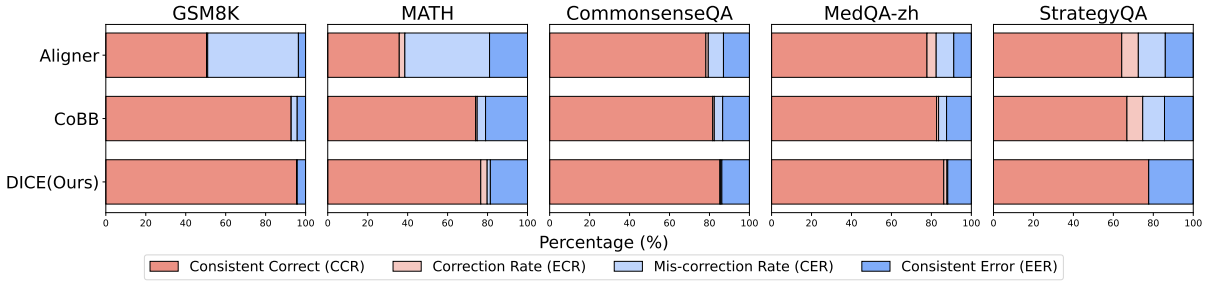


Figure 4: **The consistency analysis between natural language outputs from LLM and outputs in XML format from 1.5B SLMs in generative approaches.** We investigate the consistency in output correctness using four evaluation metrics: Mis-correction Rate (CER), Correction Rate (ECR), Consistent Error Rate (EER), and Consistent Correct Rate (CCR). These metrics provide a comprehensive insight into the strengths of the DICE framework.

DICE enables the SLM to more effectively utilize the information embedded in LLM outputs without introducing unnecessary modifications. Moreover, for questions beyond LLM's knowledge coverage, DICE still demonstrates the capability to partially correct erroneous outputs. In contrast, the Aligner and CoBB suffer from "over-correction dilemma": though they can achieve relatively high ECR, revealing the ability to correct errors for questions outside the LLM's knowledge domain, they also exhibit high CER which indicates that they fail to adequately analyze and assess the validity of LLM outputs, leading to erroneous modifications and an overall decline in performance.

4.5 Latency Analysis

In the inference stage, while our DICE framework and other model collaboration baselines enhance performance, they also introduce latency when compared to LLMs that directly employ ICL. To quantitatively assess this overhead, we compare the inference times of LLM with ICL against various model collaboration approaches. The evaluation experiment is conducted over the same 100 samples from the MATH test set on two NVIDIA A100 GPUs. The model collaboration methods (Aligner, Bbox-Adapter, CoBB, and DICE) use the Qwen2.5-72B-Instruct-GPTQ-Int4 LLM with the fine-tuned Qwen2.5-1.5B-Instruct SLM. To mitigate random variance, each method is evaluated across five independent inference runs, and the average latency is reported as the final result in Table 4.

Method	Time(s/sample)	C-Acc(%)
LLM (ICL)	**0.9191**	4.2
LLM + Aligner	1.1090	38.6
LLM + BBox-Adapter	2.2592	9.6
LLM + CoBB	1.0854	74.8
LLM + DICE(Ours)	1.0999	**79.8**

Table 4: **Inference time and performance of different methods.** The reported time represents the average inference latency per sample (in seconds) across different methods.

The experimental results indicate that our DICE

Strategy	MATH		CSQA	
	F-Acc	C-Acc	F-Acc	C-Acc
SFT	98.4	75.6	**100.0**	82.8
GRPO	92.0	73.2	89.2	77.2
SFT+GRPO	**99.8**	**78.8**	**100.0**	**85.6**

Table 5: **Ablation experiments on fine-tuning strategy used in DICE framework.** For all experiments, the 1.5B SLMs are fine-tuned for three epochs.

framework introduces an additional latency of approximately 20% compared to the LLM with ICL. However, this modest time overhead yields substantial performance gains: content accuracy improves by over 70% on the MATH test set and by an average of more than 25% across five different reasoning tasks selected in this work. The significant performance gains achieved with minimal time overhead make this trade-off entirely worthwhile. Furthermore, compared to other model collaboration methods, our DICE framework achieves optimal performance for a comparable computational cost, which demonstrates the high efficiency of our approach.

