Adapting Dynamic Sampling to Fine-Grained Rewards for Tool Learning with Curriculum Learning

Anonymous ACL submission

Abstract

Dynamic sampling is a technique employed in reinforcement Learning for Large Language Models to mitigate training instability, often caused by the gradient-decreasing problem. However, existing dynamic sampling approaches are predominantly designed for tasks with binary rewards (success/failure) and struggle to adapt to the complex reward structures of tool learning. Such task involves interdependent sub-tasks of varying difficulty and yields fine-grained, multi-faceted rewards. To bridge this gap, we introduce Dynamic Sampling with Curriculum Learning (DSCL), an algorithm tailored for the intricate dynamics of tool learning. DSCL integrates two core components: a Reward-Based Dynamic Sampler, which leverages multi-dimensional reward statistics to prioritize high-value training data; and a Task-Based Dynamic Curriculum Learning method to overcome the credit assignment problem caused by the fine-grained rewards. Extensive experiments on widely used tool learning benchmarks demonstrate the efficacy of our approach. DSCL significantly improves model performance, outperforming strong baselines by 4.75% on the BFCL V3 dataset and 4.02% on the API-Bank dataset. Our method will be publicly available at http://anonymous.com/.

1 Introduction

While Large Language Models (LLMs) have demonstrated considerable proficiency in complex reasoning tasks such as tool learning (Qin et al., 2024; Kumar et al., 2025; Qu et al., 2025; Guo et al., 2025), optimizing their performance via Reinforcement Learning (RL) presents significant challenges. Recent research primarily focuses on mitigating the training instability and improving the model's exploratory capabilities (Shao et al., 2024). So, the instance-level dynamic sampling method (Yu et al., 2025; Xi et al., 2025) and curriculum learning in the training procedure (Feng et al., 2025a) are wildly used for these problems.

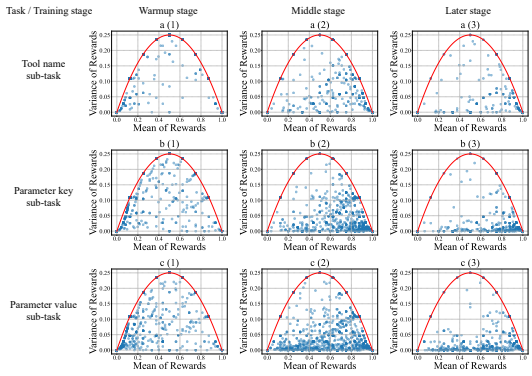

Figure 1: Reward Mean-Variance changes for each sub-task across three training stages. The red curve represents the theoretical mean-variance relationship of binary rewards, where the variance is strictly determined by the mean. The blue dots represent the distribution of fine-grained rewards for individual rollouts, providing richer supervision signals. This diversity is evident across two dimensions: (1) distinct distributions across sub-tasks (rows), and (2) evolution across training stages (columns)—from basic learning (Warmup) to steady convergence (Middle) and fully converged (Later).

Most dynamic sampling approaches in RL for LLMs are designed for tasks (e.g., mathematical reasoning tasks) relying on simple binary rewards (i.e., success or failure) and fail to address the sequential complexities of tool learning. The tool learning task can be decomposed into quantifiable, interdependent sub-tasks: format prediction, tool selection, key-value prediction, and parameter completion, yielding a multi-dimensional reward signal. This creates a critical mismatch, leading to two primary failures. First, aggregating these rich rewards into a single score induces a credit assignment problem (Harutyunyan et al., 2019; Seo et al., 2019; Zhang et al., 2025a), preventing the model from learning which specific actions are effective and failing to learn the hierarchical strategy effectively. Second, because multi-dimensional rewards have more complex statistical properties than bi-

nary ones (Figure 1), a single evaluation metric is insufficient to identify the most informative training samples, leading to stagnation in learning effectiveness and a lack of data diversity during training.

To overcome these limitations, we propose Dynamic Sampling with Curriculum Learning (DSCL), a novel methodology designed to fully exploit the potential of fine-grained process rewards in tool learning. DSCL integrates two synergistic strategies: (1) Task-Based Dynamic Curriculum Learning (TDCL): TDCL introduces a phased training strategy that leverages the logical dependencies and varying difficulties of sub-tasks in the procedural reward. By delivering stage-specific, immediate feedback, it provides the model with fine-grained guidance, thereby improving training effectiveness. (2) Reward-Based Dynamic Sampling (RDS): At each stage of the TDCL, RDS assesses data from three dimensions: the reward mean, the reward variance of a group of rollouts, and the evolution of these statistics over the training history. This provides a holistic assessment of sample difficulty, stability, and long-term learning trajectory. Extensive experiments demonstrate the effectiveness and generalizability of our proposed method. Overall, this paper offers the following contributions:

- We conduct a detailed analysis of the differences in RL training between tool learning with fine-grained rewards and math tasks with binary rewards. Furthermore, we reveal the shortcomings of existing optimization strategies on tool learning.
- We propose a two-component methodology within DSCL. RDS method dynamically samples data using multiple dimensions of the mean and variance of rewards. The TDCL adaptively manages the asynchronous convergence of interdependent sub-tasks.
- We conduct comprehensive experiments to evaluate our DSCL method against several strong baselines. The results on the BFCL V3 and API-Bank confirm its significant effectiveness and superiority.

2 Related Work

2.1 Tool Learning

The ability of LLMs to interact with external tools has evolved significantly. Foundational work like ToolLLM (Qin et al., 2023) pioneered the field by creating large-scale benchmarks for supervised fine-tuning. To overcome the limitations of static imitation, subsequent research has employed Reinforcement Learning to optimize sequential tool use, as seen in StepTool (Yu et al., 2024), Search-R1 (Jin et al., 2025), and Tool-N1 (Zhang et al., 2025b), etc. The success of RL, however, is highly dependent on effective reward engineering; previous works such as ToolRL (Qian et al., 2025) and ARTIST (Singh et al., 2025) focus on systematically addressing this challenge.

2.2 Optimization Strategies on RL

Curriculum learning and dynamic sampling are widely used to improve the effectiveness of RL-based training. A core idea of curriculum learning is to structure training from simple to complex tasks, as demonstrated by Confucius (Gao et al., 2024). A prominent strategy within this is difficulty-aware sampling, which dynamically focuses on problems of appropriate difficulty. Its effectiveness was shown at scale by Kimi K1.5 (Team et al., 2025), with focused studies like RCS (Feng et al., 2025b) and online difficulty filtering (Bae et al., 2025) confirming that an intermediate difficulty level maximizes learning effectiveness. Dynamic sampling organizes samples from an alternative perspective. Approaches like POLARIS (An et al., 2025) employ adaptive filtering to enhance exploration, while frameworks such as DAPO (Yu et al., 2025) and VAPO (Yue et al., 2025) integrate dynamic sampling directly into the policy optimization process to address scaling challenges. Razin et al. (2025) have also theoretically explored the detrimental effects of low-variance rewards on training effectiveness.

Unlike existing RL optimization methods, we produce a novel method tailored for tool learning task with fine-grained process rewards. By synergistically integrating a multi-stage training strategy based on sub-tasks decomposition with a multi-dimensional dynamic sampling policy, we achieve a significant performance boost for RL approaches on tool learning task.

3 Reward Design for Tool Learning

Following the design proposed by Qian et al. (2025), we present a formal definition of the fine-grained reward to enhance the clarity of our approach. For the query X, the reward functions compare the predicted tool calls $\widehat{Y} = \{\hat{y}_1, ..., \hat{y}_m\}$

with the ground-truth tools $Y = \{y_1, ..., y_m\}$. It includes the following four components:

Format Reward The format reward $R_{\text{format}} \in \{0, 1\}$ checks whether the model's response meets the requirement for both content and order:

$$R_{\text{format}} = \begin{cases} 1, & \text{if all required fields appear} \\ & \text{and are in the correct order} \\ 0, & \text{otherwise} \end{cases}. \quad (1)$$

Tool Name Reward The tool name reward R_{name} evaluates the correctness of each tool's name. The $N_{\widehat{Y}}$ and N_Y represent the predicted tool names and the ground truth, respectively.

$$R_{\text{name}} = \frac{|N_{\widehat{Y}} \cap N_Y|}{|N_{\widehat{Y}} \cup N_Y|} \in [0, 1]. \quad (2)$$

Parameter Key Reward The parameter key reward $R_{\text{key}} \in [0, |Y|]$ checks whether the key of the parameter of each tool is correct:

$$R_{\text{key}} = \sum_{y_i \in Y} \frac{|K(f(y_i)) \cap K(y_i)|}{|K(f(y_i)) \cup K(y_i)|}, \quad (3)$$

where function $f(y_i)$ selects the best matches tool $\hat{y}_i$ with y_i in the predicted tools. If there is no match, it returns empty. The K represents the set of keys of a specific tool.

Parameter Value Reward The parameter value reward $R_{\text{value}} \in [0, \sum_{y_i \in Y} |K(y_i)|]$ evaluates the parameter values for all the matching keys.

$$R_{\text{value}} = \sum_{y_i \in Y} \sum_{k \in K(y_i)} \frac{\mathbb{1}[V(f(y_i)^k) = V(y_i^k)]}{|V(f(y_i)) \cup V(y_i)|}, \quad (4)$$

where $V(y_j^k)$ represents the value that is corresponding to the tool y_j's key k.

