PCTD: PREFERENCE-GUIDED COUNTERFACTUAL TASK DECOMPOSITION FOR AGENT TOOL RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Task decomposition aims to transform ambiguous instructions into executable atomic subtasks, thereby guiding high-precision tool retrieval. However, our analysis reveals that directly adopting tool retrieval metrics, i.e., Recall or NDCG, as rewards for task decomposition can easily induce reward hacking in reinforcement learning-based methods. Specifically, models tend to maximize retrieval matching through strategies such as repetitive decomposition.This behavior is associated with degraded Out-of-Domain (OOD) generalization to unseen tools. To address this issue, we propose PCTD, a **P**reference-guided **C**ounterfactual **T**ask **D**ecomposition framework.PCTD uses a paired counterfactual reward to quantify the baseline-relative retrieval gain of decomposition and reduce the attribution of performance already obtainable from the raw query. Meanwhile, it introduces a preference reward to provide structure-aware sequence-level supervision across multiple decomposition-quality dimensions, encouraging the model to generate high-quality decompositions. In addition, we construct MTDTool, the task decomposition benchmark specifically designed for mobile multi-turn interactions. Extensive experiments demonstrate that PCTD alleviates repetitive decomposition and outperforms the evaluated baselines in retrieval, decomposition quality, and OOD generalization. The code is provided in the supplementary materials.

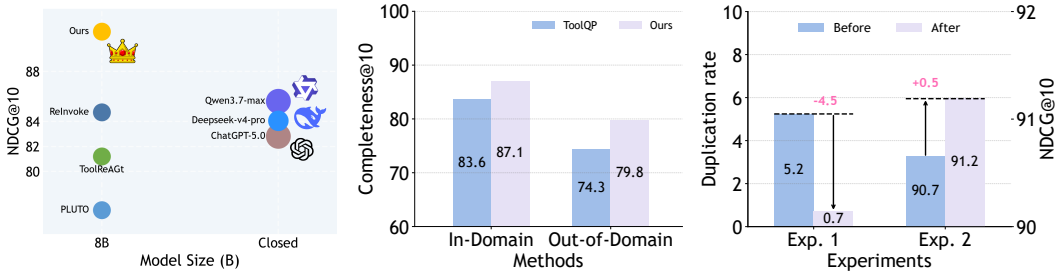


Figure 1: Overall performance and empirical study results. Middle: Performance of different models on In-Domain and OOD settings on MTDTool. Right: Model performance as a function of the decomposition repetition rate. Exp. 1 denotes the repetition rate of task decomposition results before and after applying the preference model, while Exp. 2 represents the change in evaluation metrics before and after applying the preference model.

1 INTRODUCTION

Large language model (Zhao et al., 2026b; Naveed et al., 2025; Wang et al., 2026) (LLM)-driven agents are evolving into intelligent entities capable of autonomously decomposing tasks and invoking tools to achieve complex goals (Huang et al., 2024c; Xu et al., 2024). In multi-turn mobile interaction scenarios, user requests typically rely on dialogue history, coreference relationships, and implicit slots, where a single input is often insufficient to fully describe the objective. This necessitates that agents transform ambiguous instructions into semantically clear, well-bounded, and executable atomic sub-tasks to guide high-precision tool retrieval. Early prompt-engineering based methods, such as ReAct (Yao et al., 2022), ToolReAGt (Braunschweiler et al., 2025), Q2E (Jagerman et al., 2023), and PLUTO (Huang et al., 2024a), achieved progress in few-shot scenarios through explicit chains of thought or structured plans. However, due to the lack of end-to-end optimization

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

mechanisms and heavy reliance on intrinsic knowledge, these methods are prone to decomposition deviations caused by cumulative errors when facing unseen tools or long-range dependencies. To overcome this limitation, reinforcement learning (RL)-based end-to-end methods like ToolQP (Fang & Glass, 2026) incorporated downstream tool retrieval metrics (e.g., Recall@K, NDCG@K) into mixed reward signals.Although ToolQP improves retrieval through interactive query planning and a mixed reward combining retrieval-, format-, and plan-level signals, its retrieval component evaluates the absolute performance of the final aggregated tool list and does not explicitly distinguish the benefit introduced by decomposition from the performance already attainable through direct retrieval.

First, directly employing retrieval metrics as rewards is highly susceptible to **reward hacking**. Models tend to maximize matching scores through superficial strategies such as repetitive decomposition, leading to a spurious correlation between the shallow lexical features of decomposition results and retrieval outcomes. This severely impairs generalization in OOD scenarios involving unseen tools. Second, there is a lack of preference modeling for decomposition structure. Plan-level similarity regularization (e.g., the signal used in ToolQP) does not explicitly assess structural dimensions such as logical coherence, context faithfulness, atomicity, and redundancy, leaving repetitive decompositions insufficiently constrained. Empirical results in Figure 1 validate this phenomenon: existing methods perform reasonably well on in-distribution data but suffer significant performance degradation on OOD datasets involving unseen tools (e.g., ToolQP drops from 83.6 to 74.3). Meanwhile, the reduction in repetition rate from 5.2% to 0.7% is accompanied by an increase in retrieval performance to 91.2. Thus, mitigating reward-hacking-induced spurious correlations and repetitive decomposition is vital for performance.

