TOWARD EFFECTIVE TOOL-INTEGRATED REASONING VIA SELF-EVOLVED PREFERENCE LEARNING

Yifei Chen, Guanting Dong, Zhicheng Dou*
Renmin University of China
{zhangboguodong, dou}@ruc.edu.cn

ABSTRACT

Tool-Integrated Reasoning (TIR) enables large language models (LLMs) to enhance their internal reasoning ability by integrating external tools. However, models with TIR often exhibit suboptimal behaviors, including insufficient tool calls, excessive tool calls, and overthinking after receiving tool call results. How to empower LLMs to perform TIR efficiently and accurately, while stabilizing the reasoning process, remains an open challenge. In this paper, we first analyze the impact of tool calls on model reasoning from the perspective of information entropy. We find that when tool call results are provided, the information entropy of subsequent reasoning content will show a clear trend of change, and the overall information entropy of the reasoning chain will vary depending on the number of tool calls. Based on these observations, we propose Tool-Light, a framework designed to encourage LLMs to perform TIR efficiently and accurately. Our framework consists of dataset construction and multi-stage fine-tuning. For dataset construction, we use the trained model for continuous self-evolved sampling, integrating two methods: vanilla sampling and entropy-guided sampling. At the same time, during the sampling process, we design strict criteria for selecting positive-negative pairs. For the training process, we introduce a two-stage method, which includes a Supervised Fine-Tuning (SFT), and Self-Evolved Direct Preference Optimization (DPO). Test results on 10 datasets reveal the effectiveness of Tool-Light, significantly improving the efficiency and accuracy of the model in completing TIR tasks. Our codes are released at https://github.com/RUC-NLPIR/Tool-Light.

1 INTRODUCTION

Recently, large language models (LLMs) have demonstrated strong reasoning abilities after training (Team et al., 2025; Li et al., 2025i; Huang & Chang, 2023; Sun et al., 2023). They excel in tasks such as mathematical reasoning (Shao et al., 2024; Yuan et al., 2023; Li et al., 2024b), logical puzzles (Xie et al., 2025), and code generation (Zhong et al., 2024; Li & Wang, 2025). However, when faced with more challenging tasks (such as deep information retrieval and precise computation (Min et al., 2024; Qin et al., 2024)), those trained LLMs often struggle when relying solely on their internal reasoning capabilities. To solve those problems, Tool-Integrated Reasoning (TIR) (Paranjape et al., 2023; Gou et al., 2024; Li et al., 2025b; Wang et al., 2025d) has emerged. Compared to general reasoning models, TIR models can autonomously utilize external tools during the reasoning process to compensate for the deficiencies in their internal knowledge or abilities (Chen et al., 2025a; Song et al., 2025; Li et al., 2024a; 2025b;f; Dong et al., 2025c;b).

While introducing external tools enables models to enhance performance on reasoning tasks, there still exist challenges, such as unreasonable tool calls or overthinking. For example, during the TIR process, the model often exhibits suboptimal tool call patterns, such as excessive or insufficient tool calls (Wu et al., 2025b; Bai et al., 2025). Additionally, when low-quality tool call results are provided, models may experience overthinking or even analysis paralysis (Zheng et al., 2025; Sui et al., 2025; Chen et al., 2025b). We collectively refer to them as incorrect tool calls. Recent studies have focused on optimizing tool calls with the help of reinforcement learning (RL). These

*Corresponding author.

studies include scenarios of using a single tool and scenarios of using multiple tools (Bai et al., 2025; Feng et al., 2025a; Wu et al., 2025b; Wang et al., 2025e; Dong et al., 2025a; Wang et al., 2025a). Nevertheless, existing studies typically focus on the issue of tool-overuse, neglecting both tool underuse and the impact of tool call results on subsequent reasoning processes (Qian et al., 2025). We consider that these works do not comprehensively address incorrect tool call issues.

At present, many works have analyzed reasoning tasks from the perspective of information entropy (Cui et al., 2025; Wang et al., 2025c). They find that the high-entropy part of the reasoning chain often determines reasoning direction. Inspired by these works, we conduct preliminary experiments and analyze the TIR task's information entropy. We find that after receiving tool call results, the information entropy of the model's subsequent outputs will fluctuate. We also observe that when multiple correct reasoning paths exist, the path with fewer tool calls tends to have a lower overall entropy distribution compared to those with more. Based on the limitations of existing work and the TIR task's entropy characteristics, we attempt to explore the efficiency of the TIR task from the following two perspectives:

- **From the training perspective:** Can we improve the effectiveness of tool call in TIR tasks from the perspective of post-training (including the algorithm side and the data side)?
- **From the inference perspective:** How can we apply the entropy distribution characteristics of the model's TIR process to guide data sampling?

In this paper, we propose Tool-Light, a framework that optimizes the reasoning capability and tool call effectiveness of TIR models from the data construction and algorithmic perspectives. From the data construction perspective, we introduce an innovative entropy-guided sampling method. This approach first generates a main chain, then branches from the highest-entropy positions to create multiple paths. We then apply sampling criteria to derive high-quality positive-negative pairs. To ensure data diversity and balance, we integrate this approach with direct sampling, producing a hybrid method that sustains performance while improving tool-call effectiveness. From the algorithmic perspective, we propose the "Two-stage TIR Training" pipeline. First, we perform supervised fine-tuning (SFT) on the original model, and then conduct self-evolved DPO training. In self-evolved DPO stage, we divide the training into two parts: Pre-Aligned DPO Training and Self-Evolved DPO Alignment. The former is used to improve the model's reasoning ability while reducing the redundant tool calls. The latter alternates sampling and training process, enabling the model to learn necessary tool call. At the same time, the difficulty of the training data and the model's capability can be adapted, thereby maintaining its original reasoning ability.

To comprehensively evaluate the capabilities of Tool-Light, we use ten challenging reasoning tasks, including knowledge-intensive and mathematical-reasoning tasks. Tool-Light not only ensures stable reasoning performance but also significantly improves the efficiency and accuracy of tool calls. In summary, our contributions are as follows:

- We explore and analyze the TIR paradigm from the perspective of information entropy, demonstrating the connection between TIR effectiveness and entropy change.
- We propose an innovative entropy-guided sampling strategy, which is combined with a two-stage training method incorporating a self-evolution mechanism, thereby enhancing the effectiveness of the TIR process.
- Experiment results across 10 challenging reasoning datasets prove the effectiveness of Tool-Light. Further quantitative analyses offer practical guidance for efficient tool-integrated reasoning.

2 Related Work

Effective Tool-integrated Reasoning. Tool-Integrated Reasoning (TIR) refers to guiding models to leverage external tools during reasoning (Yang et al., 2025). Teaching models to use tools correctly and rationally is a core challenge in TIR (Su et al., 2025a; Fang et al., 2025; Li et al., 2025d;c; Tao et al., 2025; Li et al., 2025j; Qiao et al., 2025; Qin et al., 2023; Schick et al., 2023; Lu et al., 2023; Dong et al., 2026). Currently, fine-tuning is the main method to guide the model in efficiently completing the TIR task (Li et al., 2025e). For example, IKEA (Huang et al., 2025) and SMART (Qian et al., 2025) design training methods based on metacognitive theory (Kontostavlou

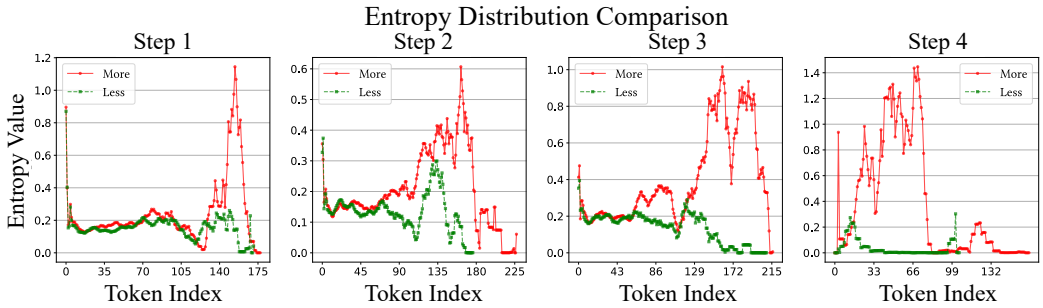

Figure 1: The entropy distribution of tool-integrated reasoning inference. "Step" refers to the reasoning steps—from the model starting inference or receiving the result of the previous tool call, up until the next tool call is triggered or the final answer is reached—including the process of triggering the tool call (but not including the result of the tool call). Step 1-4 thus represents the process of intermediate reasoning in the first, second, third, and fourth segments. "Token Index" represents the position of a token within that step. For example, $index_i$ represents the subscript index of the i-th output token in a certain step.