Total Reward As indicated above, the correctness-related reward initially has a range of $[0, 1 + |Y| + \sum_{y_i \in Y} |K(y_i)|]$, with an upper bound that varies depending on the specific tool utilized. To ensure a consistent value range across all data instances and to prevent a scale imbalance with the format reward, we employ a mapping function $\mathcal{M}$ to transform the correctness-related reward into a new range of $[-3, 3]$. This transformed value is then summed with the format reward to compute the final reward $R \in [-3, 4]$.

$$R = R_{\text{format}} + \mathcal{M}_{-3}^{3}(R_{\text{name}} + R_{\text{key}} + R_{\text{value}}), \quad (5)$$

$$\mathcal{M}_{min}^{max}(R) = \left\{ \frac{(max - min) * (R - Min(R))}{Max(R) - Min(R)} + min \right\}, \quad (6)$$

where $\mathcal{M}$ represents the mapping function, Max and Min denote the maximum and minimum functions. All these functions follow the definition of Qian et al. (2025).

4 Methodology

In this section, we will introduce our proposed Dynamic Sampling with Curriculum Learning for RL-based tool learning. As shown in Figure 2, the training framework DSCL contains two cascaded components: Task-Based Dynamic Curriculum Learning (TDCL) and Reward-Based Dynamic Sampling (RDS). The TDCL is first to partition the whole training procedure into three distinct sub-stages based on reward information derived from the current training batch and trains the model stage by stage. Subsequently, within each training stage, the RDS method is utilized. This method performs a multi-dimensional difficulty assessment for each data using its complete set of rollouts, after which a selective filtering is conducted. The combination of these two methods effectively enhances training stability and further improves optimization performance. A detailed description of these components is provided in the following subsections.

We first describe the basic formulation of the tool learning task. Let $D = \{d_1, ..., d_{\text{batchsize}}\}$ represents the dataset at the i-th batch of training. For each data $d_j \in D$, we generate G rollouts and note their corresponding rewards as $\mathcal{R}^{i,j} = \{R^{i,j,1}, ..., R^{i,j,G}\}$.

4.1 Task-Based Dynamic Curriculum Learning (TDCL)

In this section, we provide a detailed description of the TDCL method, which is specifically designed for tool learning task. To address the credit assignment problem that arises in existing training methods utilizing fine-grained rewards, our approach partitions the sub-tasks based on their logical relationships and difficulty levels. This partitioning, guided by the reward design presented in Section 3, enables the model to focus on different sets of sub-tasks during distinct training stages.

4.1.1 Building Training Curriculum

We first describe the design of our curriculum learning strategy and the mechanism for directing the

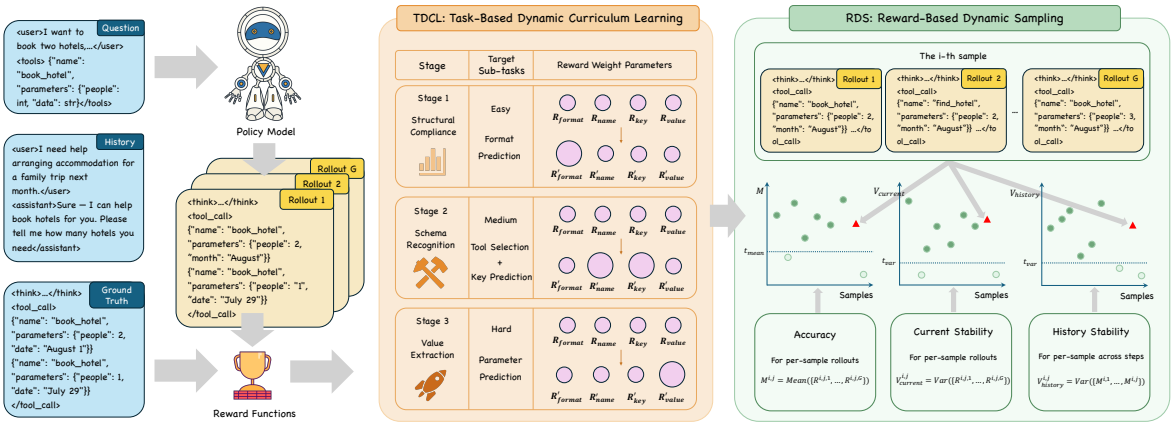

Figure 2: The training framework of our DSCL method. For each training batch, the TDCL method first determines the training stage based on the rewards of each sub-task. Then the RDS method dynamically samples the data depending on the mean and variance dimensions. Following the application of these strategies, the model is subsequently updated.

model's focus to different sub-tasks at various training stages. We divide the original training process into three stages. To better guide the model to focus on different sub-tasks at various stages of training, we modify Equation 5 into the following form:

$$R = \lambda_1 R_{\text{format}} + \mathcal{M}_{-3}^{4-\lambda_1}(\lambda_2(R_{\text{name}} + R_{\text{key}}) + \lambda_3 R_{\text{value}}). \tag{7}$$

We maintain the value range of the final reward and introduce three parameters λ_1, λ_2 and λ_3 to control the contribution of each sub-task's reward at different stages of training. Consequently, the three-stage training procedure designed for this task is detailed as follows:

Stage 1 The format of the model's output is fundamental to the downstream task of structured information extraction. Therefore, we assign a higher weight to the format reward during the initial stages of training. The corresponding parameters are set as follows: $\lambda_1 = 2.5, \lambda_2 = \lambda_3 = 0.5$.

Stage 2 The second training stage aims to target the extraction of tool name and parameter key. These sub-tasks present a simpler learning problem compared to parameter value extraction, owing to their finite (or highly restricted) solution spaces and less ambiguous evaluation criteria. Accordingly, the reward parameters are set as: $\lambda_2 = 1.5, \lambda_1 = \lambda_3 = 0.5$.

Stage 3 The final training stage is dedicated to parameter value extraction. This task presents a higher level of difficulty due to its unconstrained prediction space and its inherent dependency on the parameter keys correctly identified in the previous stage. Finally, the reward parameters are set as: $\lambda_3 = 2.5, \lambda_1 = \lambda_2 = 0.5$.

4.1.2 Learning Stage Transition

For the three training stages we designed, we dynamically assess the model's training status by monitoring the rewards from sub-tasks. This process automatically determines the model's current training stage. Specifically, we compute the mean reward for each of the four sub-tasks within every training batch. This serves as a metric to gauge the model's performance at the current batch. To ensure robustness, we then utilize the results from the most recent k batches to determine whether the model has converged on a particular sub-task. As an illustrative example, the formula for the format reward is presented below:

$$\overline{R}_{\text{format}} = \frac{\sum_{i=1}^{k} \sum_{j=1}^{batchsize} \sum_{g=1}^{G} R_{\text{format}}^{i,j,g}}{k * batchsize * G * Max(R_{\text{format}})}. \tag{8}$$

We establish the convergence threshold for each sub-task (e.g., $\hat{R}_{\text{format}}$). When the mean value surpasses the threshold (e.g., $\overline{R}_{\text{format}} > \hat{R}_{\text{format}}$), we consider the sub-task to achieve convergence and trigger the transition to the subsequent training stage. The determination of the threshold $\hat{R}_{\text{format}}$ for the transition between Stage 1 and Stage 2, and the $\hat{R}_{\text{name}}$, $\hat{R}_{\text{key}}$ between Stage 2 and Stage 3, is detailed in the Appendix B.

4.2 Reward-Based Dynamic Sampling (RDS)

To further enhance training stability, we design a multi-dimensional data evaluation methodology tailored for tool learning task with fine-grained rewards. Depending on these indicators, we divide the data into different categories and apply a unique instance-level sampling policy to each category.

4.2.1 Multi-Dimensional Indicators

In order to provide insights into the intrinsic difficulty of each data d_i in the j-th batch, we select the following three indicators to measure the difficulty of each data instance for the current model:

Accuracy For each data at the current training batch, we calculate the mean reward for all their rollouts to evaluate the average accuracy:

$$M^{i,j} = Mean(\{R^{i,j,1}, ..., R^{i,j,G}\}). \quad (9)$$

Current Stability We calculate the variance of all the rollouts for each data to evaluate the stability of the current model on this data:

$$V^{i,j}_{\text{current}} = Var(\{R^{i,j,1}, ..., R^{i,j,G}\}). \quad (10)$$

History Stability To evaluate the stability of the model's performance on each data during training, and to prevent filtering out a sample as soon as the model acquires the capability to handle it, we compute the variance of its reward across batches:

$$V^{i,j}_{\text{history}} = Var(\{M^{i,1}, ..., M^{i,j}\}). \quad (11)$$

4.2.2 Instance-Level Sampling Policy

Data Division Depending on these three indicators, we dynamically divide the data into three categories:

- a. Easy data: $M^{i,j} = 4$.
- b. Hard data: $M^{i,j} < t_{\text{mean}}$.
  - b.1 $V^{i,j}_{\text{current}} > t_{\text{var}}$ or $V^{i,j}_{\text{history}} > t_{\text{var}}$.
  - b.2 $V^{i,j}_{\text{current}} < t_{\text{var}}$ and $V^{i,j}_{\text{history}} < t_{\text{var}}$.
- c. Intermediate data: $M^{i,j} \in [t_{\text{mean}}, 4)$.
  - c.1 $V^{i,j}_{\text{current}} > t_{\text{var}}$ and $V^{i,j}_{\text{history}} > t_{\text{var}}$.
  - c.2 $V^{i,j}_{\text{current}} < t_{\text{var}}$ or $V^{i,j}_{\text{history}} < t_{\text{var}}$.