4.6 Ablation Study

Fine-tuning Strategy Ablation To validate the efficacy of our DICE framework, we conduct the ablation studies by replacing the finetuning procedure with either SFT or the GRPO algorithm exclusively. We train the 1.5B small models for 3 epochs on the structured chain-of-thought adaptation MATH and CSQA benchmarks (more training details can be found in Appendix D.2). The experimental results are presented in Table 5. It is observed that with the same fine-tuning epochs, the GRPO-only strategy exhibits suboptimal performance in both format adherence and content accuracy, which suggests that the GRPO algorithm is inefficient for models to learn format information from scratch. In contrast, the SFT-only strategy enables the model to effectively learn the user-specified output format and achieve competitive answer C-Acc compared to other baselines in Table 1, which reveals the effectiveness of the analyze-then-answer generation pattern of data construction in Section 3.1. Moreover, as discussed in Section 3.2, our dual-tuning strategy that applies GRPO after SFT enables the model to better attend to output format and final answer, achieving the highest F-Acc and C-Acc scores.

Correctness Ratio of y_o Ablation To quantify the impact of the correctness ratio of y_o in the training data and to determine the optimal ratio, we conduct a controlled experiment on the MATH dataset. In this experiment, we maintain a constant training set size of 5,000 samples while varying the proportion of correct y_o across four configurations: 100%, 75%, 50%, and 25%. All y_o outputs are generated by the Qwen2.5-72B-Instruct-GPTQ-Int4 to ensure consistency. Subsequently, we train four distinct 1.5B SLMs using these training sets and evaluate their performance on the test set by pairing them with various LLMs. The results of this experiment are presented in Appendix E. The experimental results indicate that the higher the proportion of correct y_o in the training set, the more the model tends to inherit the answers from the LLM, thereby reducing miscorrections and enabling better collaboration with stronger LLMs. In contrast, SLMs trained on datasets with lower y_o correctness ratios (e.g., 50%) exhibit stronger correction capabilities, making them more effective when paired with weaker LLMs. However, this trend does not extend linearly to extreme cases. For instance, training with only 25% correct y_o resulted in suboptimal performance across all LLM backbones. This is likely because the SLM's inherent capacity limitations prevent it from achieving high correction accuracy when exposed to predominantly incorrect examples. Overall, the model trained with 50% correct y_o not only achieved the best performance but also exhibited greater stability and robustness.

5 Conclusion

In this paper, we propose the DICE framework, a highly efficient and plug-and-play approach that adapts LLMs to structured reasoning tasks. We construct the structured chain-of-thought adaptation datasets that guide SLM to reason before generating final answers. We also design a dual-tuning strategy that leverages the strengths of both SFT and GRPO algorithms. Experimental results demonstrate that DICE achieves near-perfect format adherence while maintaining superior content accuracy, coupled with exceptional generalization capabilities. It addresses the issue of LLMs' instruction-following limitations without directly fine-tuning, achieving a dual enhancement in both reasoning and instruction-following. This innovation holds significant practical value in real-world applications with specific user requirements.

Limitations

Although our proposed DICE framework can effectively balance the trade-off between LLMs' instruction-following and reasoning capabilities, demonstrating superior performance in both structure adherence and content accuracy, it nonetheless possesses some limitations. For example, our approach introduces additional computational overhead. For each query, it first invokes LLM to generate natural language outputs, which are then refined by the SLM. However, for relatively simple questions that can be adequately addressed by SLM, invoking LLM is unnecessary and increases both computational cost and latency. Future work could incorporate a mechanism to assess question complexity beforehand, selectively engaging the LLM only when necessary, thereby optimizing resource usage.

Ethical Considerations

All models utilized in this study, except for GPT-4.1-mini, and all datasets are open-source. We downloaded the open-source models from their official releases on Hugging Face and accessed GPT models via the OpenAI API. Throughout, we strictly comply with all applicable user licenses. The datasets utilized in this research are sourced from the officially published repositories and are used exclusively for academic research purposes. All datasets are widely used and contain no personal or sensitive information. Therefore, there is no risk of personal information leakage here.