To address these dual challenges, we propose PCTD, a joint optimization framework for task decomposition and tool retrieval tailored for mobile multi-turn interactions. This framework introduces a preference reward R_{pref}, which leverages a trained reward model to score candidate decompositions against automatically constructed reference decompositions conditioned on context, providing structure-aware sequence-level supervision via relative score differences; simultaneously, it incorporates a counterfactual reward R_{cf} that uses retrieval from the raw query as a prompt-dependent baseline and assigns ranking credit only when decomposition improves over this baseline. Furthermore, existing benchmarks such as HammerBench (Wang et al., 2025) and ToolBench (Qin et al., 2024) lack fine-grained process-level annotations. To provide process-oriented evaluation of task decomposition, we construct MTDTool for complex multi-turn interactions in mobile environments. Unlike existing datasets that primarily focus on single-turn instructions or final execution outcomes, MTDTool automatically annotates dialogue state evolution, context-aware intent rewriting, and atomic task sequences for each conversational turn. The dataset tracks intent evolution in multi-turn, cross-app collaboration, standardizing how decomposition quality impacts retrieval effectiveness.

The contributions of this paper are summarized as follows: ❶ ***Insight.*** We reveal the dual limitations of existing RL-based agents in task decomposition: (i) directly using retrieval metrics as rewards is prone to reward hacking, severely impairing OOD generalization with unseen tools; and (ii) the lack of structural preference modeling further exacerbates repetitive decomposition. ❷ ***Method.*** We propose PCTD, a joint optimization framework integrating counterfactual reasoning with preference modeling. ❸ ***Dataset.*** We release MTDTool, the fine-grained benchmark for task decomposition in mobile multi-turn interactions. Generated via a state-machine-driven pipeline, it provides multi-level annotations for dialogue state evolution, intent rewriting, and atomic task sequences, filling the gap in process-oriented evaluation. ❹ ***Evaluation.*** Extensive experiments on MTDTool and mainstream benchmarks demonstrate that PCTD outperforms the evaluated baselines in tool retrieval, decomposition quality, and OOD generalization.

2 PRELIMINARY

2.1 PROBLEM FORMULATION

We formulate multi-turn agent interaction as a sequential decision process. At turn t, the agent receives a user query q_t and the dialogue history $H_t = \{q_1, o_1, \ldots, q_{t-1}, o_{t-1}\}$, forming the contextual input $x_t = (H_t, q_t)$. Given a large-scale tool library $\mathcal{T} = \{a_1, \ldots, a_M\}$ containing M APIs, the agent employs a policy model π_θ to generate a natural-language task decomposition sequence $y_t \sim \pi_\theta(\cdot \mid x_t)$. The generated sequence is parsed into a set of semantically clear atomic sub-

tasks $\mathcal{A}_t = \{\tau_1, \ldots, \tau_L\}$, which aims to mitigate semantic ambiguity, implicit slots, and cross-turn coreference in direct retrieval. Given a top-K tool retriever $R_K(\cdot)$, each atomic subtask is used to retrieve candidate tools, and the retrieved lists are merged, deduplicated, and truncated into a decomposition-aware ranked list:

$$\mathcal{L}_t^K = \mathrm{TopK}\left(\mathrm{Dedup}\left(\mathrm{Merge}\left(R_K(\tau_1), \ldots, R_K(\tau_L)\right)\right)\right). \tag{1}$$

Let $\mathcal{G}_t \subseteq \mathcal{T}$ denote the ground-truth tool set required for the contextual input x_t. Our objective is to optimize the policy π_θ such that the generated decomposition y_t maximizes the ranking consistency between the retrieved list $\mathcal{L}_t^K$ and the ground-truth tool set $\mathcal{G}_t$.

2.2 Dataset Construction

To address the evaluation needs for fine-grained task decomposition in mobile multi-turn tool invocation, this paper proposes an automated dialogue generation framework driven by a state machine. This framework models multi-turn interactions as a stochastic decision process driven by state evolution, uniformly characterizing context transitions, task decomposition, and tool execution behaviors. In contrast to traditional construction paradigms that focus solely on input-output pairs, our method explicitly models the dynamic dependencies among dialogue states, user intents, and tool calls, thereby generating structured multi-turn corpora with fine-grained structural annotations.

2.2.1 State Space Definition and Dynamic Evolution Mechanism

As illustrated in Figure 3, we model the construction of multi-turn dialogues as a state-driven stochastic evolution process. Let $\mathcal{S}$ denote the dialogue state space, which contains 11 canonical interaction behaviors (see Table 11), including task execution states (e.g., success s_1, in-progress s_2, and failure s_3), interaction states (e.g., clarification s_4 and selection s_5), and intent-management states (e.g., silence s_6, cancellation s_7, revocation s_8, supplementation s_{10}, and null s_{11}). Given a tool library $\mathcal{T}$, each tool a_i is associated with a semantically compatible state subset $\mathcal{S}_{a_i} \subseteq \mathcal{S}$, which constrains the feasible interaction states of that tool. During initialization, the state of each sampled tool is drawn from a tool-conditioned prior:

$$s_{0,i} \sim P_{\mathrm{init}}(\cdot \mid a_i), \quad s_{0,i} \in \mathcal{S}_{a_i}, \tag{2}$$

where P_{init} controls the prior frequency of basic interaction phenomena. We further impose lightweight structural constraints to ensure that the generated dialogues contain sufficient multi-turn interaction signals, such as clarification, continuation, or silence. For subsequent turns, the evolution process is decomposed into context decision, intent transition, and state transition. At turn t, the framework first samples a context decision

$$c_t \sim P_{\mathrm{ctx}}(\cdot \mid t), \quad c_t \in \{\mathrm{inherit}, \mathrm{switch}\}, \tag{3}$$

which determines whether the dialogue continues the previous context or switches to a new topic. For each active tool a_j, the framework then samples a user-side pending intent and derives the corresponding execution state:

$$p_{t,j} \sim P_{\mathrm{intent}}(\cdot \mid s_{t-1,j}, c_t), \tag{4}$$

$$s_{t,j} \sim P_{\mathrm{state}}(\cdot \mid p_{t,j}, s_{t-1,j}), \quad s_{t,j} \in \mathcal{S}_{a_j}. \tag{5}$$

This decoupled formulation separates user intent evolution from assistant execution feedback, making the generation process more controllable. Finally, we apply a lifecycle update to the active tool set: if both the pending intent $p_{t,j}$ and the execution state $s_{t,j}$ are assigned to the null state, tool a_j is removed from the current context.