& Drigas, 2021), focusing on the model's knowledge boundaries. Self-DC framework (Wang et al., 2025b) controls the model's self-behavior by leveraging internal signals. ToolPrefer-LLaMA (Chen et al., 2025c) uses the DPO algorithm to update LLM strategies. Additionally, some works efficiently complete the TIR task by optimizing the reinforcement learning training process. For instance, Search Wisely (Wu et al., 2025b) and OTC (Wang et al., 2025a) carefully designed reward functions during training, while CoRT (Li et al., 2025a) meticulously optimized the training process. However, most existing work focuses solely on reducing the excessive tool calls and does not explore the accuracy of tool calls and the reasoning process within TIR. We consider that the effectiveness of TIR includes not only reducing redundant tool calls, but also invoking necessary tools. Moreover, it also involves minimizing overthinking during the reasoning process and avoiding analysis paralysis. In this paper, we do not limit our discussion to reducing redundant tool calls, but explore the effectiveness of TIR in a broader range of scenarios.

Self-Evolved Methods in LLMs. Due to the variability and complexity of world knowledge, models need to continuously evolve to better adapt to the real-world tasks (Wu et al., 2025a; Qiao et al., 2025; Dong et al., 2025c; Li et al., 2025d; Tao et al., 2025). The self-evolution strategy refers to a paradigm for dynamically improving model output performance through continuous learning and adaptation (ang Gao et al., 2025). It can be divided into two methods: parameter-updating and in-context learning. The former involves having the model autonomously generate training data to fine-tune weights, or using the experience from interacting with the environment to drive model training(Zhou et al., 2025; Simonds et al., 2025; Zeng et al., 2025; Su et al., 2025b). The latter achieves model self-evolution through the optimization of prompt words or memory mechanisms(Chhikara et al., 2025; Yüksekgönül et al., 2024). Referring to existing research, the core of Tool-Light lies in enabling the model to learn from data generated by itself(Li et al., 2025g). At the same time, it continuously generates better training data to perform parameter-updating self-evolution.

3 PRELIMINARIES

(1) Problem Definition. Multi-TIR aims to enable the model to autonomously call multiple tools, thus completing different complex tasks. Specifically, given the input instruction $\mathcal{I}$ and the model θ, the final output answer y of multi-TIR can be expressed in the following form:

$$y \sim P_\theta(y \mid R_N) \cdot \prod_{i=1}^{N} P_\theta(R_i \mid T_{C<i}, R_{<i}, \mathcal{I}), \tag{1}$$

where i represents the reasoning step, N represents the total number of reasoning steps, $R_{<i}$, $T_{C<i}$ and R_i represent the reasoning content before the i-th step, the tool call result before the i-th step,

and the reasoning content of the i-th step, respectively. For the tool T, it may contain a variety of different tools in multi-TIR tasks.

(2) Pre-Experiment. In this section, we explore the relationship between the TIR path and the distribution of information entropy. First, we present the formula for calculating the information entropy distribution:

$$H(i) = -\sum_{j=1}^{N} P(y_{ji}|y_{<i}) log P(y_{ji}|y_{<i}), \tag{2}$$

where N represents the vocabulary length at the i-th position, $y_{<i}$ and y_{ji} represent the sequence before position i and the j-th token at position i, respectively. We use the Search-R1 (Jin et al., 2025a) model to perform TIR on multiple QA datasets. For each sample, we rollout ten chains and classify them into two groups based on the tool calls: more tool calls **(more)** and fewer tool calls **(less)**. We record the average entropy distribution for each step. Our results can be found in Figure 1.

From the results of this experiment, we can draw some conclusions:

- When the model receives a tool call result, its output information entropy initially rises, then fluctuates, and finally drops sharply before the next tool call arrives.
- For the same sample, low-entropy chains tend to involve fewer tool calls, and as reasoning progresses, tool calls' difference between high and low entropy chains becomes increasingly evident.

These conclusions guide our subsequent data sampling.

(3) Tool Design Inspired by the existing work, we choose the two most mainstream tools, the code interpreter tool and the search tool, for exploration (Dong et al., 2025a; Wang et al., 2025a; Li et al., 2025h; Jin et al., 2025a). The code interpreter tool receives a piece of code, compiles it, and returns the compilation result or error message. The search tool receives a query, and then retrieves the content most relevant to the query from the local knowledge base or web page and returns it.

4 METHOD

Overview. We propose Tool-Light, a multi-stage training pipeline aiming to improve the effectiveness of model tool calls. As shown in Figures 2 and 3, Tool-Light consists of two key components: **(1) Dataset construction**, which includes carefully designed sampling strategies to screen out training data. **(2) Two-stage TIR training paradigm**, which trains the model successively with SFT and self-evolved DPO training. In the self-evolved DPO training stage, we design pre-aligned DPO training and self-evolved DPO alignment stages to gradually improve the model's capabilities.

4.1 DATASET CONSTRUCTION

In this section, a detailed introduction to the data construction and sampling strategies is provided.

Source Data Construction. In this section, we first obtain the source data for subsequent sampling. To improve the model's ability to execute TIR tasks, we adopt the SFT data from existing work, defined as D_{sft} (Dong et al., 2025a). We use D_{sft} to train the instruct model, resulting in the model M_{sft}. Then we use the instruct model to perform inference on D_{sft}. During this process, we do not offer the model external tools. After that, we only retain the data where the model's inference answers are incorrect, resulting in D_{source}. This process can be modeled as:

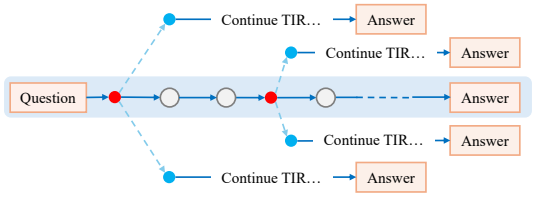

Figure 2: The overall process of entropy-guided sampling. Gray and red positions represent tool calls and the branching positions, respectively.

$$D_{\text{source}} = D_{\text{sft}}[y_{\text{label}} \neq y_{\text{inference}}], \tag{3}$$

where y_{label} and $y_{\text{inference}}$ are the golden answer and the result obtained from direct inference on that sample, respectively. $y_{\text{label}} \neq y_{\text{inference}}$ means that we only retain data where the model's inference

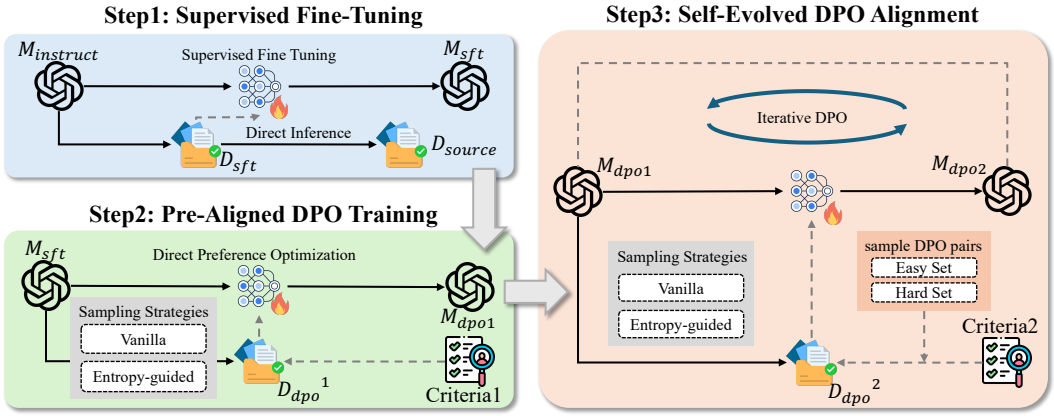

Figure 3: The overall structure of Tool-Light's training pipeline. Among them, the Self-Evolved DPO Alignment stage will conduct multiple rounds of training.

result is incorrect. We consider these data to be more challenging, and training on them can help improve the model's reasoning capabilities.

Sampling Strategy Design. After obtaining D_{source}, we use M_{sft} to perform TIR inference on it. For each question, we sample multiple TIR paths. The final set of all inference paths is denoted as D_{dpo}. For this step, we carefully design two sampling strategies, and mix the data from both strategies to construct high-quality training data. The specific strategies are described as follows:

(1) Vanilla TIR Sampling. We guide the model to generate multiple paths for each sample. This helps us construct positive-negative pairs. This process can be expressed as follows:

$$D_{\text{dpo}}^{1} = \{y | y = M_{\text{sft}}\left[I(q)\right], q \sim D_{\text{source}}\}, \tag{4}$$

where D_{dpo}^{1} represents the sampling results generated under vanilla sampling, I, q and y represent the prompt used, each question in D_{source}, and the output result from M_{sft}, respectively.

(2) Entropy-Guided Sampling. Vanilla sampling strategy incurs high inference costs and inference uncertainty. In this section, as shown in Figure 2, we propose an entropy-guided sampling strategy.

- **Entropy Distribution Calculation.** We begin by generating a primary reasoning chain, denoted as C_{main}. Each segment of the chain is considered a step. According to existing research (Liao et al., 2025), the initial inference step has a significant impact on the subsequent outputs. Therefore, for each step, we compute the entropy distribution at each token position in the first 10, 20, 30, 40, and 50 tokens. The average information entropy distribution for a subsequence is:

$$H_{\text{avg}}(i) = \frac{1}{i}\sum_{j=1}^{i} H(j), \tag{5}$$

  where $H_{\text{avg}}(i)$ represents top-i tokens' average entropy distribution. We retain the maximum $H_{\text{avg}}(i)$ and its corresponding sequence length i at each step.