For AI usage, we only use AI assistants to check typos and grammar errors when writing.

Acknowledgements

This work was supported by the National Key R&D Program of China (No. 2022ZD0162101).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, and 1 others. 2024. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034.

Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2023. A general theoretical paradigm to understand learning from human preferences. *Preprint*, arXiv:2310.12036.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, and 1 others. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Dong Chen, Shuo Zhang, Yueting Zhuang, Siliang Tang, Qidong Liu, Hua Wang, and Mingliang Xu. 2024. Improving large models with small models: Lower costs and better performance. *Preprint*, arXiv:2406.15471.

Hailin Chen, Fangkai Jiao, Mathieu Ravaut, Nawshad Farruque, Xuan Phi Nguyen, Chengwei Qin, Manan Dey, Bosheng Ding, Caiming Xiong, Shafiq Joty, and Yingbo Zhou. 2025a. StructTest: Benchmarking LLMs' Reasoning through Compositional Structured Outputs. *Preprint*, arXiv:2412.18011.

Zhe Chen, Yusheng Liao, Shuyang Jiang, Pingjie Wang, Yiqiu Guo, Yanfeng Wang, and Yu Wang. 2025b. Towards omni-rag: Comprehensive retrieval-augmented generation for large language models in medical applications. *arXiv preprint arXiv:2501.02460*.

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*.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. 2025. Xgrammar: Flexible and efficient structured generation engine for large language models. *Preprint*, arXiv:2411.15100.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization. *Preprint*, arXiv:2402.01306.

Yijiang Fan, Yuren Mao, Longbin Lai, Ying Zhang, Zhengping Qian, and Yunjun Gao. 2025. G-boost: Boosting private slms with general llms. *arXiv preprint arXiv:2503.10367*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

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*.

Jiwoo Hong, Noah Lee, and James Thorne. 2024. Reference-free monolithic preference optimization with odds ratio. *arXiv e-prints*, pages arXiv–2403.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, and 1 others. 2022. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Jiaming Ji, Boyuan Chen, Hantao Lou, Donghai Hong, Borong Zhang, Xuehai Pan, Tianyi Alex Qiu, Juntao Dai, and Yaodong Yang. 2024. Aligner: Efficient alignment by learning to correct. *Advances in Neural Information Processing Systems*, 37:90853–90890.

Shuyang Jiang, Yusheng Liao, Ya Zhang, Yanfeng Wang, and Yu Wang. 2024. Taia: Large language models are out-of-distribution data learners. *Preprint*, arXiv:2405.20192.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Jaehyung Kim, Dongyoung Kim, and Yiming Yang. 2024. Learning to correct for qa reasoning with black-box llms. *arXiv preprint arXiv:2406.18695*.

Yujin Kim, Euiin Yi, Minu Kim, Se-Young Yun, and Taehyeon Kim. 2025. Guiding reasoning in small language models with llm assistance. *arXiv preprint arXiv:2504.09923*.

Terry Koo, Frederick Liu, and Luheng He. 2024. Automata-based constraints for language model decoding. *Preprint*, arXiv:2407.08103.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. *Preprint*, arXiv:2309.06180.

Yizhi LI, Ge Zhang, Xingwei Qu, Jiali Li, Zhaoqun Li, Zekun Wang, Hao Li, Ruibin Yuan, Yinghao Ma, Kai Zhang, Wangchunshu Zhou, Yiming Liang, Lei Zhang, Lei Ma, Jiajun Zhang, Zuowen Li, Stephen W. Huang, Chenghua Lin, and Jie Fu. 2024. CIF-Bench: A Chinese Instruction-Following Benchmark for Evaluating the Generalizability of Large Language Models. *Preprint*, arXiv:2402.13109.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A Smith. 2024. Tuning language models by proxy. *arXiv preprint arXiv:2401.08565*.

Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. 2023. Faithful chain-of-thought reasoning. *Preprint*, arXiv:2301.13379.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. 2024. A comprehensive overview of large language models. *Preprint*, arXiv:2307.06435.