Where the t_{mean} and t_{var} represent the thresholds for the mean and variance, respectively. The setting of these parameters is detailed in Appendix B.

Sampling Policy We adopt different sampling strategies for data with varying levels of difficulty. Specifically, we completely filter both easy and hard data for which the current model exhibits highly stable and confident predictions. Conversely, we fully retain two categories of instances: (1) hard data where the model demonstrates significant uncertainty, indicating effective exploration; (2) intermediate data that elicit high prediction diversity. In order to maintain a stable training process (Dang and Ngo, 2025; Wang et al., 2025), the remaining data are partially retained. The corresponding sampling proportions are presented below:

$$Ratio_i = \begin{cases} 0.0, & d_i \in \{a, b.2\} \\ 0.5, & d_i \in \{c.2\} \\ 1.0, & d_i \in \{b.1, c.1\} \end{cases} . \quad (12)$$

Since the answers for this task require formatted extraction, the accuracy of the output format directly impacts the performance of subsequent subtasks. To prevent formatting errors from confounding the assessment of sample difficulty, we implement our sampling strategy only after the model has converged on the formatting task, which corresponds to the completion of stage 1 of the TDCL method. The overall framework of our method is illustrated in Figure 2.

5 Experiments Settings

5.1 Datasets

To foster a robust and generalization tool-use capability of LLMs, we follow the data composition and processing methodology of ToolRL (Qian et al., 2025) to build the training dataset. Finally, the training dataset contains 2K examples from ToolACE (Liu et al., 2024) and 1K each from Hammer (Lin et al., 2024) and xLAM (Zhang et al., 2024). The detailed information are appended in Appendix A.

As for the evaluation dataset, to ensure a rigorous and directly comparable evaluation, we assess our method on the Berkeley Function Calling Leaderboard (BFCL) (Patil et al., 2025) and API-Bank (Li et al., 2023), the same benchmarks used in ToolRL (Qian et al., 2025). We do 5 runs for each method and report the average score of the accuracies (%) with the confidence interval.

5.2 Baselines

To demonstrate the effectiveness and generalizability of our method, we conduct experiments on three widely-used foundational models: Qwen2.5-7B-

Model	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
Qwen2.5-7B-Instruct (Yang et al., 2024)	47.68±0.37	68.98±0.19	63.31±0.13	8.88±0.27	72.22±0.00	71.93±0.12
Tool-N1 (Zhang et al., 2025b)	53.91±0.13	77.50±0.07	73.39±0.09	10.00±0.08	77.78±0.00	77.85±0.07
ToolRL (Qian et al., 2025)	56.96±0.10	85.21±0.06	73.39±0.09	13.25±0.06	83.33±0.00	75.87±0.03
w/ DAPO (Yu et al., 2025)	47.35±0.23	68.27±0.10	64.55±0.06	7.38±0.29	77.78±0.00	71.80±0.08
w/ SMV (Xu et al., 2025)	57.03±0.25	85.76±0.15	73.79±0.17	12.62±0.09	83.33±0.00	77.59±0.13
w/ SMR (Bae et al., 2025)	58.18±0.21	86.08±0.17	74.87±0.10	13.74±0.11	83.33±0.00	75.43±0.13
w/ RDS (Ours)	59.55±0.10	85.54±0.05	76.10±0.06	17.38±0.06	83.33±0.00	79.86±0.05
w/ TDCL (Ours)	59.95±0.15	86.25±0.03	76.06±0.07	18.13±0.12	83.33±0.00	79.54±0.02
w/ DSCL (Ours, RDS+TDCL)	60.25±0.19	86.42±0.07	76.85±0.03	18.50±0.14	83.33±0.00	78.44±0.05
Qwen3-8B (Yang et al., 2025)	57.18±0.14	86.08±0.07	73.79±0.09	15.86±0.15	83.33±0.00	75.43±0.00
Tool-N1 (Zhang et al., 2025b)	60.52±0.04	88.06±0.00	73.43±0.08	22.00±0.00	**88.89±0.00**	71.71±0.04
ToolRL (Qian et al., 2025)	62.58±0.05	89.54±0.04	70.86±0.00	30.25±0.06	83.33±0.00	67.43±0.03
w/ DAPO (Yu et al., 2025)	64.07±0.11	89.93±0.03	75.54±0.06	31.13±0.12	77.78±0.00	82.62±0.04
w/ SMV (Xu et al., 2025)	62.96±0.07	88.34±0.05	72.88±0.06	27.13±0.08	83.33±0.00	75.94±0.03
w/ SMR (Bae et al., 2025)	63.47±0.12	89.28±0.07	72.23±0.06	30.00±0.17	83.33±0.00	76.92±0.04
w/ RDS (Ours)	66.21±0.05	89.93±0.00	77.70±0.03	**32.88±0.06**	**88.89±0.00**	83.07±0.00
w/ TDCL (Ours)	65.83±0.07	87.15±0.03	78.72±0.04	31.88±0.06	83.33±0.00	83.79±0.03
w/ DSCL (Ours, RDS+TDCL)	**66.74±0.11**	**90.50±0.04**	**79.57±0.03**	**32.88±0.07**	**88.89±0.00**	**84.58±0.04**
Llama3.1-8B-Instruct (Dubey et al., 2024)	49.63±0.23	71.23±0.18	63.27±0.21	10.60±0.16	72.22±0.00	72.91±0.19
Tool-N1 (Zhang et al., 2025b)	56.73±0.10	84.06±0.07	73.58±0.03	12.98±0.09	83.33±0.00	77.27±0.04
ToolRL (Qian et al., 2025)	58.42±0.06	86.69±0.04	74.48±0.03	14.66±0.06	83.33±0.00	77.49±0.04
w/ DAPO (Yu et al., 2025)	61.02±0.12	88.16±0.06	77.03±0.06	16.51±0.08	83.33±0.00	78.93±0.04
w/ SMV (Xu et al., 2025)	60.73±0.20	86.97±0.12	76.35±0.09	15.34±0.17	83.33±0.00	78.25±0.13
w/ SMR (Bae et al., 2025)	60.29±0.10	87.03±0.07	76.14±0.04	15.03±0.08	83.33±0.00	77.68±0.06
w/ RDS (Ours)	62.33±0.06	87.28±0.02	76.85±0.05	18.13±0.08	83.33±0.00	79.52±0.02
w/ TDCL (Ours)	61.98±0.06	87.20±0.03	77.41±0.05	17.79±0.05	83.33±0.00	79.30±0.02
w/ DSCL (Ours, RDS+TDCL)	63.17±0.13	87.26±0.06	79.33±0.03	19.85±0.06	83.33±0.00	80.10±0.05

Table 1: Results on the BFCL V3 dataset. The "underline" signifies the better score between the models with the same foundation model. The "**bold**" indicates the best score among all the systems of each language pair.

Instruct (Yang et al., 2024), Qwen3-8B (Yang et al., 2025), and Llama3.1-8B-Instruct (Dubey et al., 2024). On these models, we compare our approach against a set of carefully selected baselines.

We adopt the ToolRL (Qian et al., 2025) as our baseline and incorporate our proposed optimizations. ToolRL is characterized by its fine-grained reward design and the GRPO training algorithm. For the designation of reward, we also compare our approach with the Tool-N1 (Zhang et al., 2025b) method, which utilizes a binary reward signal for tool learning task. For the optimization strategy, we select the widely-adopted DAPO (Yu et al., 2025) method as a key baseline. Additionally, we conduct two specialized strategies that sample only by max variance (SMV) (Xu et al., 2025) or sampling only by mean reward (SMR) (Bae et al., 2025).

Since most of the sampling methods in the baselines have not been used on tool learning task, we implement the results of these methods on tool learning task based on their open-source codes. For SMV and SMR, we perform the method with curriculum learning based on our methodology. The details of the prompt are in Appendix G.

6 Experiments

6.1 Main Results

We present the results of our proposed method and other baselines of BFCL and API-Bank in Table 1 and 2. The experimental results demonstrate that the DSCL method achieves the best performance with a 4.75% gain on the BFCL V3 dataset and 4.02% gain on the API-Bank dataset. Besides, both our RDS and TDCL bring a significant improvement over the original ToolRL baseline. Notably, our methods also outperform other dynamic sampling approaches such as DAPO, SMV, and SMR. This demonstrates that our approach enhances model performance by progressively focusing the training process on more valuable data.

Additionally, we find that directly using the existing dynamic sampling methods demonstrates limited improvement on tool learning task. These methods are mainly designed for binary reward tasks. Due to the gap in tasks, our method, designed for fine-grained, multi-faceted rewards, effectively addresses this limitation. Besides, another reason is that dynamic sampling in tool learning

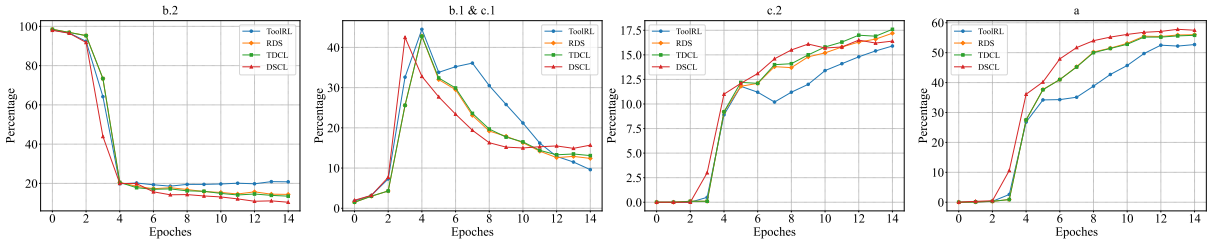

Figure 3: Data distribution across difficulty levels during training epochs on Qwen2.5-7B-Instruct. Following Section 4.2, these categories are: 1) b.2: challenging instances that the model consistently fails to answer correctly; 2) b.1 & c.1: inconsistent instances characterized by high variance in rollouts; 3) c.2: instances where the model is stable but erroneous; 4) a: instances that are stably and correctly handled.