- **Branch Sampling Execution.** We pick the top-k steps with the highest entropy, along with their sampling positions. At each position, we guide the model to continue generating several branches. Since those subsequences have higher information entropy, based on previous research (Wang et al., 2025c; Cui et al., 2025; Cheng et al., 2025), it is more likely to produce branch outputs with higher diversity. This process can be modeled as follows:

$$D_{\text{dpo}}^{2} = \{y | y_{>i} = M_{\text{sft}}[I(q) \oplus y_{<i}], q \sim D_{\text{source}}\}, \tag{6}$$

  where D_{dpo}^{2} and $y_{>i}$ represent the dataset guided by entropy sampling and the subsequent sampling output, respectively. $I(q) \oplus y_{<i}$ represents concatenating the prompt and the existing sequence together as the input. y represents concatenating $y_{>i}$ and $y_{<i}$ together. Assuming m

rollout times and an average sequence length of n, this method, in the ideal scenario, reduces computational complexity from $O(mn)$ to $O(n \log m)$. We collect all the main chains and branch chains, and construct the set of outputs based on tree-structured sampling.

Finally, D^1_{dpo} and D^2_{dpo} are mixed in a certain ratio to obtain the final dataset ${D_{\text{dpo}}}^1$ for the positive-negative pairs construction.

4.2 TWO-STAGE TIR TRAINING PARADIGM

Based on existing research (Li et al., 2025g; Dong et al., 2025a; Song et al., 2025), we propose a two-stage self-evolved training pipeline to gradually boost the effectiveness and stability of the model's TIR process. The specific pipeline is shown in Figure 3.

Supervised Fine-Tuning. First, we obtain the SFT model M_{sft} with the same setup as in ToolStar (Dong et al., 2025a). The SFT's loss function is: $\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{(x,y)\in D} \log P_\theta(y|x)$. As shown in the first step of Figure 3, this step aims to help the model quickly acquire the ability to complete TIR tasks.

Self-Evolved DPO. Inspired by existing works (Li et al., 2025g; Wu et al., 2023; Tao et al., 2024), we further fine-tune M_{sft} through a self-evolved DPO strategy, using a loss function as follows:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x,y_w,y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)}\right)\right], \tag{7}$$

where π_{ref} is the original strategy model, and β is the hyperparameter and σ is the sigmoid function. The primary objective of DPO is to maximize the relative probability of the model generating y_w compared to y_l. This step aims to endow the model with the capability to finish TIR tasks accurately and efficiently. First, we establish a specialized data sampling criterion Cri_1. It is used to obtain training data ${D_{\text{dpo}}}^1$ from M_{sft} using both vanilla and entropy-guided strategies. We then train M_{sft} on ${D_{\text{dpo}}}^1$ to yield M_{dpo1}. Subsequently, a new criterion Cri_2 is defined, guiding M_{dpo1} to sample the next iteration of training data ${D_{\text{dpo}}}^2$. We train M_{dpo1} on this dataset and get M_{dpo2}. After that, we update $M_{\text{dpo1}} \leftarrow M_{\text{dpo2}}$ and iteratively repeat the sampling and training steps until performance converges. Details on these two criteria follow in the subsequent sections.

(1) Pre-Aligned DPO Training. In each sample, we first categorize the correct or incorrect trajectories based on the F1 score. Only trajectories with 1 F1 score are considered correct, and only trajectories with 0 F1 score are considered incorrect. Then, we divide the samples into a hard set and an easy set. After that, referring to the conclusions in Section 3, we design criteria for sampling positive and negative pairs in the entropy-guided strategy:

- **Hard set:** Samples with $\leq 50\%$ correct trajectories.
- **Easy Set:** Samples with $\geq 50\%$ correct trajectories.
- **Positive Example:** Correct trajectory with the minimal number of tool calls and lowest entropy. If no correct trajectory exists, the corresponding SFT trajectory from D_{source} is used.
- **Negative Example:** An incorrect trajectory with more tool calls than the positive example.

For vanilla sampling criteria, we choose samples with $\leq 40\%$ correct trajectories as the hard set, and those with $\geq 70\%$ correct trajectories as the easy set. For each set, the shortest correct trajectory is the positive example, and the incorrect trajectory longer than the positive one is the negative example.

We set the hard set and easy set's data ratio as 2:1. The goal of training in this round is to teach the model to reduce unnecessary tool calls while avoiding excessive reasoning.

(2) Self-Evolved DPO Alignment. We use M_{dpo1} to resample from ${D_{\text{dpo}}}^1$. At the same time, we introduce the second criterion to filter out ${D_{dpo}}^2$. In Cri_2, samples whose number of correct trajectories is less than half of the incorrect ones are classified into the hard set, while the criteria for the easy set remain unchanged. For each sample generated by M_{dpo1}, if it is classified into the easy set, it means that after pre-aligned DPO training, M_{dpo1} has mastered the knowledge to solve this

problem. If it is classified into the hard set, it means that this problem is still difficult for M_{dpo1}. For these two different situations, we have also designed the sampling criteria for positive and negative examples:

- **Negative Example in Easy Set:** Incorrect trajectory with the most tool calls.
- **Positive Example in Easy Set:** Correct trajectory with fewer tool calls and lowest entropy.
- **Positive Example in Hard Set:** Correct trajectory with the longest reasoning chain.
- **Negative Example in Hard Set:** Incorrect trajectory with the shortest reasoning chain.

Based on preliminary experiments, low-entropy outputs' tool calls are often fewer. Therefore, we also incorporate the consideration of low entropy into positive examples' selection. We use the sampled D_{dpo}[2] to continue training M_{dpo1}, and continuously sample with the trained model. Self-evolved DPO alignment will continue for several loops. This step is intended to equip the model with the ability to make necessary tool calls while maintaining its original efficient TIR capability. After several rounds of training, we obtain the final model M_{dpo2}.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Dataset. We carefully select 10 datasets and categorize them into two types: **(1) mathematical-reasoning tasks**, including AIME24, AIME25 [1], AMC23 [2], MATH (Hendrycks et al., 2021), MATH500 (Lightman et al., 2024), and GSM8K (Cobbe et al., 2021), and **(2) knowledge-intensive tasks**, including HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2023).

Metrics. For mathematical-reasoning tasks, we adopt the LLM-as-Judge approach (Gu et al., 2024), utilizing the Qwen2.5-72B-Instruct model to evaluate answer correctness. For knowledge-intensive tasks, we use F1 score to evaluate answers. Additionally, we introduce two metrics: **Efficiency** and **Necessity**. **(1) Efficiency** is defined as $\text{Effi} = \frac{1}{n}\sum_{i=1}^{n}\frac{M_i}{T_i}$, where n is the total number of samples, M_i the performance on the i-th sample, and T_i the number of tool calls. It captures the model's tendency to overuse tools. **(2) Necessity** is defined as $\text{Nece} = \mathcal{M}\left(\frac{1}{n}\sum_{i=1}^{n}(N_{in}^{i} - N_{co}^{i})\right)$, where N_{in}^{i} is the count of methods for the i-th sample that make more tool calls than the current method but yield incorrect answers, N_{co}^{i} is the count of methods with fewer tool calls yet correct answers, and $\mathcal{M}$ denotes Min-Max Scaling. It reflects the model's tendency to underuse tools.

Baselines. We classify baseline methods into two types: **(1) Single-Tool-Integrated Reasoning**, including Search-o1 (Li et al., 2025f), Search-R1 (Jin et al., 2025a), DotaMath (Li et al., 2024a), ToRL (Li et al., 2025h), and ReTool (Feng et al., 2025b), and **(2) Multi-Tool-Integrated Reasoning**, including prompt engineering, ReCall (Chen et al., 2025a), and ToolStar (Dong et al., 2025a).[3]

5.2 MAIN RESULTS

Our main results are presented in Table 1, and more details can be found in Appendix 3. We can draw the following insights:

(1) Proper use of external tools can be of great help to model reasoning. For example, when external tools are introduced, the prompting-based method on the Qwen model has significant advantages over the direct inference method in mathematical reasoning tasks. However, for knowledge-intensive tasks, the effect of introducing external tools is reduced (only improved on MuSiQue dataset). This also indicates that untrained models cannot use external tools well.

(2) While training enhances the model's TIR capabilities, single-tool training hinders the development of generalization abilities. Compared to using prompt engineering methods, fine-tuning

[1] https://huggingface.co/datasets/AI-MO/aimo-validation-aime

[2] https://huggingface.co/datasets/zwhe99/amc23

[3] More implementation details can be found in Appendix B.

Table 1: Results on 10 reasoning tasks, with top two results highlighted in **bold** and underlined. Unless noted, Qwen2.5-7B-Instruct is used as the backbone. For ToRL and ReTool, their RL models generate training data inferences for SFT, which serves as the baseline. Abbreviation: 2Wiki. (2WikiMultiHopQA). HQA. (HotpotQA). MSQ. (MuSiQue). Bamb. (Bamboogle).