2.2.2 Context Control and Task Composition Strategy

To simulate topic convergence in real-world dialogues, the framework introduces a context decision variable $c_t \in \{\mathrm{inherit}, \mathrm{switch}\}$ at each evolution node, where the switching probability follows an exponential decay.

$$P_{\mathrm{switch}}(t) = \max(p_{\min}, p_0 \cdot \alpha^{t-1}), \tag{6}$$

where p_0 denotes the baseline probability of topic switching in the initial turn, α is the decay coefficient governing the rate of topic convergence, and $p_{\min}$ sets the lower bound for maintaining

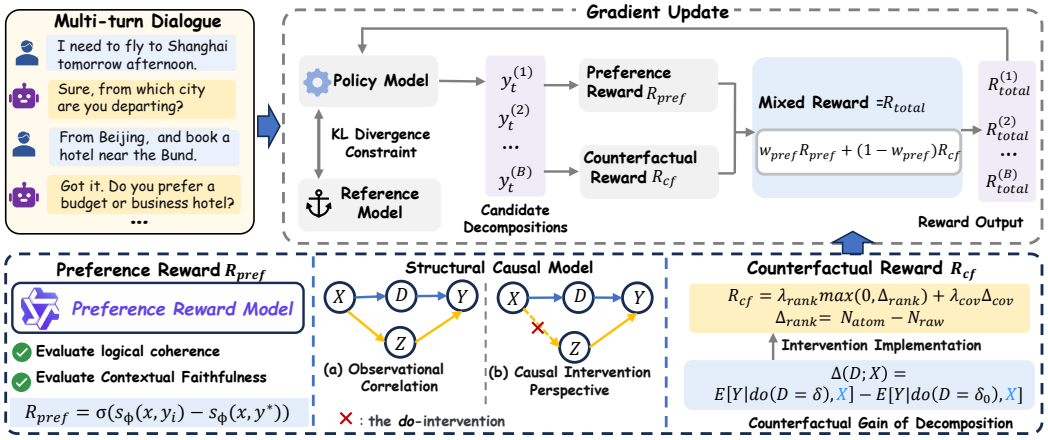


Figure 2: Overview of PCTD.

minimal topic diversity in later dialogue stages. This mechanism ensures high intent variability in early turns while progressively enhancing contextual coherence and stability as the dialogue progresses. At the task structure level, the system supports proportional sampling of three modes: single-tool, multi-tool collaboration, and multi-intent parallelism, while allowing dynamic injection of new tasks during context inheritance. Tool sampling integrates global random weighting with scene-guided knowledge, using pre-defined or LLM-generated templates to ensure domain-specific logical consistency.

2.2.3 Dual-Channel Decoupled Generation and Quality Constraints

Corresponding to the content synthesis module, the framework separately generates user queries q_t and assistant responses o_t through two channels (Appendix G.3). The user-side generator integrates available tool lists, historical state trajectories, and linguistic style features, employing natural language strategies to avoid mechanical concatenation and explicitly completing missing information dimensions in clarification states. The assistant-side generator strictly adheres to state-semantic mapping rules, ensuring precise alignment between response content and current execution states (e.g., success confirmation, failure attribution). To guarantee corpus quality, we embed a structured hard-constraint layer into the generation process, including: (i) the one-tool-one-request atomicity principle; (ii) colloquial conversion of terminology; (iii) normalization of temporal expressions; and (iv) zero-response specifications for silence states. These constraints act as quality filters throughout the generation process, ensuring that the output data possesses both linguistic naturalness and logical rigor.

2.2.4 Fine-Grained Annotation and Evaluation Support

At the output end of the pipeline, an independent task decomposition module maps user requests containing multiple intents into an atomic task sequence $\mathcal{A}_t$ and establishes semantic associations with the candidate tool set $\mathcal{T}_{\text{cand},t}$:

$$q_t, H_t \rightarrow \mathcal{A}_t \rightarrow \mathcal{T}_{\text{cand},t}. \tag{7}$$

This process constructs a three-level annotation structure (*Dialogue Context–Atomic Tasks–Candidate Tools*), which not only supports standard ranking metrics but also precisely locates model bottlenecks such as context misunderstanding, task decomposition errors, or tool retrieval failures. The final dataset comprises Dialogue ID, Turn Index, Scene Label, Interaction Content, Execution States, Intent Trajectories, and Decomposition Results, forming a high-quality structured corpus. A complete example of the multi-turn dialogue generation process is provided in Appendix G.3.

2.3 Joint Reward Mechanism

In the scenario of contextual task atomization, the efficacy of model-generated decomposition results should not be determined solely by superficial textual rationality, but rather by their actual

contribution to downstream tool retrieval performance and the semantic quality of the decomposition structure itself. To this end, we propose a joint reward mechanism that integrates counterfactual feedback with preference alignment, guiding the policy model to learn effective semantic decomposition strategies.