Model	Overal Acc	Level 1	Level 2	Level 3
Qwen2.5-7B-Instruct (Yang et al., 2024)	58.29±0.18	65.41±0.14	43.28±0.00	44.27±0.47
Tool-N1 (Zhang et al., 2025b)	60.47±0.00	70.68±0.00	46.27±0.00	36.64±0.00
ToolRL (Qian et al., 2025)	61.81±0.07	72.18±0.00	56.72±0.00	32.82±0.46
w/ DAPO (Yu et al., 2025)	58.96±0.08	66.17±0.10	41.79±0.00	45.80±0.00
w/ SMV (Xu et al., 2025)	61.64±0.12	69.67±0.14	49.25±0.00	43.51±0.45
w/ SMR (Bae et al., 2025)	62.31±0.10	72.43±0.11	56.72±0.00	34.35±0.45
w/ RDS (Ours)	63.48±0.00	72.18±0.00	56.72±0.00	40.46±0.00
w/ TDCL (Ours)	63.65±0.07	74.19±0.11	61.19±0.00	32.82±0.00
w/ DSCL (Ours, RDS+TDCL)	64.99±0.00	73.18±0.00	65.67±0.00	39.69±0.00
Qwen3-8B (Yang et al., 2025)	62.48±0.16	72.18±0.19	56.72±0.00	35.88±0.00
Tool-N1 (Zhang et al., 2025b)	63.32±0.10	70.43±0.11	55.22±0.00	45.80±0.49
ToolRL (Qian et al., 2025)	64.32±0.10	72.68±0.12	62.69±0.00	39.69±0.45
w/ DAPO (Yu et al., 2025)	66.67±0.00	74.19±0.00	64.18±0.00	45.04±0.00
w/ SMV (Xu et al., 2025)	65.49±0.07	73.43±0.11	61.19±0.00	43.51±0.00
w/ SMR (Bae et al., 2025)	65.66±0.09	72.93±0.14	62.69±0.00	45.04±0.00
w/ RDS (Ours)	67.17±0.00	74.69±0.00	64.18±0.00	45.80±0.00
w/ TDCL (Ours)	67.67±0.11	**75.44±0.11**	61.19±0.00	47.33±0.45
w/ DSCL (Ours, RDS+TDCL)	**68.17±0.07**	**75.44±0.11**	64.18±0.00	**48.09±0.00**
Llama3.1-8B-Instruct (Dubey et al., 2024)	61.63±0.15	67.90±0.13	65.67±0.00	40.46±0.45
Tool-N1 (Zhang et al., 2025b)	62.90±0.10	69.60±0.15	65.67±0.00	41.06±0.00
ToolRL (Qian et al., 2025)	62.48±0.00	68.42±0.00	67.16±0.00	41.98±0.00
w/ DAPO (Yu et al., 2025)	63.83±0.07	69.76±0.11	68.66±0.00	43.28±0.00
w/ SMV (Xu et al., 2025)	63.08±0.09	68.97±0.13	67.16±0.00	43.07±0.00
w/ SMR (Bae et al., 2025)	63.16±0.10	69.18±0.11	67.46±0.67	42.76±0.00
w/ RDS (Ours)	64.48±0.00	70.16±0.00	68.66±0.00	45.04±0.00
w/ TDCL (Ours)	64.99±0.07	71.18±0.12	68.66±0.00	44.27±0.00
w/ DSCL (Ours, RDS+TDCL)	66.50±0.09	71.93±0.11	**71.64±0.00**	47.33±0.45

Table 2: Results on the API-Bank dataset.

requires warm-up first and cannot start sampling directly, which is analyzed in Section 6.4. This result highlights the need for a specially designed dynamic sampling method to address this task.

6.2 Improvement on Hard Data

To conduct a more in-depth analysis of how our method enhances the model's capabilities, we analyze the proportion of the training data at different difficulty levels for every training epoch. Figure 3 illustrates the performance of various training approaches (the other two models are appended in Appendix C), showing the evolving proportions of data across four difficulty levels. The experimental results reveal the limitations of conventional optimization strategies on hard samples, as they fail to further improve the model's capability on these data. In contrast, our proposed method, which integrates staged training with a multi-dimensional dynamic sampling strategy, enhances the model's performance on these difficult instances, thereby improving the effectiveness of the training process.

6.3 Analysis on the Sampling Data

We first analyze the model trained with all the data. Figure 4 illustrates the distribution of means and variances of training data with varying difficulty across different training stages and sub-tasks. Specifically, we classify the difficulty of the data based on the number of tools, the number of tool parameters, and the number of dialogue turns, with higher values of these metrics corresponding to increased data complexity. It can be found that, compared to easy data, the distribution of hard data across each sub-plot exhibits characteristics of lower means and higher variances. Furthermore, this phenomenon becomes more pronounced with the progression of training. Therefore, these samples should be given greater emphasis in the later stages of training; otherwise, the model will struggle to overcome its performance bottleneck.

Based on this analysis, we record samples that are selected for training versus those that are excluded by our DSCL method for comparison. As illustrated in Figure 5, the majority of the data filtered out by DSCL shows a significant overlap with the simple samples presented in Figure 4. Furthermore, excluding the initial warmup phase where no sampling occurs, we tally the number of samples used in training versus those discarded in the subsequent two stages. The counts are $(2243, 1397)$ and $(1950, 1690)$, indicating a gradual decrease in the quantity of data deemed valuable. These comparisons confirm that the DSCL method successfully and continuously focuses on valuable and challenging samples throughout training.

6.4 Training Tips on RDS

Tool learning requires strict response formatting to extract tool information in a structured manner. As shown in Table 3, directly applying RDS without curriculum learning to Qwen2.5-7B-Instruct leads

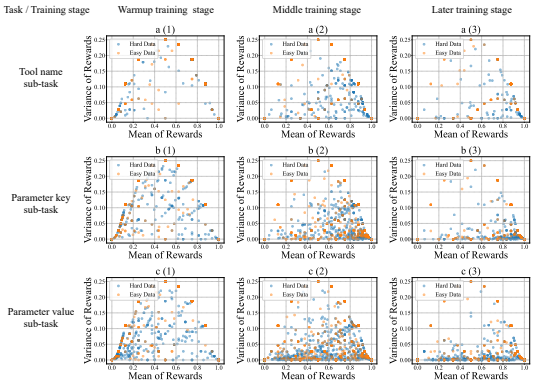

Figure 4: Mean-Variance Scatter Plots of hard data (blue point) and easy data (orange point) across different stages of ToolRL training. We map the values of each reward to the interval $[0, 1]$.

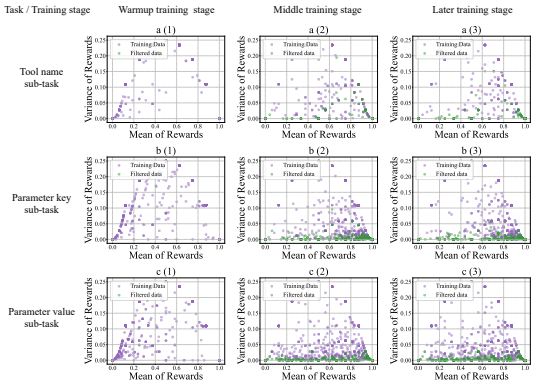

Figure 5: Mean-Variance Scatter Plots of training data (purple points) and filtered data (green points) rewards across different stages of DSCL training. We map the values of each reward to the interval $[0, 1]$.

Model	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc
Qwen2.5-7B-Instruct	47.68	68.98	63.31	8.88
ToolRL	56.96	85.21	73.39	13.25
+RDS w/o CL	47.34	68.46	64.02	7.75
+RDS	60.48	87.27	76.63	18.25
Qwen3-8B	57.18	86.08	73.79	15.86
ToolRL	62.58	89.54	70.86	30.25
+RDS w/o CL	64.72	88.65	75.34	31.45
+RDS	66.21	89.93	77.70	32.88
Llama3.1-8B-Instruct	49.63	71.23	63.27	10.60
ToolRL	58.42	86.69	74.48	14.66
+RDS w/o CL	60.94	86.82	75.46	16.75
+RDS	62.33	87.28	76.85	18.13

Table 3: Ablation study on the impact of curriculum learning (CL) in RDS. Due to space limitations, we only show the most representative metrics on the BFCL V3.

to catastrophic performance degradation. This collapse occurs because aggressive dynamic sampling filters out the majority of training instances before the model has acquired basic formatting competence. To address this, we incorporate curriculum learning into our dynamic sampling approach by disabling RDS during the initial warmup stage. RDS is activated only after the training reward stabilises, as determined through dynamic monitoring. This design ensures the model first establishes reliable output formatting before dynamic sampling takes effect, enabling it to fully leverage the benefits of difficulty-aware data selection.