Method	Mathematical-Reasoning Tasks						Knowledge-Intensive Tasks				Avg.
	AIME24	AIME25	AMC23	MATH	MATH500	GSM8K	HQA	2Wiki.	MSQ	Bamb	
Direct Inference											
Qwen2.5-7B-Instruct	0.0	6.7	30.0	68.6	57.2	71.4	26.1	25.6	7.9	36.5	33.0
Llama3.1-8B-Instruct	0.0	3.3	15.0	52.8	33.4	75.0	16.2	13.7	7.4	23.2	24.0
Single-TIR Methods											
Search-o1	6.7	10.0	37.5	73.6	61.8	80.2	41.1	35.4	13.2	39.8	39.9
Search-R1	16.7	6.7	45.0	81.2	63.8	82.4	48.7	40.0	24.1	47.4	45.6
DotaMath	16.7	10.0	50.0	74.6	62.2	82.6	26.2	21.7	6.5	28.6	37.9
ToRL	30.0	26.7	**67.5**	87.0	**80.2**	89.2	41.3	35.4	9.5	36.9	50.4
ReTool	23.3	**30.0**	62.5	84.8	78.4	86.2	31.5	29.0	11.1	35.8	47.3
Multi-TIR Methods											
Prompting-Based	6.7	13.3	47.5	73.8	62.2	69.4	21.1	23.8	9.9	25.5	35.3
ReCall	3.3	6.7	27.5	73.2	54.6	79.8	51.9	54.0	**25.0**	55.5	43.2
Tool-Star	30.0	26.7	65.0	85.6	77.2	89.4	54.7	55.7	22.8	**58.8**	56.6
Ours											
Tool-Light (Llama)	10.0	6.7	30.0	59.4	56.8	76.6	41.3	33.5	12.2	41.3	36.8
Tool-Light (Qwen)	**33.3**	23.3	**67.5**	**87.4**	79.0	**92.0**	**57.7**	**56.1**	**25.0**	58.7	**58.0**

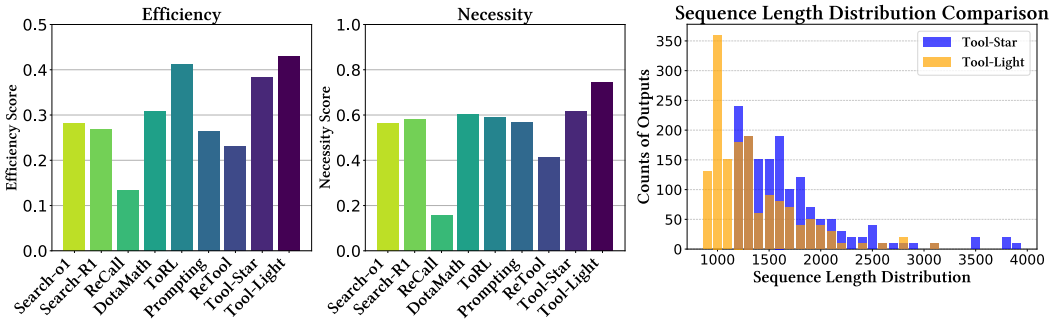

Figure 4: Efficiency, Necessity, and Sequence Length differences between Tool Light and baselines.

can effectively teach the model to use tools. For example, Search-R1 outperforms Search-o1 by 5.7 on average, while Tool-Star outperforms prompting-based methods by 21.3 on average. It is worth noting that models trained for single-TIR tend to have poor generalization performance. For example, Search-R1 performs well on knowledge-intensive tasks but poorly on mathematical-reasoning tasks, while ToRL shows the opposite pattern. The Tool-Star and Tool-Light frameworks perform well on both types of tasks, highlighting the generalization advantages of multi-TIR training.

(3) Multi-round DPO training allows for further enhancement of the model's performance. Compared with the simple SFT method, Tool-Light framework additionally introduces two operations: Pre-Aligned DPO Training and Self-Evolved DPO Alignment. This leads to an overall improvement in the model's performance. For example, in mathematical-reasoning tasks, Tool-Light achieves optimal performance on four datasets, and ranks among the top two on all datasets for Knowledge-Intensive tasks. Notably, by solely adopting the DPO method, we outperform most baselines that are trained using GRPO when it comes to average performance. This fully demonstrates the effectiveness of our training pipeline design.

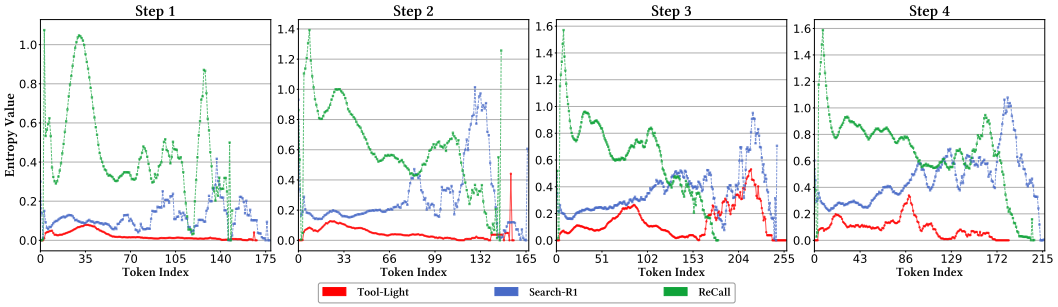

Figure 5: Output sequences' entropy distribution under different methods.

5.3 Quantitative Analysis

Tool-Use Effectiveness Analysis In this section, we analyze the differences in TIR effectiveness between Tool-Light and other baseline methods. As shown in Figure 4, Tool-Light achieves the best performance in Efficiency and Necessity metrics, which indicates the superiority of our designed training process. Regarding the output sequence length metric, Tool-Light has a shorter length distribution than Tool-Star, and outperforms Tool-Star in both result correctness and reasoning effectiveness. It shows that our training method can effectively reduce the phenomenon of overthinking. Detailed cases are list in Appendix D.

Entropy Distribution Analysis As described in Section 3, in this section, we explore the impact of introducing the entropy-guided sampling strategy on final output results' information entropy distribution. The results are shown in Figure 5. The existing TIR models tend to produce output sequences with relatively high entropy. In contrast, Tool-Light results in lower entropy distributions for its output sequences. This might be due to the consideration of information entropy during sampling. Our training method introduces the learning of low-entropy path outputs. Since high-entropy tokens can sometimes trigger excessive reasoning in the model, our method effectively mitigates the phenomenon of overthinking. This also aligns with the results in Figure 4. [4]

Impact of DPO Loop Number In this section, we investigate how the number of training loops in the self-evolved DPO alignment stage influences the model's performance. For this stage, we increase the loop count from 1 to 5 and record the Performance, Efficiency, and Necessity metrics. The results are shown in Table 2's 1-5 rows. As the number of training loop increases, all metrics reach optimal values in the second training round and then decline. This may be because the self-evolved method is initially able to obtain sufficient positive-negative pairs. However, as the evolution proceeds, there are fewer sample pairs beneficial for training, and the model may overfit to the distribution of the training set, leading to lower performance on the test set.

Table 2: Ablation experiment results for various aspects: *1/1 strategy ratio* indicates a 1:1 data ratio for two sampling strategies, *p-r.* and *n-r.* denote random positive and negative example selection.

Method	Performance	Efficiency	Necessity
Tool-Light (*2 loop*)	58.0	0.44	0.75
Ablation for self-evolved Loops			
w. 1 loop	57.9 (-0.1)	0.42 (-0.02)	0.71 (-0.04)
w. 3 loop	56.1 (-1.9)	0.39 (-0.05)	0.73 (-0.02)
w. 4 loop	56.4 (-1.6)	0.37 (-0.07)	0.71 (-0.04)
w. 5 loop	54.1 (-3.9)	0.36 (-0.08)	0.72 (-0.03)
Ablation for Sampling Criteria			
w. 1/1 data ratio	56.9 (-1.1)	0.44	0.76 (+0.01)
w. p-r.	53.6 (-4.4)	0.42 (-0.02)	0.63 (-0.12)
w. n-r.	53.9 (-4.1)	0.41 (-0.03)	0.74 (-0.01)

Impact of Sampling Criteria In this section, we explore the impact of sampling criteria on the training effect. We modify the original criteria from the following aspects: the data ratio of two strategies is 1:1, positive examples are

[4] We also explore the impact of different data ratios on the final training results during sampling. The experimental results and analyses are provided in Appendix C.

randomly selected, and negative examples are randomly selected. The results are shown in rows 6 - 8 of Table 2. We can find that the different data ratio has the least impact on the training effect, while the change in the positive-negative example selection criteria has a greater impact on the results. This indicates that for the DPO training in Tool-Light, it is very important to fully distinguish between positive and negative examples.