2.3.1 COUNTERFACTUAL BASELINE CONSTRUCTION AND REWARD MECHANISM

Figure 2 provides the causal motivation for our paired comparison. In the diagram, X denotes the multi-turn dialogue input, D the decomposition strategy, Y retrieval-ranking quality, and Z input- and environment-level nuisance factors, including high-frequency keywords, tool priors, and model parametric memory. When absolute Recall is optimized directly, retrieval performance already obtainable from the raw input may be incorrectly attributed to the generated decomposition. We therefore compare the decomposition branch δ with a non-decomposition baseline δ_0 under the same input, tool library, retriever, and ground-truth tool set. Their difference measures the baseline-relative retrieval benefit associated with atomization under this shared-environment assumption, which we denote using paired potential outcomes under a fixed retrieval environment E:

$$\Delta(D; X, E) = Y(\delta; X, E) - Y(\delta_0; X, E), \tag{8}$$

where δ denotes the generated atomized decomposition, δ_0 directly retrieves from the raw context X, and E denotes the fixed tool library, retriever, and ground-truth tool set. Under the paired-environment assumption, $\Delta(D; X, E)$ measures a controlled baseline-relative difference. We use this as an operational counterfactual quantity; the formulation does not claim causal identification beyond the stated assumption, nor does it eliminate candidate-dependent shortcuts introduced by the generated decomposition itself.

In practice, we instantiate the potential outcome Y with the retrieval ranking metric NDCG@K. Given the multi-turn dialogue context $x = (H, q)$ and the ground-truth tool set $\mathcal{G}_t$, we first evaluate the non-decomposition branch by directly retrieving tools with the raw context:

$$\mathcal{L}^K_{\text{raw}} = R_K(x), \quad N_{\text{raw}} = \text{NDCG@K}\left(\mathcal{L}^K_{\text{raw}}, \mathcal{G}_t\right), \tag{9}$$

where $R_K(\cdot)$ denotes the top-K retriever over the tool library $\mathcal{T}$. For the decomposition branch, a candidate decomposition $y_{t,i}$ is parsed into an ordered sequence of atomic subtasks:

$$\mathcal{A}_{t,i} = \text{Parse}(y_{t,i}) = (\tau_{t,i,1}, \tau_{t,i,2}, \ldots, \tau_{t,i,m_{t,i}}). \tag{10}$$

Each subtask is used as an independent retrieval query. The retrieved tool lists are then merged, deduplicated, and truncated to obtain the decomposition-aware ranked list:

$$\begin{aligned}\mathcal{L}^K_{\text{atom}} = \text{TopK}\,(\text{Dedup}\,(\text{Merge}\,(&R_K(\tau_{t,i,1}), \ldots, \\ &R_K(\tau_{t,i,m_{t,i}})))).\end{aligned} \tag{11}$$

The retrieval quality of the decomposition branch is computed as:

$$N_{\text{atom}} = \text{NDCG@K}\left(\mathcal{L}^K_{\text{atom}}, \mathcal{G}_t\right). \tag{12}$$

The paired comparison defines the ranking-level counterfactual gain as:

$$\Delta_{\text{rank}} = N_{\text{atom}} - N_{\text{raw}}. \tag{13}$$

To capture whether decomposition improves complete tool coverage, we further define:

$$\Delta_{\text{cov}} = \mathbb{I}\left[\mathcal{G}_t \subseteq \mathcal{L}^K_{\text{atom}}\right] - \mathbb{I}\left[\mathcal{G}_t \subseteq \mathcal{L}^K_{\text{raw}}\right], \tag{14}$$

where $\mathbb{I}[\cdot]$ is the indicator function. The counterfactual retrieval-gain reward is then formulated as:

$$R_{\text{cf}} = \lambda_{\text{rank}} \max(0, \Delta_{\text{rank}}) + \lambda_{\text{cov}} \Delta_{\text{cov}}. \tag{15}$$

Here, the first term rewards a decomposition only when its NDCG@K exceeds the raw-query baseline, while the second term measures the change in complete tool coverage. The shared retrieval environment makes the raw score an instance-specific reference for improvement. This design reduces attribution to raw-query difficulty but does not by itself prevent candidate-dependent strategies such as redundant or keyword-stacked subtasks.

2.4 Joint Reward Modeling and Policy Optimization

To compensate for the semantic blind spots of pure retrieval metrics, we introduce an independent preference reward R_{pref} based on a specially trained Preference Reward Model (PRM) that evaluates decomposition results across seven dimensions: completeness, accuracy, coreference resolution, standardized expression, contextual consistency, atomicity, and non-redundancy (detailed in Appendix Table 3). Let $s_\phi(x, y)$ denote the raw quality logit assigned by the PRM to a decomposition y under context x, and let $y^\star$ denote an automatically constructed reference decomposition. PRM construction is detailed in Appendix B.2. For each candidate decomposition $y_{t,i}$, we define the preference reward as:

$$R_{\text{pref}}(x_t, y_{t,i}) = \sigma\left(s_\phi(x_t, y_{t,i}) - s_\phi(x_t, y_t^\star)\right), \tag{16}$$

where $\sigma(\cdot)$ is the Sigmoid function. This Bradley–Terry-style reward measures candidate preference against the automatically constructed reference, providing a stable quality anchor for structural supervision. We integrate it with the counterfactual retrieval reward R_{cf} to yield a unified training signal:

$$R = w_{\text{pref}} R_{\text{pref}} + (1 - w_{\text{pref}}) R_{\text{cf}}, \tag{17}$$