However, Qwen3-8B and Llama3.1-8B-Instruct present a markedly different picture: applying RDS without curriculum learning still yields a performance *gain* over the ToolRL baseline, rather than a collapse. We attribute this difference to the stronger instruction-following ability of these two models, which allows them to maintain well-formed output formatting even when a large fraction of training data is filtered out by dynamic sampling, thereby preserving sufficient valid training signal throughout training. Nevertheless, incorporating the curriculum learning warmup consistently brings further gains for both models, confirming that the warmup stage is universally beneficial: it accelerates early reward stabilisation and provides a more favourable initialisation point from which RDS can operate, regardless of the backbone's baseline instruction-following strength.

7 Conclusion

In this work, we address the critical mismatch between conventional dynamic sampling methods designed for binary rewards and the fine-grained reward structures inherent in tool learning task. In order to overcome these gaps, we introduced Dynamic Sampling with Curriculum Learning (DSCL), a novel framework that synergistically combines the Task-Based Dynamic Curriculum Learning (TDCL) and the Reward-Based Dynamic Sampling (RDS) methods. TDCL manages the asynchronous learning of interdependent sub-tasks, while RDS leverages multi-dimensional reward statistics to perform more holistic and effective data sampling. Our extensive experiments on the BFCL V3 and API-Bank benchmarks confirm the superiority of our approach. To demonstrate the robustness and generalizability, we apply DSCL to three widely used Large Language Models. The results show the effectiveness in harnessing fine-grained process rewards to improve the stability and performance of reinforcement learning for large language models in complex tool-use scenarios across all models.

8 Limitations

In this work, based on the GRPO reinforcement learning method, we carefully designed a dynamic sampling method for the tool learning task. The experiment demonstrates that our method further improves the effect of reinforcement learning on tool learning. However, our work still has the following limitations:

In order to facilitate alignment with our subsequent ToolRL method, we adopt their dataset settings, including the number and ratio of the training data. After introducing our dynamic sampling method, we can further explore the training data settings that are suitable for our method and further improve its performance.

The current design of the reward function has been weighted according to the difficulty of different sub-tasks in tool learning. This is similar to the idea of our task-based dynamic curriculum learning method. We can further improve the reward design methods, such as combining the rewards of each sub-task in the same proportion, which can further leverage the advantages of our method and improve performance.

References

Chenxin An, Zhihui Xie, Xiaonan Li, Lei Li, Jun Zhang, Shansan Gong, Ming Zhong, Jingjing Xu, Xipeng Qiu, Mingxuan Wang, and Lingpeng Kong. 2025. Polaris: A post-training recipe for scaling reinforcement learning on advanced reasoning models.

Sanghwan Bae, Jiwoo Hong, Min Young Lee, Hanbyul Kim, JeongYeon Nam, and Donghyun Kwak. 2025. Online difficulty filtering for reasoning oriented reinforcement learning. *arXiv preprint arXiv:2504.03380*.

Quy-Anh Dang and Chris Ngo. 2025. Reinforcement learning for reasoning in small llms: What works and what doesn't. *arXiv preprint arXiv:2503.16219*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. 2025a. Retool: Reinforcement learning for strategic tool use in llms. *arXiv preprint arXiv:2504.11536*.

Zihao Feng, Xiaoxue Wang, Ziwei Bai, Donghang Su, Bowen Wu, Qun Yu, and Baoxun Wang. 2025b. Improving generalization in intent detection: Grpo with reward-based curriculum sampling. *arXiv preprint arXiv:2504.13592*.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18030–18038.

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*.

Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, and 1 others. 2019. Hindsight credit assignment. *Advances in neural information processing systems*, 32.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*.

Komal Kumar, Tajamul Ashraf, Omkar Thawakar, Rao Muhammad Anwer, Hisham Cholakkal, Mubarak Shah, Ming-Hsuan Yang, Phillip HS Torr, Salman H Khan, and Fahad Shahbaz Khan. 2025. Llm post-training: A deep dive into reasoning large language models. *CoRR*.

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

Qiqiang Lin, Muning Wen, Qiuying Peng, Guanyu Nie, Junwei Liao, Jun Wang, Xiaoyun Mo, Jiamu Zhou, Cheng Cheng, Yin Zhao, and 1 others. 2024. Hammer: Robust function-calling for on-device language models via function masking. *arXiv preprint arXiv:2410.04587*.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, and 1 others. 2024. Toolace: Winning the points of llm function calling. *arXiv preprint arXiv:2409.00920*.

Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. 2025. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and

Heng Ji. 2025. Toolrl: Reward is all tool learning needs. *arXiv preprint arXiv:2504.13958*.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Xuanhe Zhou, Yufei Huang, Chaojun Xiao, and 1 others. 2024. Tool learning with foundation models. *ACM Computing Surveys*, 57(4):1–40.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, and 1 others. 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. *arXiv preprint arXiv:2307.16789*.

Changle Qu, Sunhao Dai, Xiaochi Wei, Hengyi Cai, Shuaiqiang Wang, Dawei Yin, Jun Xu, and Ji-Rong Wen. 2025. Tool learning with large language models: A survey. *Frontiers of Computer Science*, 19(8):198343.

Noam Razin, Zixuan Wang, Hubert Strauss, Stanley Wei, Jason D Lee, and Sanjeev Arora. 2025. What makes a reward model a good teacher? an optimization perspective. *arXiv preprint arXiv:2503.15477*.

Minah Seo, Luiz Felipe Vecchietti, Sangkeum Lee, and Dongsoo Har. 2019. Rewards prediction-based credit assignment for reinforcement learning with sparse binary rewards. *IEEE Access*, 7:118776–118791.

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*.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2025. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, EuroSys '25, page 1279–1297, New York, NY, USA. Association for Computing Machinery.

Joykirat Singh, Raghav Magazine, Yash Pandya, and Akshay Nambi. 2025. Agentic reasoning and tool integration for llms via reinforcement learning. *arXiv preprint arXiv:2505.01441*.

Kimi Team, Angang Du, Bofei Gao, Bowei Xing, Changjiu Jiang, Cheng Chen, Cheng Li, Chenjun Xiao, Chenzhuang Du, Chonghua Liao, and 1 others. 2025. Kimi k1. 5: Scaling reinforcement learning with llms. *arXiv preprint arXiv:2501.12599*.

Shangshang Wang, Julian Asilis, Ömer Faruk Akgül, Enes Burak Bilgin, Ollie Liu, and Willie Neiswanger. 2025. Tina: Tiny reasoning models via lora. *arXiv preprint arXiv:2504.15777*.

Zhiheng Xi, Xin Guo, Yang Nan, Enyu Zhou, Junrui Shen, Wenxiang Chen, Jiaqi Liu, Jixuan Huang, Zhihao Zhang, Honglin Guo, and 1 others. 2025. Bapo: Stabilizing off-policy reinforcement learning for llms via balanced policy optimization with adaptive clipping. *arXiv preprint arXiv:2510.18927*.

Yixuan Even Xu, Yash Savani, Fei Fang, and Zico Kolter. 2025. Not all rollouts are useful: Down-sampling rollouts in llm reinforcement learning. *arXiv preprint arXiv:2504.13818*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, and 1 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 40 others. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, and 1 others. 2025. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*.

Yuanqing Yu, Zhefan Wang, Weizhi Ma, Zhicheng Guo, Jingtao Zhan, Shuai Wang, Chuhan Wu, Zhiqiang Guo, and Min Zhang. 2024. Steptool: A step-grained reinforcement learning framework for tool learning in llms.

Yu Yue, Yufeng Yuan, Qiying Yu, Xiaochen Zuo, Ruofei Zhu, Wenyuan Xu, Jiaze Chen, Chengyi Wang, TianTian Fan, Zhengyin Du, and 1 others. 2025. Vapo: Efficient and reliable reinforcement learning for advanced reasoning tasks. *arXiv preprint arXiv:2504.05118*.

Jianguo Zhang, Tian Lan, Ming Zhu, Zuxin Liu, Thai Hoang, Shirley Kokane, Weiran Yao, Juntao Tan, Akshara Prabhakar, Haolin Chen, and 1 others. 2024. xlam: A family of large action models to empower ai agent systems. *arXiv preprint arXiv:2409.03215*.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, and 1 others. 2025a. Agent learning via early experience. *arXiv preprint arXiv:2510.08558*.

Shaokun Zhang, Yi Dong, Jieyu Zhang, Jan Kautz, Bryan Catanzaro, Andrew Tao, Qingyun Wu, Zhiding Yu, and Guilin Liu. 2025b. Nemotron-research-tool-n1: Exploring tool-using language models with reinforced reasoning. *arXiv preprint arXiv:2505.00024*.

A Dataset

In this section, we give the detailed information of our training dataset in Table 4:

Dataset	Key Contribution	Samples
ToolACE	**Diversity & Complexity**: Covers single, parallel, dependent, and non-tool-use scenarios to teach *when* and *how* to use tools.	2,000
Hammer (Masked)	**Semantic Reasoning**: Uses function masking to force the model to understand API descriptions instead of relying on names.	1,000
xLAM	**Strategic Planning**: Features complex tasks requiring orchestration of multiple tools and management of dependencies.	1,000

Table 4: RL Training Dataset Composition and Rationale. We adopt the data strategy from ToolRL, combining three specialized datasets for a total of 4,000 training samples. Each dataset is chosen to foster a specific capability: ToolACE for foundational diversity, Hammer for semantic reasoning, and xLAM for advanced strategic planning.