OpenAI. 2025. Gpt-4.1 mini documentation. `https://platform.openai.com/docs/models/gpt-4.1-mini`.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. 2023. Comblm: Adapting black-box language models through small fine-tuned models. *arXiv preprint arXiv:2305.16876*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, and 1 others. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Connor Shorten, Charles Pierse, Thomas Benjamin Smith, Erika Cardenas, Akanksha Sharma, John Trengrove, and Bob van Luijt. 2024. Structuredrag: Json response formatting with large language models. *arXiv preprint arXiv:2408.11061*.

Gaurav Srivastava, Shuxiang Cao, and Xuan Wang. 2025. Towards reasoning ability of small language models. *Preprint*, arXiv:2502.11569.

Haotian Sun, Yuchen Zhuang, Wei Wei, Chao Zhang, and Bo Dai. 2024. Bbox-adapter: Lightweight adapting for black-box large language models. *arXiv preprint arXiv:2402.08219*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2018. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. 2024. Let me speak freely? a study on the impact of format restrictions on performance of large language models. *arXiv preprint arXiv:2408.02442*.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, and 1 others. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Brandon T. Willard and Rémi Louf. 2023. Efficient guided generation for large language models. *Preprint*, arXiv:2307.09702.

Congying Xia, Chen Xing, Jiangshu Du, Xinyi Yang, Yihao Feng, Ran Xu, Wenpeng Yin, and Caiming Xiong. 2024. Fofo: A benchmark to evaluate llms' format-following capability. *arXiv preprint arXiv:2402.18667*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, and 1 others. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. *Preprint*, arXiv:2203.14465.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2025. Is in-context learning sufficient for instruction following in llms? *Preprint*, arXiv:2405.19874.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Wenhao Zheng, Yixiao Chen, Weitong Zhang, Souvik Kundu, Yun Li, Zhengzhong Liu, Eric P. Xing, Hongyi Wang, and Huaxiu Yao. 2025. Citer: Collaborative inference for efficient large language model decoding with token-level routing. *Preprint*, arXiv:2502.01976.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *Preprint*, arXiv:2403.13372.

Ruiqi Zhong, Dhruba Ghosh, Dan Klein, and Jacob Steinhardt. 2021. Are larger pretrained language models uniformly better? comparing performance at the instance level. *Preprint*, arXiv:2105.06020.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Datasets and Metrics

This section presents further details about the datasets and the two evaluation metrics used in the work.

A.1 Datasets

In the experiments of this work, we select five commonly used datasets from four types of reasoning tasks: mathematical reasoning, commonsense reasoning, task-specific reasoning, and implicit reasoning.

- **GSM8K** (Cobbe et al., 2021), short for "Grade School Math 8K", is a benchmark consisting of 8,500 question-answering math problems designed to evaluate the fundamental mathematical reasoning abilities of models. In this dataset, each problem is paired with step-by-step reasoning and the numerical answer.

- **MATH** (Hendrycks et al., 2021) is a more challenging question-answering dataset consisting of 12,500 algebra, calculus, geometry, and precalculus problems sourced from high school mathematics competitions. Compared to GSM8K, MATH requires deeper mathematical knowledge and more sophisticated multi-step problem-solving capabilities.

- **CommonsenseQA** (Talmor et al., 2018) is a multiple-choice question-answering benchmark designed to evaluate model's commonsense reasoning ability. It contains 12.1k questions sourced from ConceptNet, requiring models to leverage general world knowledge and contextual understanding to select the correct answers from the given five options.

- **MedQA** (Jin et al., 2021) is a domain-specific dataset containing 61,097 multiple-choice questions from the United States Medical Licensing Examination (English), Chinese National Medical Licensing Exam (simplified Chinese), and Taiwan's Medical Licensing Exam (traditional Chinese). In this paper, we only sample data from the simplified Chinese subset to enrich the language diversity of our experiment, so we denote the dataset as MedQA-zh.

- **StrategyQA** (Geva et al., 2021) is a binary benchmark for evaluating model's strategic reasoning and implicit task understanding capabilities. It includes 2,290 True/False questions that require models to decompose complex problems into executable reasoning steps.

Notably, to reduce computational costs, we randomly sample 1,000 training examples and 500 test examples from each dataset as our original training and evaluation sets (the test set of StrategyQA contains only 229 examples). When constructing the new structured chain-of-thought dataset illustrated in Section 3.1, we prompt the LLM to sample five responses for each training sample. This allows us to expand the size of the training set to approximately 5,000 examples without increasing the number of LLM invocations, although some samples are filtered out after the two-stage rationale generation process.