where $w_{\text{pref}} \in [0, 1]$ balances structural decomposition quality and counterfactual retrieval gain. To efficiently optimize the policy without a separate critic, we adopt Group Relative Policy Optimization (GRPO) (Guo et al., 2025). For each input x_t, GRPO samples B candidates from the old policy $\pi_{\theta_{old}}$ and estimates the advantage $A_i = \frac{R_i - \text{mean}(\mathbf{R})}{\text{std}(\mathbf{R}) + \epsilon}$ for each via group-wise normalization of their unified rewards $\mathbf{R}$. The final objective maximizes the clipped surrogate loss with a KL penalty:

$$J_{GRPO}(\theta) = \mathbb{E}\left[\frac{1}{B}\sum_{i=1}^{B}\left(\min\left(r_i(\theta)A_i, clip(r_i(\theta), 1-\epsilon, 1+\epsilon)A_i\right) - \beta D_{KL}(\pi_\theta || \pi_{ref})\right)\right], \tag{18}$$

where $r_i(\theta) = \frac{\pi_\theta(y_i|x_t)}{\pi_{\theta_{old}}(y_i|x_t)}$ is the importance sampling ratio, and β controls the KL divergence against a reference model π_{ref}. By integrating our dual rewards into this framework, the policy is driven to produce decompositions that are both semantically well-formed and highly effective for downstream tool retrieval. We further analyze the connections and differences with existing RL-based methods in Appendix F.

3 Experiments

3.1 Datasets and Baselines

Evaluation Benchmarks We evaluate PCTD on two benchmarks: ToolRet (Shi et al., 2025) and MTDTool. ToolRet aggregates 35 widely-used tool-calling datasets spanning three domains (Web, Code, and Custom), with a total corpus of 44K tools. MTDTool is a dataset specifically constructed for complex multi-turn mobile interaction scenarios, covering 237 vertical-domain tools across diverse mobile contexts. The training and test splits of ToolRet strictly follow the setup in ToolQP: we sample 10k instances from Web-domain data and synthesize tool-calling trajectories using Qwen3.6-Plus to form the training set. Separately, 3.1k Web-domain samples form the In-Domain test set, whose tools are exposed during training, while the remaining ToolRet test data constitute the OOD test set. For MTDTool, we partition the test set into In-Domain and OOD subsets based on tool vertical domains. Detailed specifications are provided in the Appendix E.1.

Baselines To comprehensively evaluate task decomposition and tool retrieval performance, we compare against three representative categories of baselines: (1) closed-source LLMs (ChatGPT-5.0[1], Qwen3-series (Yang et al., 2025), MiniMax-M2.5[2], and DeepSeek-V4-Pro[3]), which serve to establish the upper bound of general-purpose models in zero-shot or few-shot decomposition scenarios; (2) prompting-based methods (Q2E (Wang et al., 2023), ReInvoke (Chen et al., 2024), ToolReAGt (Braunschweiler et al., 2025), and PLUTO (Huang et al., 2024b)), which optimize decomposition strategies via explicit chain-of-thought or structured retrieval augmentation, representing

[1] https://chat.openai.com/
[2] https://minimaxi.com/news/minimax-m25
[3] https://www.DeepSeek.com/

Table 1: Overall performance comparison of PCTD and baselines across two benchmarks. Best and second-best results are bolded and underlined, respectively.

Methods	In-Domain				Out-of-Domain				Average	
	ToolRet		MTDTool		ToolRet		MTDTool		Overall	
	N@10	C@10	N@10	C@10	N@10	C@10	N@10	C@10	N@10	C@10
Base Retriever										
Qwen3-4B-embedding	56.82	59.32	68.35	61.79	36.98	38.10	50.57	42.22	53.18	50.36
Closed-Source LLMs										
ChatGPT-5.0	60.30	64.74	82.81	82.50	37.96	39.53	73.22	74.24	63.57	65.25
DeepSeek-V4-Pro	58.23	61.94	83.06	78.70	32.85	33.12	73.26	69.22	61.85	60.75
MiniMax-M2.5	60.32	63.65	84.11	79.18	32.00	32.60	74.51	70.18	62.74	61.40
Prompting Methods (Qwen3-4B)										
Q2E	53.80	56.48	70.36	65.43	29.57	29.88	54.84	46.39	52.14	49.55
ReInvoke	59.55	63.55	85.68	<u>80.26</u>	34.57	34.97	75.08	<u>69.37</u>	63.72	<u>62.04</u>
ToolReAGt	51.79	57.48	77.09	68.81	29.26	34.24	72.54	64.58	57.67	56.28
PLUTO	<u>59.96</u>	<u>65.82</u>	77.69	56.29	34.40	34.24	<u>77.29</u>	58.11	62.34	53.62
RL-Based Methods (Qwen3-4B)										
ToolQP	56.27	62.58	<u>86.72</u>	76.99	<u>37.67</u>	<u>39.93</u>	74.60	68.28	<u>63.82</u>	61.95
Ours	**64.71**	**68.71**	**89.17**	**84.11**	**39.27**	**41.11**	**78.82**	**74.87**	**67.99**	**67.20**
Δ	4.75	2.89	2.45	3.85	1.60	1.18	1.53	5.50	4.17	5.16
Prompting Methods (Qwen3-8B)										
Q2E	55.71	57.74	70.43	66.03	36.47	37.80	54.66	46.32	54.32	51.97
ReInvoke	<u>57.84</u>	61.94	84.74	80.01	34.14	34.51	74.93	70.79	62.91	61.81
ToolReAGt	52.24	58.52	81.20	74.57	31.94	35.08	76.98	70.12	60.59	59.57
PLUTO	56.73	<u>64.25</u>	76.91	61.73	33.16	35.04	75.93	60.36	60.68	55.35
RL-Based Methods (Qwen3-8B)										
ToolQP	56.47	63.23	<u>89.72</u>	<u>83.61</u>	<u>37.77</u>	<u>39.36</u>	<u>77.92</u>	<u>74.33</u>	<u>65.47</u>	<u>65.13</u>
Ours	**64.79**	**68.71**	**91.19**	**87.08**	**39.82**	**41.50**	**82.74**	**79.77**	**69.64**	**69.27**
Δ	6.95	4.46	1.47	3.47	2.05	2.14	4.82	5.44	4.17	4.14