Main Category	Subtotal	Evaluation Dimension	Count	Evaluation Purpose
Non-Live (Single-Turn)	**1,390**	Non-Live AST	1,150	To test for **syntactic correctness**.
		Irrelevance Detection (Static)	240	To test for **hallucination avoidance**.
Live (Single-Turn)	**2,251**	Live AST	1,351	To test for **executable correctness**.
		Relevance Detection (Live)	18	To test for **correct tool selection**.
		Irrelevance Detection (Live)	882	To test for **hallucination avoidance in a live context**.
Multi-Turn	**800**	Multi-Turn Dialogue	800	To test for **stateful and sequential reasoning**.

Table 5: A Visually Enhanced Breakdown of the BFCL Benchmark (Revised Layout)

ToolACE (Liu et al., 2024): Build a foundational understanding of diverse and complex tool interactions.

Hammer (Lin et al., 2024): Promote deep semantic reasoning over superficial name-matching.

xLAM (Zhang et al., 2024): Develop advanced strategic planning for multi-step, complex tasks.

BFCL: Analyzing Core Tool Invocation Mechanics. BFCL V3 (Patil et al., 2025) is utilized to measure the fundamental correctness of tool use. Its principal strength is a diagnostic, multi-faceted structure that deconstructs a single tool use into granular dimensions, as detailed in Table 5. This approach is critical for isolating specific failure modes—such as distinguishing an error in tool selection from one in parameter formatting—thereby enabling a precise, component-level analysis of our model's fidelity.

API-Bank: Evaluating Multi-Step Planning and Reasoning. Complementing BFCL's focus on mechanics, the API-Bank benchmark(Li et al., 2023) evaluates higher-order reasoning within realistic, multi-turn dialogues. It employs a hierarchical evaluation that tests progressively complex skills, from basic tool execution to autonomous, multi-step planning. The benchmark's foundation on manually annotated, authentic API interactions allows us to rigorously validate our model's improvements in strategic tool use.

B Hyperparameter Settings

Hyperparameter	Value
Modifications:	
Number of Rollouts	8
Our Method:	
k	7
t_{mean}	0.50
t_{var}	0.10
$\hat{R}_{\text{format}}$	0.90
$\hat{R}_{\text{name}}$	0.75
$\hat{R}_{\text{key}}$	0.70

Table 6: Hyperparameter settings.

B.1 Settings

For our training process, we mainly follow the setting of ToolRL, all the modifications and parameters of our method are shown in the Table 6. Since each epoch in our training process consists of 7 batches, we set $k = 7$ to calculate the statistics over the most recent window of 7 batches, which

epoch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Parameters: $t_{mean} = 0.25$, $t_{var} = 0.1$															
b.2	98.1	96.6	92.4	64.2	19.1	19.4	18.7	17.9	19.0	18.7	18.9	19.1	18.7	18.9	18.4
b.1 & c.1	1.8	3.2	7.3	32.6	44.5	33.8	35.2	36.1	30.5	25.8	21.2	16.2	12.9	11.0	9.6
c.2	0.0	0.0	0.0	0.6	9.6	12.5	11.9	10.9	11.7	12.8	14.2	15.0	15.9	15.9	16.3
a	0.0	0.1	0.3	2.6	26.8	34.2	34.3	35.1	38.8	42.7	45.7	49.7	52.5	54.2	55.7
Parameters: $t_{mean} = 0.50$, $t_{var} = 0.1$															
b.2	98.1	96.6	92.4	64.2	19.8	20.2	19.3	18.6	19.5	19.5	19.7	20.1	19.8	19.9	19.3
b.1 & c.1	1.8	3.2	7.3	32.6	44.5	33.8	35.2	36.1	30.5	25.8	21.2	16.2	12.9	11.0	9.6
c.2	0.0	0.0	0.0	0.5	8.9	11.8	11.2	10.2	11.2	12.0	13.4	14.1	14.8	14.9	15.4
a	0.0	0.1	0.3	2.6	26.8	34.2	34.3	35.1	38.8	42.7	45.7	49.7	52.5	54.2	55.7
Parameters: $t_{mean} = 0.75$, $t_{var} = 0.1$															
b.2	98.1	96.6	92.4	64.3	21.5	22.2	21.2	20.1	21.5	21.4	21.5	22.0	21.7	21.8	21.2
b.1 & c.1	1.8	3.2	7.3	32.6	44.5	33.8	35.2	36.1	30.5	25.8	21.2	16.2	12.9	11.0	9.6
c.2	0.0	0.0	0.0	0.5	7.2	9.8	9.3	8.7	9.2	10.2	11.6	12.2	12.9	13.0	13.5
a	0.0	0.1	0.3	2.6	26.8	34.2	34.3	35.1	38.8	42.7	45.7	49.7	52.5	54.2	55.7
Parameters: $t_{mean} = 1.00$, $t_{var} = 0.1$															
b.2	98.1	96.6	92.4	64.4	23.1	24.7	23.8	22.6	24.3	24.0	24.5	25.3	25.3	25.2	25.1
b.1 & c.1	1.8	3.2	7.3	32.6	44.5	33.8	35.2	36.1	30.5	25.8	21.2	16.2	12.9	11.0	9.6
c.2	0.0	0.0	0.0	0.3	5.6	7.3	6.7	6.2	6.4	7.5	8.6	8.9	9.3	9.6	9.6
a	0.0	0.1	0.3	2.6	26.8	34.2	34.3	35.1	38.8	42.7	45.7	49.7	52.5	54.2	55.7

Table 7: Variation of data difficulty distribution with different values of t_{mean}.

epoch	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Parameters: $t_{mean} = 0.50$, $t_{var} = 0.1$															
b.2	98.1	96.6	92.4	64.2	19.8	20.2	19.3	18.6	19.5	19.5	19.7	20.1	19.8	19.9	19.3
b.1 & c.1	1.8	3.2	7.3	32.6	44.5	33.8	35.2	36.1	30.5	25.8	21.2	16.2	12.9	11.0	9.6
c.2	0.0	0.0	0.0	0.5	8.9	11.8	11.2	10.2	11.2	12.0	13.4	14.1	14.8	14.9	15.4
a	0.0	0.1	0.3	2.6	26.8	34.2	34.3	35.1	38.8	42.7	45.7	49.7	52.5	54.2	55.7
Parameters: $t_{mean} = 0.50$, $t_{var} = 0.5$															
b.2	98.2	96.7	92.7	65.5	25.9	28.5	27.0	26.9	27.5	27.2	27.4	27.8	27.2	26.5	26.2
b.1 & c.1	1.8	3.1	6.9	31.1	36.9	23.6	26.2	25.2	19.6	15.5	12.0	8.0	6.2	5.7	5.4
c.2	0.0	0.1	0.0	0.8	10.4	13.8	12.5	12.8	14.1	14.6	14.9	14.6	14.2	13.6	12.8
a	0.0	0.1	0.3	2.6	26.8	34.2	34.3	35.1	38.8	42.7	45.7	49.7	52.5	54.2	55.7
Parameters: $t_{mean} = 0.50$, $t_{var} = 1.0$															
b.2	98.2	96.7	92.7	65.5	25.9	28.5	24.1	23.2	23.6	22.5	22.6	23.0	27.2	26.5	26.2
b.1 & c.1	1.8	3.1	6.9	31.1	36.9	23.6	27.1	26.5	21.1	17.2	13.6	9.9	6.2	5.7	5.4
c.2	0.0	0.1	0.0	0.8	10.4	13.8	13.4	13.9	15.2	16.1	16.4	15.8	14.2	13.6	12.8
a	0.0	0.1	0.3	2.6	26.8	34.2	35.3	36.4	40.1	44.1	47.5	51.2	52.5	54.2	55.7

Table 8: Variation of data difficulty distribution with different values of t_{var}.

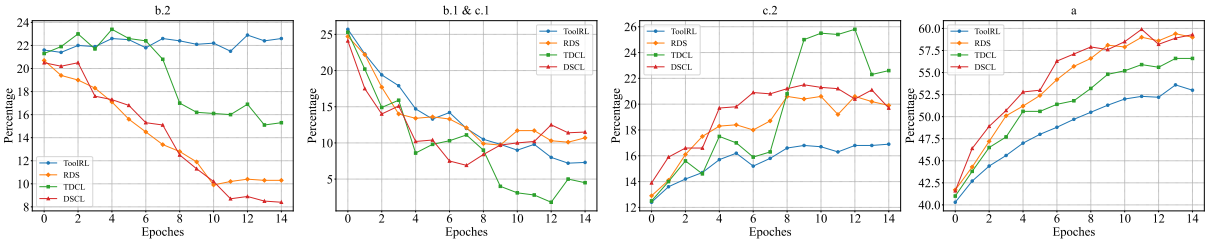

Figure 6: The distribution of data across four difficulty levels during different training epochs on Qwen3-8B.