A.2 Metrics

We design two metrics: format accuracy and content accuracy, to respectively evaluate the model's format adherence and reasoning capability:

- **Format accuracy (F-Acc)**: Proportion of samples with correct output format in all outputs. We employ string matching to extract the model outputs. An output is classified as having correct formatting only if it contains all specified keywords in the required format or can be automatically converted into the specific structure using standard toolkits.

- **Content accuracy (C-Acc)**: Content accuracy is measured by calculating the Exact Match (EM) score between the extracted final answer from the model's response and the ground truth. Notably, final answers can only be fully extracted when the output format is correct; for format-incorrect samples, their final answers remain unextractable and are therefore judged as incorrect. Consequently, any output with correct content must necessarily have a correct format.

XML

```
```xml format(\n)
<?xml version="1.0" encoding="UTF8"?>(\n)
<Answer>(\n)
<Step by step reasoning>{"step1":"{reasoning1 placeholder}",
"step2":"{reasoning2 placeholder}",...}</Step by step
reasoning>(\n)
<Final answer>{answer placeholder}</Final answer>(\n)
</Answer>(\n)
```
```

JSON

```
```json format(\n)
{"step by step reasoning":{"step1":"{reasoning1 placeholder}",
"step2":"{reasoning2 placeholder}",...},"final answer":
"{answer placeholder}"}(\n)
```
```

YAML

```
```yaml format(\n)
step_by_step_reasoning:(\n)
  -step1: "{reasoning1 placeholder}"(\n)
  -step2: "{reasoning2 placeholder}"(\n)
  ...(\n)
final_answer:(\n)
  -answer: {answer placeholder}(\n)
```
```

Figure 5: Format templates for GSM8K, MATH, and StrategyQA datasets. The symbol (\n) indicates the presence of a newline character at that specific position.

B Format Details

In our experiment, we employed three formats: XML, JSON, and YAML. For the three question-answering datasets, including GSM8K, MATH,

XML

```xml format(\n)
<?xml version="1.0" encoding="UTF8"?>(\n)
<Answer>(\n)
<Step by step reasoning>{"step1":"{reasoning1 placeholder}",
"step2":"{reasoning2 placeholder}",...}</Step by step
reasoning>(\n)
<Answer label>{answer label placeholder}</Answer label>(\n)
<Result>{result placeholder}</Result>(\n)
</Answer>(\n)
```

JSON

```json format(\n)
{"step by step reasoning":{"step1":"{reasoning1 placeholder}",
"step2":"{reasoning2 placeholder}",...},"answer label":"{answer
label placeholder}","result": "{result placeholder}"}(\n)
```

YAML

```yaml format(\n)
step_by_step_reasoning:(\n)
  -step1: "{reasoning1 placeholder}"(\n)
  -step2: "{reasoning2 placeholder}"(\n)
  ...(\n)
answer_label:(\n)
  -label: "{answer label placeholder}"(\n)
result:(\n)
  -answer: "{answer placeholder}"(\n)
```

Figure 6: Format templates for CommonsenseQA and MedQA datasets. The symbol (\n) indicates the presence of a newline character at that specific position.

and StrategyQA, the formatted output incorporates step-by-step reasoning along with the final answer. For multiple-choice datasets including CommonsenseQA and MedQA, the model is additionally required to generate the selected option label along with its corresponding answer, in addition to the reasoning steps. The basic templates of the three formats for GSM8K, MATH, StrategyQA and CommonsenseQA, MedQA are shown in Figure 5 and 6, respectively. As illustrated in Figure 7, we provide representative examples from MATH, CommonsenseQA, and StrategyQA.

C Algorithm Details of DICE

In this section, we elaborate on the core details of the GRPO algorithm and present the complete DICE algorithm in Algorithm 1.