lightweight paradigms without parameter updates; and (3) the RL-based end-to-end optimization method ToolQP (Fang & Glass, 2026), which incorporates downstream tool retrieval metrics into mixed reward signals to jointly optimize decomposition and retrieval processes. Detailed descriptions of all baselines are provided in the Appendix A. Detailed implementation settings are provided in the Appendix B.1.

3.2 Main Results

Table 1 presents a comprehensive performance comparison between PCTD and various baselines on the ToolRet and MTDTool benchmarks. **First**, PCTD improves OOD robustness by optimizing retrieval gains relative to the direct-retrieval baseline rather than rewarding the absolute decomposition score alone. Under the OOD setting of MTDTool, PCTD (Qwen3-8B) achieves an N@10 of 82.74, outperforming the second-best RL baseline ToolQP (77.92) by 4.82. It also achieves an OOD N@10 of 39.82 on ToolRet, surpassing ToolQP (37.77). This pattern is consistent with reduced dependence on training-distribution shortcuts, although performance differences alone do not establish causal invariance. **Second**, tackling repetitive decomposition and reward hacking due to the lack of structural constraints, PCTD achieves a substantial quality leap. In the In-Domain scenario of MTDTool, PCTD (Qwen3-8B) reaches a C@10 of 87.08, outperforming ToolQP (83.61) by 3.47 points. Together with Figure 1, where the reduction in repetition rate from 5.2% to 0.7% is accompanied by an increase in performance to 91.2, these results indicate that the preference reward suppresses redundant subtasks and mitigates reward-hacking behavior. Ultimately, PCTD achieves the best performance across all settings on both benchmarks, with average N@10 and C@10 reaching 69.64 and 69.27, respectively. It comprehensively surpasses prompt engineering and RL baselines, supporting the effectiveness of jointly optimizing baseline-relative retrieval gain and structural quality. Additionally, we observe that some prompting methods (e.g., Q2E, ReInvoke) perform slightly worse on the 8B model than on the 4B model, suggesting fixed prompting strategies are sensitive to model

scale and prone to over-reasoning in larger models. In Table 10, we further analyze PCTD's performance variations with different retrieval models, with detailed results provided in Appendix G.2.

3.3 Ablation Studies

To validate the effectiveness of each component within PCTD, we conducted systematic ablation studies on the MTDTool benchmark. **First**, in the comparison of training paradigms, while the Prompting approach outperforms the base retriever without decomposition, it remains constrained by the model's inherent knowledge, resulting in OOD performance (N@10 of 71.44) that significantly lags behind parameterized methods. Although Supervised Fine-Tuning improves ID performance, its OOD generalization hits a bottleneck (N@10 of merely 75.23) due to overfitting to surface-level patterns. In contrast, the reinforcement learning-based PCTD achieves a substantial improvement via end-to-end optimization, reaching an OOD N@10 of 82.74 (a 7.51-point improvement over SFT). This demonstrates the advantage of end-to-end reward optimization, without by itself establishing that the policy recovers genuine causal structure. **Second**, ablation results validate our dual-track reward design. Removing preference rewards lacks structural constraints on coherence and atomicity, causing a 4.15-point N@10 drop in OOD scenarios. Removing the counterfactual reward causes a larger 5.45-point drop, showing the value of the baseline-relative retrieval signal; this ablation does not by itself identify the mechanism as causal. Ultimately, PCTD optimizes accuracy and generalization by synergizing functional correction with structural constraints.

Table 2: Ablation study on PCTD.

Methods	In-Domain		Out-of-Domain	
	N@10	C@10	N@10	C@10
Base Retriever	68.35	61.79	50.57	42.22
Prompting vs SFT vs RLVR				
+ Prompt	81.27	79.88	71.44	70.52
+ SFT	85.35	79.43	75.23	71.08
+ RLVR	**91.19**	**87.08**	**82.74**	**79.77**
RL-reward Ablation Study				
w/o Preference Reward	90.65	86.18	78.59	75.79
w/o Counterfactual Reward	89.04	84.37	77.29	73.99
PCTD	**91.19**	**87.08**	**82.74**	**79.77**

3.4 Dataset Evaluation

We categorize MTDTool into single-turn and multi-turn dialogues with 10 fine-grained scenario labels, and evaluate the performance under both In-Domain and OOD settings. The experimental results are shown in Table 6. The OOD results are detailed in Appendix D. Scenario-level data reveals clear difficulty gradients. In single-turn settings, performance drops significantly as tool composition complexity increases, declining from an N@10 of 95.48 and a C@10 of 100.00 in the Single Tool scenario to 86.02 and 73.27, respectively, in the Cross-category Multi-tool scenario. In multi-turn settings, structured scenarios such as Intent Selection remain robust, maintaining an N@10 of approximately 95. Conversely, scenarios requiring long-range state tracking exhibit notable degradation; for instance, the C@10 for State Hybrid drops to 70.35, and the N@10 for Multi-turn Continuation falls to 81.29. These results demonstrate that MTDTool is highly discriminative, effectively evaluating the challenges posed by tool composition and multi-turn state evolution. Evaluations regarding OOD robustness and decomposition quality based on the LLM-as-Judge paradigm are provided in Appendix C and D.