6 CONCLUSION

In this paper, we propose Tool-Light, a framework empowering models to efficiently and accurately complete TIR tasks. We first analyze the TIR process from the perspective of information entropy. Based on this analysis, we introduce Tool-Light, a framework improving model's TIR capabilities from both data construction and training aspects. Within this framework, we design a new entropy-guided sampling strategy. Based on this method, we construct a two-stage TIR training paradigm. The trained model achieves excellent results in terms of overall performance, tool calls effectiveness, and simplicity of the inference process. We confirm that our work can provide more assistance for future research on TIR.

7 ACKNOWLEDGEMENT

This work was supported by the National Natural Science Foundation of China No.62272467. The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

8 ETHICS STATEMENT

All authors of this work have strictly adhered to the ICLR Code of Ethics. It does not involve any research with human subjects, practices to data set releases, potentially harmful insights, methodologies and applications, potential conflicts of interest and sponsorship, discrimination/bias/fairness concerns, privacy and security issues, legal compliance, and research integrity issues.

9 REPRODUCIBILITY STATEMENT

To ensure the reproducibility of Tool-Light's experiments, we detail all the experimental settings in Appendix B. Among them, the data ratio based on the difficulty level of data sampling, as well as the data ratio between the vanilla sampling method and the entropy-guided sampling method, are respectively provided in Section 4.2 and Appendix C.

To further facilitate reproducibility, we provide detailed code in the Supplementary Material, including the entropy-guided sampling, SFT and DPO. Upon acceptance, we will make our experimental data and model checkpoints publicly available.

REFERENCES

Huan ang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Yiran Wu, Hongru Wang, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Qihan Ren, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: On path to artificial super intelligence, 2025. URL https://arxiv.org/abs/2507.21046.

Fei Bai, Yingqian Min, Beichen Zhang, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Towards effective code-integrated reasoning. *CoRR*, abs/2505.24480, 2025. doi: 10.48550/ARXIV.2505.24480. URL https://doi.org/10.48550/arXiv.2505.24480.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. Research: Learning

to reason with search for llms via reinforcement learning. *CoRR*, abs/2503.19470, 2025a. doi: 10.48550/ARXIV.2503.19470. URL https://doi.org/10.48550/arXiv.2503.19470.

Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *CoRR*, abs/2503.09567, 2025b. doi: 10.48550/ARXIV.2503.09567. URL https://doi.org/10.48550/arXiv.2503.09567.

Sijia Chen, Yibo Wang, Yi-Feng Wu, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Lijun Zhang. Advancing tool-augmented large language models: Integrating insights from errors in inference trees, 2025c. URL https://arxiv.org/abs/2406.07115.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. Reasoning with exploration: An entropy perspective, 2025. URL https://arxiv.org/abs/2506.14758.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. Mem0: Building production-ready AI agents with scalable long-term memory. *CoRR*, abs/2504.19413, 2025. doi: 10.48550/ARXIV.2504.19413. URL https://doi.org/10.48550/arXiv.2504.19413.

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

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, Zhiyuan Liu, Hao Peng, Lei Bai, Wanli Ouyang, Yu Cheng, Bowen Zhou, and Ning Ding. The entropy mechanism of reinforcement learning for reasoning language models. *CoRR*, abs/2505.22617, 2025. doi: 10.48550/ARXIV.2505.22617. URL https://doi.org/10.48550/arXiv.2505.22617.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *CoRR*, abs/2505.16410, 2025a. doi: 10.48550/ARXIV.2505.16410. URL https://doi.org/10.48550/arXiv.2505.16410.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, Guorui Zhou, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. Agentic reinforced policy optimization. *CoRR*, abs/2507.19849, 2025b. doi: 10.48550/ARXIV.2507.19849. URL https://doi.org/10.48550/arXiv.2507.19849.

Guanting Dong, Yutao Zhu, Chenghao Zhang, Zechen Wang, Ji-Rong Wen, and Zhicheng Dou. Understand what LLM needs: Dual preference alignment for retrieval-augmented generation. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov (eds.), *Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025- 2 May 2025*, pp. 4206–4225. ACM, 2025c. doi: 10.1145/3696410.3714717. URL https://doi.org/10.1145/3696410.3714717.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, et al. Agent-world: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, Shibin Wu, Zhengwei Tao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Towards general agentic intelligence via environment scaling, 2025. URL https://arxiv.org/abs/2509.13311.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms, 2025a. URL https://arxiv.org/abs/2504.11536.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. *CoRR*, abs/2504.11536, 2025b. doi: 10.48550/ARXIV.2504.11536. URL https://doi.org/10.48550/arXiv.2504.11536.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Minlie Huang, Nan Duan, and Weizhu Chen. Tora: A tool-integrated reasoning agent for mathematical problem solving. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=Ep0TtjVoap.

Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Yuanzhuo Wang, and Jian Guo. A survey on llm-as-a-judge. *CoRR*, abs/2411.15594, 2024. doi: 10.48550/ARXIV.2411.15594. URL https://doi.org/10.48550/arXiv.2411.15594.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In Joaquin Vanschoren and Sai-Kit Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/be83ab3ecd0db773eb2dc1b0a17836a1-Abstract-round2.html.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps. In Donia Scott, Núria Bel, and Chengqing Zong (eds.), *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pp. 6609–6625. International Committee on Computational Linguistics, 2020. doi: 10.18653/V1/2020.COLING-MAIN.580. URL https://doi.org/10.18653/v1/2020.coling-main.580.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jie Huang and Kevin Chen-Chuan Chang. Towards reasoning in large language models: A survey. In Anna Rogers and Jordan L. Boyd-Graber and (eds.), *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pp. 1049–1065. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-ACL.67. URL https://doi.org/10.18653/v1/2023.findings-acl.67.

Ziyang Huang, Xiaowei Yuan, Yiming Ju, Jun Zhao, and Kang Liu. Reinforced internal-external knowledge synergistic reasoning for efficient adaptive search agent. *CoRR*, abs/2505.07596, 2025. doi: 10.48550/ARXIV.2505.07596. URL https://doi.org/10.48550/arXiv.2505.07596.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *CoRR*, abs/2503.09516, 2025a. doi: 10.48550/ARXIV.2503.09516. URL https://doi.org/10.48550/arXiv.2503.09516.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In Guodong Long, Michale Blumestein, Yi Chang, Liane Lewin-Eytan, Zi Helen Huang, and Elad Yom-Tov (eds.), *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pp. 737–740. ACM, 2025b. doi: 10.1145/3701716.3715313. URL https://doi.org/10.1145/3701716.3715313.

Eirini-Zoi Kontostavlou and Athanasios Drigas. How metacognition supports giftedness in leadership A review of contemporary literature. *Int. J. Adv. Corp. Learn.*, 14(2):4–16, 2021. doi: 10.3991/IJAC.V14I2.23237. URL https://doi.org/10.3991/ijac.v14i2.23237.

Chengpeng Li, Guanting Dong, Mingfeng Xue, Ru Peng, Xiang Wang, and Dayiheng Liu. Dotamath: Decomposition of thought with code assistance and self-correction for mathematical reasoning. *CoRR*, abs/2407.04078, 2024a. doi: 10.48550/ARXIV.2407.04078. URL https://doi.org/10.48550/arXiv.2407.04078.

Chengpeng Li, Zheng Yuan, Hongyi Yuan, Guanting Dong, Keming Lu, Jiancan Wu, Chuanqi Tan, Xiang Wang, and Chang Zhou. Mugglemath: Assessing the impact of query and response augmentation on math reasoning. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pp. 10230–10258. Association for Computational Linguistics, 2024b. doi: 10.18653/V1/2024.ACL-LONG.551. URL https://doi.org/10.18653/v1/2024.acl-long.551.

Chengpeng Li, Zhengyang Tang, Ziniu Li, Mingfeng Xue, Keqin Bao, Tian Ding, Ruoyu Sun, Benyou Wang, Xiang Wang, Junyang Lin, and Dayiheng Liu. Cort: Code-integrated reasoning within thinking, 2025a. URL https://arxiv.org/abs/2506.09820.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. START: self-taught reasoner with tools. *CoRR*, abs/2503.04625, 2025b. doi: 10.48550/ARXIV.2503.04625. URL https://doi.org/10.48550/arXiv.2503.04625.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Rui Ye, Yida Zhao, Liwen Zhang, Litu Ou, Dingchu Zhang, Xixi Wu, Jialong Wu, Xinyu Wang, Zile Qiao, Zhen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor-v2: Bridging the chasm to proprietary agents via synthetic data and scalable reinforcement learning, 2025c. URL https://arxiv.org/abs/2509.13305.

Kuan Li, Zhongwang Zhang, Huifeng Yin, Liwen Zhang, Litu Ou, Jialong Wu, Wenbiao Yin, Baixuan Li, Zhengwei Tao, Xinyu Wang, Weizhou Shen, Junkai Zhang, Dingchu Zhang, Xixi Wu, Yong Jiang, Ming Yan, Pengjun Xie, Fei Huang, and Jingren Zhou. Websailor: Navigating superhuman reasoning for web agent. *CoRR*, abs/2507.02592, 2025d. doi: 10.48550/ARXIV.2507.02592. URL https://doi.org/10.48550/arXiv.2507.02592.