C.1 GRPO Algorithm Details

In the fine-tuning process of LLMs, reinforcement learning (RL) plays a pivotal role (Schulman et al., 2017; Rafailov et al., 2024; Azar et al., 2023; Ethayarajh et al., 2024). Although the traditional Proximal Policy Optimization (PPO (Schulman et al., 2017)) algorithm has been widely adopted for LLM fine-tuning, it requires maintaining a separate value network comparable in size to the policy model for advantage function estimation, leading to substantial memory consumption and computational overhead in large-scale scenarios. To address these challenges, the Group Relative Policy Optimization (GRPO (Shao et al., 2024)) algorithm is proposed, which seeks to minimize dependence on value networks while preserving the stability and efficiency of policy updates.

The GRPO framework operates by sampling a group of actions from the current policy and calculating relative advantages within this group, thereby eliminating the need for a critic model. The advantage estimation can be formulated as:

$$A_i = \frac{r_i - \text{mean}(\{r_1, r_2, \cdots, r_G\})}{\text{std}(\{r_1, r_2, \cdots, r_G\})} \tag{8}$$

where G denotes the group size (number of sampled actions per iteration). The complete loss function of GRPO can be expressed as:

$$\mathcal{L}_{GRPO}(\theta) = \frac{1}{G}\sum_{i=1}^{G}\Bigg(\min\Bigg(\frac{\pi_\theta(o_i)}{\pi_{\theta_{\text{old}}}(o_i)}A_i, \text{clip}\Big(\frac{\pi_\theta(o_i)}{\pi_{\theta_{\text{old}}}(o_i)}, 1-\varepsilon, 1+\varepsilon\Big)A_i\Bigg) - \beta\mathbb{D}_{KL}(\pi_\theta \,\|\, \pi_{\text{ref}})\Bigg) \tag{9}$$

C.2 Overview of DICE Algorithm

The overview algorithm of our proposed DICE framework is shown in Algorithm 1.

D Implementation Details

D.1 Structured Data Construction

For SFT and Aligner, we constructed the training sets using only the original data. Since the CommonsenseQA and MedQA datasets provide only the answers without any reasoning information, the target structured outputs for these two datasets in SFT and Aligner contain only the selected option label and the corresponding answer, without step-by-step reasoning. The step-by-step reasoning in GSM8K, MATH, and StrategyQA is derived from the original benchmarks.

For BBox-Adapter, we first prompt the Qwen2.5-72B-Instruct-GPTQ-Int4 to generate reasoning for each training sample in CommonsenseQA and MedQA. Then we use the reasoning and ground-truth answer of all five benchmarks to construct standard structured outputs. Subsequently, we instruct the LLM to generate five candidates for each question in the training set with a 2-shot prompt. The small model was then trained using both the standard structured outputs and candidates.

For CoBB, we generate positive reasoning via Qwen2.5-72B-Instruct-GPTQ-Int4 and randomly sample reasoning from other questions as negative reasoning. This approach allowed us to construct both positive and negative structured outputs.

D.2 Fine-tuning Details

Our experiments were conducted on NVIDIA A100 GPUs (80G memory). When running all baselines and our proposed DICE method, we utilized the vLLM (Kwon et al., 2023), LLama-Factory (Zheng et al., 2024), and SWIFT (Zhao et al., 2024) frameworks for model fine-tuning.

Main Experiment In the main experiments (Tables 1 and 2), we use Qwen2.5-72B-Instruct-GPTQ-Int4 as the large language model, and Qwen2.5-0.5B-Instruct, Qwen2.5-1.5B-Instruct, and Qwen2.5-3B-Instruct as the small models. All models are trained on two A100 GPUs with bf16 precision. Other detailed experimental configurations are: (1) For SFT and Aligner baselines, LoRA fine-tuning is applied with rank 32 and alpha 64 for 3 epochs. The training process uses a batch size of 64, learning rate of 2×10^{-4}, warmup ratio of 0.1, and weight decay of 0.1. (2) BBox-Adapter adopts full fine-tuning for 3 epochs. During training, the candidate count is set to 5, max length to 1, batch size to 100, and the model operates in classification mode. The remaining parameters are kept consistent with the original paper. (3) CoBB utilizes LoRA rank 32 and alpha 64 for 5 epochs. The hyperparameter λ is fixed at 0.1, while the training employs a batch size of 64 and a learning rate of 1×10^{-5}, maintaining other parameters as in the original work. (4) For our proposed DICE framework, both SFT and GRPO stages adopt LoRA rank 32 and alpha 64. During the SFT stage, the SLM is trained for 2 epochs with batch size 64, learning rate 2×10^{-4}, warmup ratio 0.1, and weight decay 0.1. In the subsequent GRPO stage, the hyperparameter G is set to 16, the learning rate is reduced to 1×10^{-5}, temperature is set to 0.8, batch size increases to 128, warmup ratio decreases to 0.05, and training for 1 additional epoch.