3.5 Deeper Analysis

Effect of the raw-query baseline under GRPO. For a fixed prompt x, $N_{\text{raw}}(x)$ and the raw-query coverage indicator are constant across all sampled candidates. Therefore, a purely linear subtraction would not change group-relative advantages: the raw coverage term cancels exactly, and $N_{\text{atom}} - N_{\text{raw}}$ differs from N_{atom} only by a group-wise constant. The operational effect of the baseline arises from the nonlinear term $[N_{\text{atom}} - N_{\text{raw}}]_+$. It converts direct-retrieval performance into a prompt-dependent acceptance threshold, so the ranking component gives credit only to candidates that outperform the raw query. This effect is conditional rather than universal. If every candidate lies above the threshold, the ranking component is equivalent, up to a constant, to optimizing N_{atom}; if every candidate lies below it, the ranking component provides no relative signal. It changes relative advantages when candidates lie on different sides of the threshold. Accordingly, we attribute the

optimization effect to baseline-relative hinge gating, not to subtraction alone, and interpret the paired difference as an operational counterfactual under the shared-environment assumption rather than as strict causal identification.

When might PCTD fail? We further analyze that improper reward weight settings can lead to the failure of PCTD. As shown in Figure 5 (a)-(c), when the preference reward weight $w_{pref} = 0$, the model relies solely on counterfactual signals for optimization. Although it maintains high retrieval accuracy In-Domain (N@10 = 90.6), the duplication rate surges to 5.24%, and OOD performance drops sharply to 78.6. This confirms that hinge-gated retrieval feedback alone is insufficient to prevent repetitive decomposition; the preference reward supplies the structural constraint responsible for the large reduction in repetition. Conversely, an excessively high w_{pref} may slightly sacrifice retrieval accuracy due to over-regularization, despite suppressing redundancy. Thus, balancing baseline-relative retrieval gains with structural constraints is crucial for retrieval accuracy.

4 Related Work

4.1 Planning and Tool Retrieval in Agents

Research on agent planning and tool retrieval primarily follows two paradigms (Huang et al., 2024c; Yehudai et al., 2025): (1) Prompt-based and In-Context Learning approaches, established by ReAct (Yao et al., 2022) and ToolReAct (Braunschweiler et al., 2025). Subsequent works like Toolformer (Schick et al., 2023), Gorilla (Patil et al., 2024), and ToolLLM (Qin et al., 2024) achieve effective tool selection via explicit chains of thought or API sequences; and (2) Joint optimization via Retrieval-Augmented Generation (RAG) (Zhao et al., 2026a) and Reinforcement Learning (Liu et al., 2026). Methods such as Plan-RAG (Verma et al., 2024), TOOLQP (Fang & Glass, 2026), ToolRL (Qian et al., 2026), RLTR (Li et al., 2025), and GiGPO (Feng et al., 2026) decompose instructions into structured sub-queries and refine policies using GRPO (Guo et al., 2025) to enhance long-horizon planning.However, retrieval-reward components in existing RL-based methods optimize the final retrieval outcome without explicitly measuring improvement over direct retrieval. This can leave policies vulnerable to shortcut behaviors, including repetitive decomposition, and weaker OOD generalization. To address this, PCTD combines a prompt-dependent, baseline-relative retrieval reward with preference-based structural constraints, jointly optimizing incremental retrieval benefit and decomposition quality.

4.2 Benchmarks for Tool Retrieval

Existing benchmarks fall into two categories: tool-interaction benchmarks (e.g., API-Bank (Li et al., 2023a), ToolBench (Qin et al., 2024), MINT-Bench (Wang et al., 2024b), τ-Bench (Yao et al., 2024), UltraTool (Huang et al., 2024a), ToolRet (Shi et al., 2025)), focusing on tool selection and complex invocation; and task-oriented benchmarks (e.g., AgentBench (Liu et al., 2024), GAIA (Mialon et al., 2024), WebArena (Zhou et al., 2024), TaskBench (Shen et al., 2024)), emphasizing end-to-end planning and completion rates. Although recent efforts like HammerBench (Wang et al., 2025) extend evaluation to mobile scenarios, mainstream benchmarks remain constrained by Outcome Bias, relying predominantly on final success rates. They critically lack fine-grained process annotations for multi-turn interactions, such as context inheritance, coreference resolution, intent evolution, and task atomization. To bridge this gap, we introduce MTDTool, providing multi-level annotations of intent evolution chains and ranked tool candidates.

5 Conclusion

In this work, we show that directly optimizing retrieval-based rewards is associated with repetitive decomposition and weaker OOD generalization in RL-based task decomposition. We propose PCTD, which combines a baseline-relative counterfactual reward for incremental retrieval improvement with preference rewards for structural supervision. We further introduce MTDTool, a fine-grained benchmark for mobile multi-turn task decomposition. Extensive experiments show that PCTD reduces redundant decomposition and achieves superior performance in tool retrieval, decomposition quality, and OOD robustness.

REFERENCES

Norbert Braunschweiler, Rama Doddipatla, and Tudor-Catalin Zorila. Toolreagt: tool retrieval for llm-based complex task solution via retrieval augmented generation. In *Proceedings of the 3rd Workshop on Towards Knowledgeable Foundation Models (KnowFM)*, pp. 75–83, 2025.