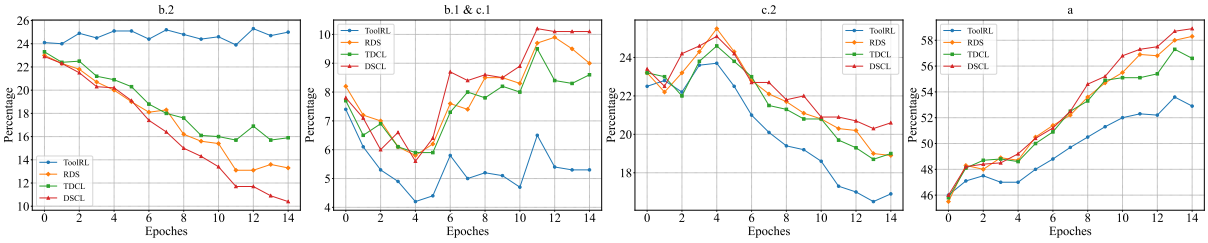

Figure 7: The distribution of data across four difficulty levels during different training epochs on Llama3.1-8B-Instruct.

corresponds to one approximate epoch. All the models are trained with the GRPO method using the verl (Sheng et al., 2025) framework and conducted on 8 H20 GPUs with 96GB of RAM for 15 epochs.

B.2 Experiment on Different Hyperparameters

To determine the optimal parameters for partitioning data by difficulty, we conducte comparative experiments with varying values of t_{mean} and t_{var}. Specifically, for each training epoch, we record the accuracy of all the rollouts for each training data during the ToolRL training process. The parameters, t_{mean} and t_{var}, are used to classify data difficulty based on the rollout performance of each instance. Our method is to leverage these parameters to identify data instances that the model consistently fails to solve under the current training methodology (ToolRL). Therefore, we leverage the training of the Qwen2.5-7B-Instruct model as a case study to demonstrate the results under different parameter settings. As shown in Table 7 and 8, our chosen parameters ($t_{\text{mean}} = 0.50$, $t_{\text{var}} = 0.1$) can stably identify the data that the model consistently fails to solve, which corresponds to the proportion of the b.2 category remaining largely constant. In contrast, for other parameter settings, such as $t_{\text{mean}} = 1.00$, $t_{\text{var}} = 0.1$ and $t_{\text{mean}} = 0.50$, $t_{\text{var}} = 1.0$, the proportion of the b.2 category is unstable during training, even showing a trend of decreasing before increasing. This contradicts our empirical expectation that the proportion of hard examples should remain stable or decrease as training progresses. Based on the analysis above, we finally determine the parameter setting.

C Improvement on Hard Data of Other Foundation Models

In this section, we supplement Figure 6 and 7 with the data distribution across different difficulty levels for the Qwen3-8B and Llama3.1-8B-Instruct models. As can be observed, the overall trend is consistent with that shown in Figure 3, indicating that our method effectively mitigates the model's limitations on difficult data (b.2). A notable difference from Section 6.2 is that due to the stronger capabilities of the Qwen3-8B and Llama3.1-8B-Instruct models, they possess formatting abilities from the very beginning of the training process. Consequently, these two models exhibit a faster convergence speed.

D Tool Complexity and Task Difficulty Analysis

We analyze how task complexity factors affect training performance across epochs. Specifically, we examine two complexity dimensions: (1) the number of available tool APIs and (2) the number of tool parameters. Our analysis covers four task categories: the overall task, tool name selection, parameter key extraction, and parameter value completion.

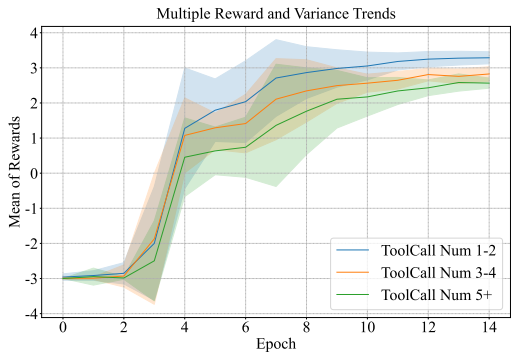

Figure 8: The relationship between training epochs and overall mean reward (the lines) and their corresponding variances (the shaded area) in ToolRL, categorized by the number of tool APIs.

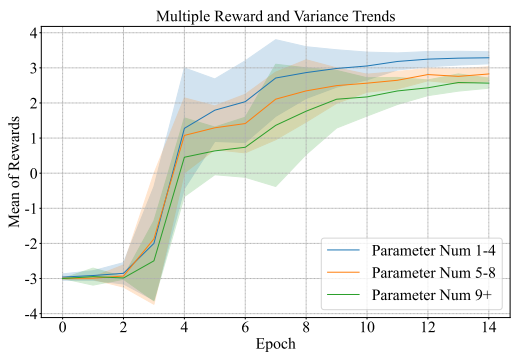

Figure 9: The relationship between training epochs and overall mean reward (the lines) and their corresponding variances (the shaded area) in ToolRL, categorized by the number of tools' parameters.

For the mean reward, both complexity factors show consistent trends across all task categories. As the number of tool APIs or tool parameters increases, mean rewards consistently decrease during training (Figure 8). This suggests that increased tool complexity directly reduces task performance. For the variance of reward, the opposite trend emerges. As the number of tool APIs or tool parameters increases, the reward variance increases consistently across all categories (Figure 9). This indicates that complex tool environments introduce greater performance instability and learning difficulty. Together, these findings demonstrate that task difficulty increases proportionally with both the size of the available tool set and the complexity of individual tool parameters.

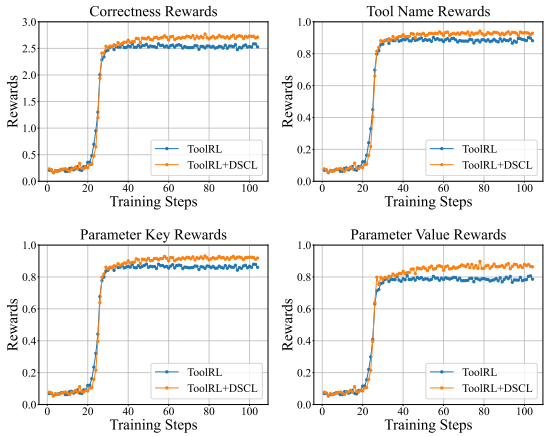

Figure 10: Training rewards of each sub-task. We map the values of each reward to the interval $[0, 1]$ for each sub-task. The correctness rewards represent the total reward of the three sub-tasks.

E Training Analysis

In this section, we will analyze the training process of our method in detail. As shown in Figure 10, we present the reward curves from the baseline ToolRL training compared to the one integrated with our DSCL method. It can be seen that after the introduction of our method, each sub-task can still show an upward trend after the training stabilizes, which demonstrates that our method can continuously provide valuable data to the model through the dynamic sampling strategy. Besides, on the more difficult sub-task, the parameter value task, our method shows a greater improvement compared with the original method, which is due to the fact that our curriculum learning method makes the model pay more attention to this sub-task in the mid-to-late training stage. At the same time, cooperating with the dynamic sampling method to select the corresponding data for training for this more difficult sub-task also helps. This result proves the effectiveness of our method.

F Hyperparameter Sensitivity Analysis

F.1 How default thresholds were determined.

TDCL stage-transition thresholds ($\hat{R}_{\text{format}}$, $\hat{R}_{\text{name}}$, $\hat{R}_{\text{key}}$) were identified by inspecting the per-sub-task reward curves obtained during a preliminary ToolRL baseline. Specifically, we set each threshold at the empirical convergence point of the corresponding reward curve. This data-driven procedure yields $\hat{R}_{\text{format}} = 0.90$, $\hat{R}_{\text{name}} = 0.75$, and $\hat{R}_{\text{key}} = 0.70$.

RDS difficulty thresholds ($t_{\text{mean}}, t_{\text{var}}$) determine the data-partition boundaries used to classify instances into the Easy (a), Hard (b), and Intermedi-

ate (c) buckets, which are defined in Section 4.2. As discussed in Appendix B, we selected the default values (t_{mean}=0.50, t_{var}=0.10) by analysing the per-instance accuracy distribution during ToolRL training. Concretely, we chose the values that make the proportion of consistently-failing instances (category b.2) remain stable throughout training (Tables 7 and 8), which is our key empirical criterion: the hard set should shrink monotonically as the model improves, and not exhibit erratic fluctuation.

F.2 Sensitivity of TDCL to Stage-Transition Thresholds

We perturb each TDCL threshold individually while holding the others at their defaults. We present the results in Tables 9, 10 and 11 on the BFCL V3 benchmark.

TDCL exhibits low sensitivity to threshold variations across all tested settings, indicating that its performance is generally robust to moderate hyperparameter changes. We attribute this stability to the structural design of TDCL: the three-stage curriculum follows the natural dependency order of tool-learning sub-tasks (format → name/key → value), so moderate threshold adjustments mainly affect the timing of stage transitions rather than the effectiveness of the overall learning process.

Relaxing the format threshold leads to a slight performance decline, suggesting that entering the second stage before format compliance is sufficiently stable can introduce minor errors that carry over to later stages. However, the overall effect remains limited, indicating that format learning is acquired relatively early and is not a major source of instability.

A slightly lower tool-name threshold results in comparable or marginally better performance. This suggests that allowing earlier exposure to parameter-value learning, while tool-name matching is still improving, may encourage beneficial joint optimisation across stages. In other words, the transition between the second and third stages appears to tolerate a certain degree of overlap.