Weizhen Li, Jianbo Lin, Zhuosong Jiang, Jingyi Cao, Xinpeng Liu, Jiayu Zhang, Zhenqiang Huang, Qianben Chen, Weichen Sun, Qiexiang Wang, Hongxuan Lu, Tianrui Qin, Chenghao Zhu, Yi Yao, Shuying Fan, Xiaowan Li, Tiannan Wang, Pai Liu, King Zhu, He Zhu, Dingfeng Shi, Piaohong Wang, Yeyi Guan, Xiangru Tang, Minghao Liu, Yuchen Eleanor Jiang, Jian Yang, Jiaheng Liu, Ge Zhang, and Wangchunshu Zhou. Chain-of-agents: End-to-end agent foundation models via multi-agent distillation and agentic RL. *CoRR*, abs/2508.13167, 2025e. doi: 10.48550/ARXIV.2508.13167. URL https://doi.org/10.48550/arXiv.2508.13167.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. Search-o1: Agentic search-enhanced large reasoning models. *CoRR*, abs/2501.05366, 2025f. doi: 10.48550/ARXIV.2501.05366. URL https://doi.org/10.48550/arXiv.2501.05366.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yutao Zhu, Yongkang Wu, Ji-Rong Wen, and Zhicheng Dou. Webthinker: Empowering large reasoning models with deep research capability. *CoRR*, abs/2504.21776, 2025g. doi: 10.48550/ARXIV.2504.21776. URL https://doi.org/10.48550/arXiv.2504.21776.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated RL. *CoRR*, abs/2503.23383, 2025h. doi: 10.48550/ARXIV.2503.23383. URL https://doi.org/10.48550/arXiv.2503.23383.

Zhong-Zhi Li, Duzhen Zhang, Ming-Liang Zhang, Jiaxin Zhang, Zengyan Liu, Yuxuan Yao, Haotian Xu, Junhao Zheng, Pei-Jie Wang, Xiuyi Chen, Yingying Zhang, Fei Yin, Jiahua Dong, Zhijiang Guo, Le Song, and Cheng-Lin Liu. From system 1 to system 2: A survey of reasoning large language models. *CoRR*, abs/2502.17419, 2025i. doi: 10.48550/ARXIV.2502.17419. URL https://doi.org/10.48550/arXiv.2502.17419.

Zijian Li, Xin Guan, Bo Zhang, Shen Huang, Houquan Zhou, Shaopeng Lai, Ming Yan, Yong Jiang, Pengjun Xie, Fei Huang, Jun Zhang, and Jingren Zhou. Webweaver: Structuring web-scale evidence with dynamic outlines for open-ended deep research, 2025j. URL https://arxiv.org/abs/2509.13312.

Zongjie Li and Shuai Wang. Reasoning as a resource: Optimizing fast and slow thinking in code generation models, 2025. URL https://arxiv.org/abs/2506.09396.

Baohao Liao, Xinyi Chen, Sara Rajaee, Yuhui Xu, Christian Herold, Anders Søgaard, Maarten de Rijke, and Christof Monz. Lost at the beginning of reasoning. *CoRR*, abs/2506.22058, 2025. doi: 10.48550/ARXIV.2506.22058. URL https://doi.org/10.48550/arXiv.2506.22058.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=v8L0pN6EOi.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models, 2023. URL https://arxiv.org/abs/2304.09842.

Yingqian Min, Zhipeng Chen, Jinhao Jiang, Jie Chen, Jia Deng, Yiwen Hu, Yiru Tang, Jiapeng Wang, Xiaoxue Cheng, Huatong Song, Wayne Xin Zhao, Zheng Liu, Zhongyuan Wang, and Ji-Rong Wen. Imitate, explore, and self-improve: A reproduction report on slow-thinking reasoning systems. *CoRR*, abs/2412.09413, 2024. doi: 10.48550/ARXIV.2412.09413. URL https://doi.org/10.48550/arXiv.2412.09413.

Bhargavi Paranjape, Scott M. Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Túlio Ribeiro. ART: automatic multi-step reasoning and tool-use for large language models. *CoRR*, abs/2303.09014, 2023. doi: 10.48550/ARXIV.2303.09014. URL https://doi.org/10.48550/arXiv.2303.09014.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pp. 5687–5711. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.FINDINGS-EMNLP.378. URL https://doi.org/10.18653/v1/2023.findings-emnlp.378.

Cheng Qian, Emre Can Acikgoz, Hongru Wang, Xiusi Chen, Avirup Sil, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. SMART: self-aware agent for tool overuse mitigation. *CoRR*, abs/2502.11435, 2025. doi: 10.48550/ARXIV.2502.11435. URL https://doi.org/10.48550/arXiv.2502.11435.

Zile Qiao, Guoxin Chen, Xuanzhong Chen, Donglei Yu, Wenbiao Yin, Xinyu Wang, Zhen Zhang, Baixuan Li, Huifeng Yin, Kuan Li, Rui Min, Minpeng Liao, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webresearcher: Unleashing unbounded reasoning capability in long-horizon agents, 2025. URL https://arxiv.org/abs/2509.13309.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and Pengfei Liu. O1 replication journey: A strategic progress report - part 1. *CoRR*, abs/2410.18982, 2024. doi: 10.48550/ARXIV.2410.18982. URL https://doi.org/10.48550/arXiv.2410.18982.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis, 2023. URL https://arxiv.org/abs/2307.16789.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. URL https://arxiv.org/abs/2302.04761.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300, 2024. doi: 10.48550/ARXIV.2402.03300. URL https://doi.org/10.48550/arXiv.2402.03300.

Toby Simonds, Kevin Lopez, Akira Yoshiyama, and Dominique Garmier. Self rewarding self improving. *CoRR*, abs/2505.08827, 2025. doi: 10.48550/ARXIV.2505.08827. URL https://doi.org/10.48550/arXiv.2505.08827.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *CoRR*, abs/2503.05592, 2025. doi: 10.48550/ARXIV.2503.05592. URL https://doi.org/10.48550/arXiv.2503.05592.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, Zile Qiao, Zhongwang Zhang, Huifeng Yin, Shihao Cai, Runnan Fang, Zhengwei Tao, Wenbiao Yin, Chenxiong Qian, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Scaling agents via continual pre-training, 2025a. URL https://arxiv.org/abs/2509.13310.

Xuerui Su, Shufang Xie, Guoqing Liu, Yingce Xia, Renqian Luo, Peiran Jin, Zhiming Ma, Yue Wang, Zun Wang, and Yuting Liu. Trust region preference approximation: A simple and stable reinforcement learning algorithm for LLM reasoning. *CoRR*, abs/2504.04524, 2025b. doi: 10.48550/ARXIV.2504.04524. URL https://doi.org/10.48550/arXiv.2504.04524.

Yang Sui, Yu-Neng Chuang, Guanchu Wang, Jiamu Zhang, Tianyi Zhang, Jiayi Yuan, Hongyi Liu, Andrew Wen, Shaochen Zhong, Hanjie Chen, and Xia Ben Hu. Stop overthinking: A survey on efficient reasoning for large language models. *CoRR*, abs/2503.16419, 2025. doi: 10.48550/ARXIV.2503.16419. URL https://doi.org/10.48550/arXiv.2503.16419.

Jiankai Sun, Chuanyang Zheng, Enze Xie, Zhengying Liu, Ruihang Chu, Jianing Qiu, Jiaqi Xu, Mingyu Ding, Hongyang Li, Mengzhe Geng, Yue Wu, Wenhai Wang, Junsong Chen, Zhangyue Yin, Xiaozhe Ren, Jie Fu, Junxian He, Wu Yuan, Qi Liu, Xihui Liu, Yu Li, Hao Dong, Yu Cheng, Ming Zhang, Pheng-Ann Heng, Jifeng Dai, Ping Luo, Jingdong Wang, Ji-Rong Wen, Xipeng Qiu, Yike Guo, Hui Xiong, Qun Liu, and Zhenguo Li. A survey of reasoning with foundation models. *CoRR*, abs/2312.11562, 2023. doi: 10.48550/ARXIV.2312.11562. URL https://doi.org/10.48550/arXiv.2312.11562.

Zhengwei Tao, Ting-En Lin, Xiancai Chen, Hangyu Li, Yuchuan Wu, Yongbin Li, Zhi Jin, Fei Huang, Dacheng Tao, and Jingren Zhou. A survey on self-evolution of large language models. *CoRR*, abs/2404.14387, 2024. doi: 10.48550/ARXIV.2404.14387. URL https://doi.org/10.48550/arXiv.2404.14387.