Ablation Study In the ablation study, we utilize Qwen2.5-72B-Instruct-GPTQ-Int4 as the LLM and Qwen2.5-1.5B-Instruct as the original SLM. All experiments employ LoRA with a rank of 32 and an alpha of 64, conducted at bf16 precision. In the SFT-only setting, we set the learning rate to 2e-4, batch size to 64, warmup ratio and weight decay to 0.1, and train for 3 epochs. In the GRPO-only setting, we set the hyperparameter G to 16, learning rate to 2e-5, batch size to 128, temperature to 0.9, and also train for 3 epochs. All other hyperparameters remain consistent with those used in the main experiments.

E Correctness Ratio of y_o Ablation Experiment Result

The content accuracy of models trained with different correctness ratio of y_o and inferenced with different LLMs are illustrated in Table 6.

Ratio	100%	75%	50%	25%
Qwen2.5-72B + DICE	79.0	**79.4**	78.0	74.6
Qwen2.5-7B + DICE	71.8	70.6	**72.0**	67.8
Llama3-8B + DICE	33.6	33.0	**34.4**	34.0
GPT-4.1-mini + DICE	67.8	68.8	**70.8**	65.4
Average	63.1	63.0	**63.8**	60.5

Table 6: **Ablation experiments on the correctness ratio of y_o.** The models' name Qwen2.5-72B, Qwen2.5-7B, and Llama3-8B are short for Qwen2.5-72B-Instruct-GPTQ-Int4, Qwen2.5-7B-Instruct, and Meta-Llama-3-8B-Instruct.

MATH

- Sample

Question: Evaluate $\\left\\lceil3\\left(6-\\frac12\\right)\\right\\rceil$.
Answer: Firstly, $3\\left(6-\\frac12\\right)=18-1-\\frac12=17-\\frac12$. Because $0\\le\\frac12<1$, we have $\\left\\lceil17-\\frac12\\right\\rceil =\\boxed{17}$.

- Structured Output

```xml format\n<?xml version="1.0" encoding="UTF-8"?>\n<Answer>\n<Step by step reasoning>{"step1":"Firstly, $3\\left(6-\\frac12\\right)=18-1-\\frac12=17-\\frac12$.","step2":"Because $0\\le\\frac12<1$, we have $\\left\\lceil17-\\frac12\\right\\rceil=\\boxed{17}$."}</Step by step reasoning>\n<Final answer>17</Final answer>\n</Answer>\n```

```json format\n{"step by step reasoning":{"step1":"Firstly, $3\\left(6-\\frac12\\right)=18-1-\\frac12=17-\\frac12$.","step2":"Because $0\\le\\frac12<1$, we have $\\left\\lceil17-\\frac12\\right\\rceil=\\boxed{17}$."},"final answer": "17"}\n```

```yaml format\nstep_by_step_reasoning:\n -step1: "Firstly, $3\\left(6-\\frac12\\right)=18-1-\\frac12=17-\\frac12$."\n -step2: "Because $0\\le\\frac12<1$, we have $\\left\\lceil17-\\frac12\\right\\rceil=\\boxed{17}$."\nfinal_answer:\n -answer: 17\n```

CommonsenseQA

- Sample

Question: Google Maps and other highway and street GPS services have replaced what?
Choices: A.united states, B.mexico, C.countryside, D.atlas, E.oceans
Answer: D
LLM-generated Reasoning: The question is asking about what has been replaced by Google Maps and other GPS services. We need to analyze each option to determine which one is most likely to be replaced by GPS services. A. United States: This is a country, not a tool or service that could be replaced by GPS. B. Mexico: This is also a country, not a tool or service. C. Countryside: This refers to rural areas, not a tool or service. D. Atlas: An atlas is a collection of maps, which is a tool used for navigation and understanding geographic locations. E. Oceans: This refers to large bodies of water, not a tool or service. The most logical answer is that GPS services have replaced atlases.

- Structured Output