Among the three thresholds, the parameter-key threshold is the most sensitive. Lowering it too much consistently harms performance, implying that parameter-key identification is a critical prerequisite for downstream parameter-value learning. If the model enters the final stage before this capability is sufficiently established, errors in key prediction can propagate and weaken subsequent supervision.

F.3 Sensitivity of RDS to t_{mean} and t_{var}

To evaluate the sensitivity of RDS, we vary one hyperparameter at a time while keeping the other fixed at its default value. As shown in Tables 12, 13, and 14, RDS remains effective across all tested configurations, but its performance depends on whether the reward statistics can provide a clear and stable notion of sample difficulty. This finding directly supports the core design intuition of RDS: reward-based data selection is beneficial only when the partition reflects meaningful learning status rather than arbitrary score fluctuations.

Due to the fact that t_{mean} mainly controls the strength of difficulty filtering. The results show that, if it is set too high, too many samples are treated as Hard, which over-reduces the usable training set and weakens generalisation on more complex tasks. If it is set too low, the filtering effect becomes too weak, allowing excessively easy examples to remain and thus diluting the curriculum signal. This indicates that the benefit of RDS does not come from aggressive filtering itself, but from maintaining an appropriate balance between challenging supervision and training-data diversity.

By contrast, t_{var} is the more critical hyperparameter because it determines whether RDS can reliably identify genuinely unstable samples. Increasing t_{var} consistently harms performance, suggesting that an overly loose variance criterion blurs the distinction between uncertain and already-mastered instances. Once this distinction becomes unreliable, the selected data no longer reflects the model's true learning needs, and the curriculum loses its intended effect. This result highlights that the success of RDS relies not only on estimating difficulty, but also on capturing the stability of that difficulty over training.

Overall, the sensitivity study leads to two conclusions. First, RDS is reasonably robust, since its gains persist across a range of settings. Second, its effectiveness is fundamentally tied to the quality of the reward-based partition: RDS works best when the mean reward controls the overall filtering boundary, while variance serves as a reliable indicator of learning uncertainty.

G Prompts

We follow the prompt designation of ToolRL (Qian et al., 2025). The specific prompts for our system and user are as follows.

Configuration	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
TDCL ($\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70)	**59.95**	**86.25**	**76.06**	**18.13**	**83.33**	**79.54**
$\hat{R}_{\text{format}}$=0.85, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70	59.72	86.30	75.98	18.06	83.33	79.03
$\hat{R}_{\text{format}}$=0.80, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70	59.61	86.15	75.80	17.95	83.33	78.85
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.70, $\hat{R}_{\text{key}}$=0.70	59.98	86.47	76.19	18.22	83.33	79.65
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.65, $\hat{R}_{\text{key}}$=0.70	59.88	86.35	76.05	18.10	83.33	79.45
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.65	59.56	86.03	75.73	17.80	83.33	79.42
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.60	58.85	85.45	74.80	17.15	83.33	78.60

Table 9: Sensitivity of TDCL to stage-transition thresholds on Qwen2.5-7B-Instruct. The **bold** row is the default.

Configuration	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
TDCL ($\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70)	**65.83**	**87.15**	**78.72**	**31.88**	**83.33**	**83.79**
$\hat{R}_{\text{format}}$=0.85, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70	65.55	87.22	78.61	31.73	83.33	83.34
$\hat{R}_{\text{format}}$=0.80, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70	65.42	87.03	78.41	31.53	83.33	83.07
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.70, $\hat{R}_{\text{key}}$=0.70	65.88	87.33	78.87	32.00	83.33	83.93
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.65, $\hat{R}_{\text{key}}$=0.70	65.74	87.23	78.67	31.80	83.33	83.68
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.65	65.38	86.90	78.34	31.46	83.33	83.64
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.60	64.58	86.30	77.37	30.73	83.33	82.81

Table 10: Sensitivity of TDCL to stage-transition thresholds on Qwen3-8B. The **bold** row is the default.

Configuration	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
TDCL ($\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70)	**61.98**	**87.20**	**77.41**	**17.79**	**83.33**	**79.30**
$\hat{R}_{\text{format}}$=0.85, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70	61.78	87.24	77.35	17.74	83.33	78.82
$\hat{R}_{\text{format}}$=0.80, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.70	61.68	87.12	77.19	17.64	83.33	78.65
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.70, $\hat{R}_{\text{key}}$=0.70	62.00	87.40	77.51	17.86	83.33	79.40
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.65, $\hat{R}_{\text{key}}$=0.70	61.93	87.29	77.39	17.77	83.33	79.22
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.65	61.63	87.00	77.11	17.51	83.33	79.20
$\hat{R}_{\text{format}}$=0.90, $\hat{R}_{\text{name}}$=0.75, $\hat{R}_{\text{key}}$=0.60	61.03	86.45	76.26	16.89	83.33	78.45

Table 11: Sensitivity of TDCL to stage-transition thresholds on Llama3.1-8B-Instruct. The **bold** row is the default.

Configuration	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
RDS (t_{mean}=0.50, t_{var}=0.10)	**59.55**	**85.54**	**76.10**	**17.38**	**83.33**	**79.86**
Varying t_{mean} (fixed t_{var}=0.10):						
t_{mean}=0.25	59.32	85.73	75.36	16.98	83.33	79.40
t_{mean}=0.75	58.03	85.03	75.38	15.22	83.33	79.48
t_{mean}=1.00	57.56	85.29	75.46	13.58	83.33	79.20
Varying t_{var} (fixed t_{mean}=0.50):						
t_{var}=0.50	57.49	85.27	75.02	13.08	83.33	78.31
t_{var}=1.00	57.12	85.12	74.98	12.78	83.33	78.62

Table 12: Sensitivity of RDS to t_{mean} and t_{var} on Qwen2.5-7B-Instruct. The **bold** row is the default configuration.

Configuration	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
RDS (t_{mean}=0.50, t_{var}=0.10)	**66.21**	**89.93**	**77.70**	**32.88**	**88.89**	**83.07**
Varying t_{mean} (fixed t_{var}=0.10):						
t_{mean}=0.25	65.94	90.08	76.88	32.33	88.89	82.55
t_{mean}=0.75	64.56	89.31	76.85	30.43	88.89	82.62
t_{mean}=1.00	64.06	89.58	76.95	28.73	88.89	82.32
Varying t_{var} (fixed t_{mean}=0.50):						
t_{var}=0.50	63.96	89.61	76.55	28.23	88.89	81.42
t_{var}=1.00	63.56	89.48	76.45	27.73	88.89	81.62

Table 13: Sensitivity of RDS to t_{mean} and t_{var} on Qwen3-8B. The **bold** row is the default configuration.

Configuration	Overal Acc	Non-Live AST Acc	Live Acc	Multi Turn Acc	Relevance Detection	Irrelevance Detection
RDS (t_{mean}=0.50, t_{var}=0.10)	**62.33**	**87.28**	**76.85**	**18.13**	**83.33**	**79.52**
Varying t_{mean} (fixed t_{var}=0.10):						
t_{mean}=0.25	62.08	87.49	76.20	17.68	83.33	79.10
t_{mean}=0.75	60.88	86.83	76.20	16.08	83.33	79.17
t_{mean}=1.00	60.48	87.06	76.30	14.48	83.33	78.90
Varying t_{var} (fixed t_{mean}=0.50):						
t_{var}=0.50	60.38	87.03	75.83	13.98	83.33	78.07
t_{var}=1.00	59.98	86.90	75.77	13.68	83.33	78.37

Table 14: Sensitivity of RDS to t_{mean} and t_{var} on Llama3.1-8B-Instruct. The **bold** row is the default configuration.

Prompt for Our System

You are a helpful multi-turn dialogue assistant capable of leveraging tool calls to solve user tasks and provide structured
chat responses.

Available Tools
In your response, you can use the following tools:
1. Name: getGastroenterologyReport
Description: Retrieve gastroenterology report for a patient
Parameters: {"patient_id": {"description": "The unique identifier of the patient", "type": "string", "default": ""}, ...}
2. ...

Steps for Each Turn
1. **Think:** Recall relevant context and analyze the current user goal.
2. **Decide on Tool Usage:** If a tool is needed, specify the tool and its parameters.
3. **Respond Appropriately:** If a response is needed, generate one while maintaining consistency across user queries.

Output Format
<think> Your thoughts and reasoning </think>
<tool_call>
{"name": "Tool name", "parameters": {"Parameter name": "Parameter content", "... ...": "... ..."}}
{"name": "... ...", "parameters": {"... ...": "... ...", "... ...": "... ..."}}
...
</tool_call>
<response> AI's final response </response>

Important Notes
1. You must always include the '<think>' field to outline your reasoning. Provide at least one of '<tool_call>' or '<response>'. Decide whether to use '<tool_call>' (possibly multiple times),
'<response>', or both.
2. You can invoke multiple tool calls simultaneously in the '<tool_call>' fields. Each tool call should be a JSON object with a "name" field and an "parameters" field containing a dictionary of parameters. If no parameters are needed, leave the "parameters" field an empty dictionary.
3. Refer to the previous dialogue records in the history, including the user's queries, previous '<tool_call>', '<response>', and any tool feedback noted as '<obs>' (if exists).

Prompt for User

Dialogue History
<user> {{ Initial User Input }} </user>

<think> Round 1 Model Thought </think>
{{ Round 1 model output <tool_call> or <response> }}
<obs> Round 1 Observation </obs>
... ...

<user> {{ User Input }} </user>
... ...