Zhengwei Tao, Jialong Wu, Wenbiao Yin, Junkai Zhang, Baixuan Li, Haiyang Shen, Kuan Li, Liwen Zhang, Xinyu Wang, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webshaper: Agentically data synthesizing via information-seeking formalization. *CoRR*, abs/2507.15061, 2025. doi: 10.48550/ARXIV.2507.15061. URL https://doi.org/10.48550/arXiv.2507.15061.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, Chuning Tang, Congcong Wang, Dehao Zhang, Enming Yuan, Enzhe Lu, Fengxiang Tang, Flood Sung, Guangda Wei, Guokun Lai, Haiqing Guo, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haotian Yao, Haotian Zhao, Haoyu Lu, Haoze Li, Haozhen Yu, Hongcheng Gao, Huabin Zheng, Huan Yuan, Jia Chen, Jianhang Guo, Jianlin Su, Jianzhou Wang, Jie Zhao, Jin Zhang, Jingyuan Liu, Junjie Yan, Junyan Wu, Lidong Shi, Ling Ye, Longhui Yu, Mengnan Dong, Neo Zhang, Ningchen Ma, Qiwei Pan, Qucheng Gong, Shaowei Liu, Shengling Ma, Shupeng Wei, Sihan Cao, Siying Huang, Tao Jiang, Weihao Gao, Weimin Xiong, Weiran He, Weixiao Huang, Wenhao Wu, Wenyang He, Xianghui Wei, Xianqing Jia, Xingzhe Wu, Xinran Xu, Xinxing Zu, Xinyu Zhou, Xuehai Pan, Y. Charles, Yang Li, Yangyang Hu, Yangyang Liu, Yanru Chen, Yejie Wang, Yibo Liu, Yidao Qin, Yifeng Liu, Ying Yang, Yiping Bao, Yulun Du, Yuxin Wu, Yuzhi Wang, Zaida Zhou, Zhaoji Wang, Zhaowei Li, Zhen Zhu, Zheng Zhang, Zhexu Wang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Ziyao Xu, and Zonghan

Yang. Kimi k1.5: Scaling reinforcement learning with llms. *CoRR*, abs/2501.12599, 2025. doi: 10.48550/ARXIV.2501.12599. URL https://doi.org/10.48550/arXiv.2501.12599.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Hongru Wang, Cheng Qian, Wanjun Zhong, Xiusi Chen, Jiahao Qiu, Shijue Huang, Bowen Jin, Mengdi Wang, Kam-Fai Wong, and Heng Ji. Acting less is reasoning more! teaching model to act efficiently, 2025a. URL https://arxiv.org/abs/2504.14870.

Hongru Wang, Boyang Xue, Baohang Zhou, Tianhua Zhang, Cunxiang Wang, Huimin Wang, Guanhua Chen, and Kam-Fai Wong. Self-dc: When to reason and when to act? self divide-and-conquer for compositional unknown questions. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 6510–6525. Association for Computational Linguistics, 2025b. doi: 10.18653/V1/2025.NAACL-LONG.331. URL https://doi.org/10.18653/v1/2025.naacl-long.331.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, abs/2212.03533, 2022. doi: 10.48550/ARXIV.2212.03533. URL https://doi.org/10.48550/arXiv.2212.03533.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, Yuqiong Liu, An Yang, Andrew Zhao, Yang Yue, Shiji Song, Bowen Yu, Gao Huang, and Junyang Lin. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for LLM reasoning. *CoRR*, abs/2506.01939, 2025c. doi: 10.48550/ARXIV.2506.01939. URL https://doi.org/10.48550/arXiv.2506.01939.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. RAGEN: understanding self-evolution in LLM agents via multi-turn reinforcement learning. *CoRR*, abs/2504.20073, 2025d. doi: 10.48550/ARXIV.2504.20073. URL https://doi.org/10.48550/arXiv.2504.20073.

Ziliang Wang, Xuhui Zheng, Kang An, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Stepsearch: Igniting llms search ability via step-wise proximal policy optimization. *CoRR*, abs/2505.15107, 2025e. doi: 10.48550/ARXIV.2505.15107. URL https://doi.org/10.48550/arXiv.2505.15107.

Jialong Wu, Baixuan Li, Runnan Fang, Wenbiao Yin, Liwen Zhang, Zhengwei Tao, Dingchu Zhang, Zekun Xi, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Webdancer: Towards autonomous information seeking agency. *CoRR*, abs/2505.22648, 2025a. doi: 10.48550/ARXIV.2505.22648. URL https://doi.org/10.48550/arXiv.2505.22648.

Peilin Wu, Mian Zhang, Xinlu Zhang, Xinya Du, and Zhiyu Zoey Chen. Search wisely: Mitigating sub-optimal agentic searches by reducing uncertainty. *CoRR*, abs/2505.17281, 2025b. doi: 10.48550/ARXIV.2505.17281. URL https://doi.org/10.48550/arXiv.2505.17281.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. Self-evolved diverse data sampling for efficient instruction tuning. *CoRR*, abs/2311.08182, 2023. doi: 10.48550/ARXIV.2311.08182. URL https://doi.org/10.48550/arXiv.2311.08182.

Tian Xie, Zitian Gao, Qingnan Ren, Haoming Luo, Yuqian Hong, Bryan Dai, Joey Zhou, Kai Qiu, Zhirong Wu, and Chong Luo. Logic-rl: Unleashing LLM reasoning with rule-based reinforcement learning. *CoRR*, abs/2502.14768, 2025. doi: 10.48550/ARXIV.2502.14768. URL https://doi.org/10.48550/arXiv.2502.14768.

Dayu Yang, Tianyang Liu, Daoan Zhang, Antoine Simoulin, Xiaoyi Liu, Yuwei Cao, Zhaopu Teng, Xin Qian, Grey Yang, Jiebo Luo, and Julian J. McAuley. Code to think, think to code: A survey on

code-enhanced reasoning and reasoning-driven code intelligence in llms. *CoRR*, abs/2502.19411, 2025. doi: 10.48550/ARXIV.2502.19411. URL https://doi.org/10.48550/arXiv.2502.19411.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pp. 2369–2380, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. Scaling relationship on learning mathematical reasoning with large language models. *CoRR*, abs/2308.01825, 2023. doi: 10.48550/ARXIV.2308.01825. URL https://doi.org/10.48550/arXiv.2308.01825.

Mert Yüksekgönül, Federico Bianchi, Joseph Boen, Sheng Liu, Zhi Huang, Carlos Guestrin, and James Zou. Textgrad: Automatic "differentiation" via text. *CoRR*, abs/2406.07496, 2024. doi: 10.48550/ARXIV.2406.07496. URL https://doi.org/10.48550/arXiv.2406.07496.

Yongcheng Zeng, Xinyu Cui, Xuanfa Jin, Qirui Mi, Guoqing Liu, Zexu Sun, Mengyue Yang, Dong Li, Weiyu Ma, Ning Yang, Jian Zhao, Jianye Hao, Haifeng Zhang, and Jun Wang. Evolving llms' self-refinement capability via synergistic training-inference optimization, 2025. URL https://arxiv.org/abs/2502.05605.

Yuxiang Zheng, Dayuan Fu, Xiangkun Hu, Xiaojie Cai, Lyumanshan Ye, Pengrui Lu, and Pengfei Liu. Deepresearcher: Scaling deep research via reinforcement learning in real-world environments. *CoRR*, abs/2504.03160, 2025. doi: 10.48550/ARXIV.2504.03160. URL https://doi.org/10.48550/arXiv.2504.03160.

Li Zhong, Zilong Wang, and Jingbo Shang. Debug like a human: A large language model debugger via verifying runtime execution step by step. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pp. 851–870. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-ACL.49. URL https://doi.org/10.18653/v1/2024.findings-acl.49.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. *CoRR*, abs/2506.01716, 2025. doi: 10.48550/ARXIV.2506.01716. URL https://doi.org/10.48550/arXiv.2506.01716.

Method	Mathematical-Reasoning Tasks						Knowledge-Intensive Tasks				Avg.
	AIME24	AIME25	AMC23	MATH	MATH500	GSM8K	HQA	2Wiki.	MSQ	Bamb	
Prompting-Based	6.7	13.3	47.5	73.8	62.2	69.4	21.1	23.8	9.9	25.5	35.3
Tool-Light (w. SFT)	30.0	**26.7**	65.0	85.6	77.2	89.4	54.7	55.7	22.8	58.8	56.6
Tool-Light (w. PA.)	30.0	**26.7**	**67.5**	85.8	**80.4**	90.8	55.5	**56.5**	24.3	56.0	57.4
Tool-Light (w. SE.)											
w. SE. 1 Loop	30.0	**26.7**	**67.5**	86.4	78.2	**92.2**	56.7	55.6	**26.4**	**59.5**	57.9
w. SE. 2 Loop	**33.3**	23.3	**67.5**	**87.4**	79.0	92.0	**57.7**	56.1	25.0	58.7	**58.0**

Table 3: Full results of Tool-Light on 10 reasoning tasks, with the best two results highlighted in **bold** and underline. We use Qwen2.5-7B-Instruct as the backbone model. Abbreviation: 2Wiki. (2WikiMultiHopQA). HQA. (HotpotQA). MSQ. (MuSiQue). Bamb. (Bamboogle). SE. (only conducting pre-alignment DPO training after SFT). PA. (conducting pre-alignment training and several self-evolved DPO alignment loops after SFT).