```xml format\n<?xml version="1.0" encoding="UTF-8"?>\n<Answer>\n<Step by step reasoning>{"step1":"The question is asking about what has been replaced by Google Maps and other GPS services.","step2":"We need to analyze each option to determine which one is most likely to be replaced by GPS services.","step3":"A. United States: This is a country, not a tool or service that could be replaced by GPS.","step4":"B. Mexico: This is also a country, not a tool or service.","step5":"C. Countryside: This refers to rural areas, not a tool or service.","step6":"D. Atlas: An atlas is a collection of maps, which is a tool used for navigation and understanding geographic locations.","step7":"E. Oceans: This refers to large bodies of water, not a tool or service.","step8":"The most logical answer is that GPS services have replaced atlases."}</Step by step reasoning>\n<Answer label>D</Answer label>\n<Result>atlas</Result>\n</Answer>\n```

```json format\n{"step by step reasoning":{"step1":"The question is asking about what has been replaced by Google Maps and other GPS services.","step2":"We need to analyze each option to determine which one is most likely to be replaced by GPS services.","step3":"A. United States: This is a country, not a tool or service that could be replaced by GPS.","step4":"B. Mexico: This is also a country, not a tool or service.","step5":"C. Countryside: This refers to rural areas, not a tool or service.","step6":"D. Atlas: An atlas is a collection of maps, which is a tool used for navigation and understanding geographic locations.","step7":"E. Oceans: This refers to large bodies of water, not a tool or service.","step8":"The most logical answer is that GPS services have replaced atlases."},"answer label": "D","result": "atlas"}\n```

```yaml format\nstep_by_step_reasoning:\n -step1: "The question is asking about what has been replaced by Google Maps and other GPS services."\n -step2: "We need to analyze each option to determine which one is most likely to be replaced by GPS services."\n -step3: "A. United States: This is a country, not a tool or service that could be replaced by GPS."\n -step4: "B. Mexico: This is also a country, not a tool or service."\n -step5: "C. Countryside: This refers to rural areas, not a tool or service."\n -step6: "D. Atlas: An atlas is a collection of maps, which is a tool used for navigation and understanding geographic locations."\n -step7: "E. Oceans: This refers to large bodies of water, not a tool or service."\n -step8: "The most logical answer is that GPS services have replaced atlases."\nanswer_label:\n -label: "D" \nresult:\n -answer: "atlas"\n```

StrategyQA

- Sample

Question: Would a Monoamine Oxidase candy bar cheer up a depressed friend?
Answer: False
Reasoning: Depression is caused by low levels of serotonin, dopamine and norepinephrine. Monoamine Oxidase breaks down neurotransmitters and lowers levels of serotonin, dopamine and norepinephrine.

- Structured Output

```xml format\n<?xml version="1.0" encoding="UTF-8"?>\n<Answer>\n<Step by step reasoning>{"step1":"Depression is caused by low levels of serotonin, dopamine and norepinephrine.","step2":"Monoamine Oxidase breaks down neurotransmitters and lowers levels of serotonin, dopamine and norepinephrine."}</Step by step reasoning>\n<Final answer>False</Final answer>\n</Answer>\n```

```json format\n{"step by step reasoning":{"step1":"Depression is caused by low levels of serotonin, dopamine and norepinephrine.","step2": "Monoamine Oxidase breaks down neurotransmitters and lowers levels of serotonin, dopamine and norepinephrine."},"answer": False}\n```

```yaml format\nstep_by_step_reasoning:\n -step1: "Depression is caused by low levels of serotonin, dopamine and norepinephrine."\n -step2: "Monoamine Oxidase breaks down neurotransmitters and lowers levels of serotonin, dopamine and norepinephrine."\nfinal_answer:\n -answer: False```

Figure 7: Examples from MATH, CommonsenseQA, and StrategyQA. The original CommonsenseQA dataset does not contain reasoning information; therefore, we instruct the LLM to generate reasoning.
```