A The Use of Large Language Models (LLMs)

In this study, we use LLM to assist in literature reading, and we do not use any large language models during other research ideation and/or writing phase.

B More Details of Main Results

Implementation Details. We use LoRA training method (Hu et al., 2022). During training, the LoRA rank is set to 8, the epoch is set to 3, the learning rate is 7e-6, and the batch size is 8. For the self-evolved DPO alignment, we train for two rounds. During testing, we set the temperature to 1 and limit the maximum call count for each tool to 4. For knowledge-intensive tasks, we use E5-base-v2 (Wang et al., 2022) with Wikipedia retrieval based on FlashRAG's settings (Jin et al., 2025b). For mathematical-reasoning tasks, we use Bing Web Search for retrieval. To avoid overly lengthy during web search, we introduce Web Browser mechanism following Tool-Star.

Full Results at Tool-Light's Different Stages We present the full results of Qwen2.5-7B-Instruct model at Tool-Light framework's different stages. The results can be found in Table 3. From these results, we can draw the following insights: (1) As each training phase of Tool-Light progresses, the performance of the model gradually improves. For example, the performance of the model has significantly improved after SFT. After multiple rounds of DPO training, its performance further increases. (2) As the self-evolved DPO alignment loops proceed, the model's performance gradually converges. The performance of the model with the self-evolved DPO alignment is better than with only pre-aligned DPO training. However, with the increase of self-evolution rounds, the model's performance improves slowly. This may be because it is difficult to further generate samples beneficial for model training, resulting in only a small improvement in the model's performance after training.

C Impact of Different Data Ratios

In this section, we explore different data ratios' impact of two sampling methods on training performance. We sample data using the vanilla method and the entropy-guided method separately. Then, we mix the data in ratios ranging from 1:7 to 7:1. Meanwhile, we ensure that other settings remain unchanged. The model is then retrained in the first round of DPO, and the results are shown in Figure 6.

Results show that the model's performance in the TIR task is strongly related to the mixing ratio of the two data types. As the proportion of data with the vanilla method increases, both the model's performance and the number of tool calls improve. This is likely because training on this type of data makes the model more inclined to use tools to solve problems. However, the Efficiency metric does not continuously improve with the increasing data ratio, likely due to excessive tool call reducing

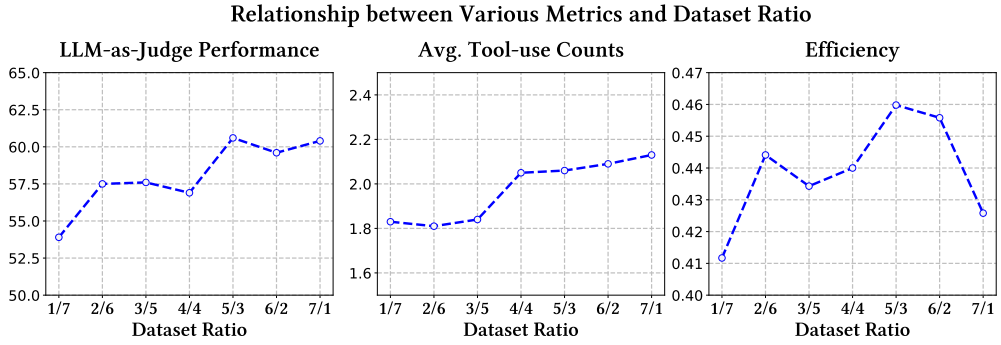

Figure 6: The variation of metrics with changes in the data ratio.

the efficiency of the model's reasoning. Based on the above analysis, we set the data ratio used in the experiment to 13:7.

D Case Study

To better demonstrate the superiority of Tool-Light over other methods in efficiency and necessity metrics, we conduct a case study. We compare the reasoning trajectories of Tool-Light with those of Tool-Star. As shown in Table 4, when addressing the same problem, even if the final answer is correct, Tool-Light's result achieves fewer tool calls compared to that of Tool-Star. And as shown in Table 5, when faced with a more difficult problem, the Tool-Light method will proactively call external tools to assist in answering questions when necessary.

Table 4: A comparison example between Tool-Light and Tool-Star on MATH500 dataset.

Example #1 from MATH500 Dataset

Question:
The square root of t is greater than 2 and less than 3.5. How many integer values of t satisfy this condition?
Golden Answer:
8

Tool-Light's TIR Stragety

<think> To determine how many integer values of t satisfy the condition $2 < \sqrt{t} < 3.5$, we can follow these steps:
1. Square all parts of the inequality to eliminate the square root:

$$2^2 < t < (3.5)^2$$

This simplifies to:

$$4 < t < 12.25$$

2. Identify the integer values of t that fall within this range. The integers between 4 and 12.25 are 5, 6, 7, 8, 9, 10, 11, and 12.
3. Count the number of these integers.
Let's implement this in Python to ensure accuracy. </think><python>

```python
# Define the range
lower_bound = 4
upper_bound = 12.25

# Find the integer values in the range
integer_values = [t for t in range(int(lower_bound) + 1, int(
    upper_bound) + 1)]

# Count the number of integer values
num_integer_values = len(integer_values)
print(num_integer_values)
```

</python><result> 8 </result> <answer>The number of integer values of t that satisfy the condition $2 < \sqrt{t} < 3.5$ is $\boxed{8}$.</answer>

Tool-Star's TIR Stragety

<think>Okay, so I need to figure out how many integer values of t satisfy the condition that the square root of t is between 2 and 3.5. Hmm, let me start by understanding what exactly the question is asking. First, the square root of t has to be greater than 2 and less than 3.5. That means $\sqrt(t) > 2$ and $\sqrt(t) < 3.5$. But since we're dealing with real numbers, maybe even irrational ones, but the question specifies that t must be an integer. So t has to be a perfect square between 4 (since 2 squared is 4) and... well, 3.5 squared. Let me compute that.
</think><python>

```python
lower = 2 ** 2
upper = 3.5 ** 2
print(f'Lower bound: {lower}, Upper bound: {upper}')
```

</python><result> Lower bound: 4, Upper bound: 12.25 </result>So $\sqrt(t)$ needs to be strictly greater than 2, which gives $t > 4$, and strictly less than 3.5, which gives $t < (3.5)^2 \approx 12.25$. Therefore, t must be integers from 5 up to 12 inclusive. To confirm:
</think><python>

```python
start = int(2**2) + 1
end = int(3.5**2)
count = end - start + 1
print(f'Start: {start}, End: {end}, Count: {count}')
```

</python><result> Start: 5, End: 12, Count: 8 </result><think>Therefore, the number of integers satisfying the condition is indeed 8. </think><answer> $\boxed{8}$ </answer>

Table 5: A comparison example between Tool-Light and Tool-Star on AIME24 dataset.

Example #2 from AIME24 Dataset

Question:
Find the largest possible real part of

$$(75+117i)z+\frac{96+144i}{z}$$

where z is a complex number with $|z|=4$.
Golden Answer:
540

Tool-Light's TIR Stragety

<think> To solve the problem, we need to find the maximum value of the expression $(75+117i)z+\frac{96+144i}{z}$ where z is a complex number with $|z|=4$.
Let's represent z as $4e^{i\theta}$, where θ is a real number. Then we can write:

$$(75+117i)z+\frac{96+144i}{z}=(75+117i)4e^{i\theta}+\frac{96+144i}{4e^{i\theta}}.$$

This can be simplified to:

$$4(75+117i)e^{i\theta}+\frac{24+36i}{e^{i\theta}}.$$

(*More reasoning contentare omitted here***)**

```python
import numpy as np

# Define the complex numbers
a = 75 + 117j
b = 96 + 144j

# Define the magnitude of z
r = 4

# Initialize the maximum real part
max_real_part = -np.inf
max_theta = 0

# Evaluate the expression for a range of theta values
for theta in np.linspace(0, 2 * np.pi, 1000):
    z = r * np.exp(1j * theta)
    expr = a * z + b / z
    real_part = np.real(expr)
    if real_part > max_real_part:
        max_real_part = real_part
        max_theta = theta

print(max_real_part)
```

</python><result> 539.9979693595761 </result><answer>The largest possible real part of the expression $(75+117i)z+\frac{96+144i}{z}$ where z is a complex number with $|z|=4$ is $\boxed{540}$.</answer>

Tool-Star's TIR Stragety

<think> To solve the problem, we need to find the maximum value of the expression $(75+117i)z+\frac{96+144i}{z}$ where z is a complex number with $|z|=4$. Let's represent z as $4e^{i\theta}$, where θ is a real number. This way, we can express z in terms of its polar form.
(*More reasoning contentare omitted here***)**
However, we should confirm this by calculating the critical points. The derivative of $324\cos\theta-504\sin\theta$ is $-324\sin\theta-504\cos\theta$. Setting this to zero gives us the critical points. But since we know the maximum value of the expression $a\cos\theta+b\sin\theta$ is $\sqrt{a^2+b^2}$, we can directly use this formula.
Thus, the largest possible real part is:
</think><answer>$\boxed{594}$ </answer>