Yanfei Chen, Jinsung Yoon, Devendra Singh Sachan, Qingze Wang, Vincent Cohen-Addad, Mohammadhossein Bateni, Chen-Yu Lee, and Tomas Pfister. Re-invoke: Tool invocation rewriting for zero-shot tool retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4705–4726, 2024.

Wei Fang and James Glass. Beyond single-shot: Multi-step tool retrieval via query planning. *arXiv preprint arXiv:2601.07782*, 2026.

Lang Feng, Zhenghai Xue, Tingcong Liu, and Bo An. Group-in-group policy optimization for llm agent training. *Advances in Neural Information Processing Systems*, 38:46375–46408, 2026.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638, 2025.

Shijue Huang, Wanjun Zhong, Jianqiao Lu, Qi Zhu, Jiahui Gao, Weiwen Liu, Yutai Hou, Xingshan Zeng, Yasheng Wang, Lifeng Shang, et al. Planning, creation, usage: Benchmarking llms for comprehensive tool utilization in real-world complex scenarios. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4363–4400, 2024a.

Tenghao Huang, Dongwon Jung, Vaibhav Kumar, Mohammad Kachuee, Xiang Li, Puyang Xu, and Muhao Chen. Planning and editing what you retrieve for enhanced tool learning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 975–988, 2024b.

Xu Huang, Weiwen Liu, Xiaolong Chen, Xingmei Wang, Hao Wang, Defu Lian, Yasheng Wang, Ruiming Tang, and Enhong Chen. Understanding the planning of llm agents: A survey. *arXiv preprint arXiv:2402.02716*, 2024c.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*, 2023.

Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. Api-bank: A comprehensive benchmark for tool-augmented llms. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 3102–3116, 2023a.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023b.

Zhiwei Li, Yong Hu, and Wenqing Wang. Encouraging good processes without the need for good answers: Reinforcement learning for llm agent planning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1654–1666, 2025.

Shuo Liu, Zeyu Liang, Xueguang Lyu, and Christopher Amato. Llm collaboration with multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 32150–32158, 2026.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, et al. Agentbench: Evaluating llms as agents. In *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024.

Grégoire Mialon, Clémentine Fourrier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: a benchmark for general ai assistants. In *International Conference on Learning Representations*, volume 2024, pp. 9025–9049, 2024.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E Gonzalez. Gorilla: Large language model connected with massive apis. *Advances in Neural Information Processing Systems*, 37:126544–126565, 2024.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tur, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs. *Advances in Neural Information Processing Systems*, 38:105523–105553, 2026.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, et al. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. *Advances in neural information processing systems*, 36:68539–68551, 2023.

Yongliang Shen, Kaitao Song, Xu Tan, Wenqi Zhang, Kan Ren, Siyu Yuan, Weiming Lu, Dongsheng Li, and Yueting Zhuang. Taskbench: Benchmarking large language models for task automation. *Advances in Neural Information Processing Systems*, 37:4540–4574, 2024.

Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. Retrieval models aren't tool-savvy: Benchmarking tool retrieval for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 24497–24524, 2025.

Prakhar Verma, Sukruta Prakash Midigeshi, Gaurav Sinha, Arno Solin, Nagarajan Natarajan, and Amit Sharma. Plan-rag: Planning-guided retrieval augmented generation. 2024.

Jun Wang, Jiamu Zhou, Xihuai Wang, Xiaoyun Mo, Haoyu Zhang, Qiqiang Lin, Jincheng Jincheng, Muning Wen, Weinan Zhang, and Qiuying Peng. Hammerbench: Fine-grained function-calling evaluation in real mobile assistant scenarios. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 3350–3376, 2025.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9414–9423, 2023.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11897–11916, 2024a.

Peng-Yuan Wang, Tian-Shuo Liu, Chenyang Wang, Ziniu Li, Yidi Wang, Shu Yan, Chengxing Jia, Xu-Hui Liu, Xinwei Chen, Jiacheng Xu, et al. A survey on large language models for mathematical reasoning. *ACM Computing Surveys*, 58(8):1–35, 2026.

Xingyao Wang, Zihan Wang, Jiateng Liu, Yangyi Chen, Lifan Yuan, Hao Peng, and Heng Ji. Mint: Evaluating llms in multi-turn interaction with tools and language feedback. In *International Conference on Learning Representations*, volume 2024, pp. 32593–32627, 2024b.

Qiancheng Xu, Yongqi Li, Heming Xia, and Wenjie Li. Enhancing tool retrieval with iterative feedback from large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9609–9619, 2024.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. tau-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

Asaf Yehudai, Lilach Eden, Alan Li, Guy Uziel, Yilun Zhao, Roy Bar-Haim, Arman Cohan, and Michal Shmueli-Scheuer. Survey on evaluation of llm-based agents. *arXiv preprint arXiv:2503.16416*, 2025.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. Retrieval-augmented generation for ai-generated content: A survey. *Data Science and Engineering*, pp. 1–29, 2026a.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, et al. A survey of large language models. *Frontiers of Computer Science*, 20(12):2012627, 2026b.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

APPENDIX

Contents

Appendix **13**

A Baselines **14**

A.1 Closed-Source LLMs 14

A.2 Prompting Methods 14

A.3 RL-Based Method 15

A.4 Re-ranking methods 15

B Implementation Details **15**

B.1 PCTD Implementation Details 15

B.2 Preference Reward Model Construction 16

C Evaluation of Task Decomposition Quality **16**

C.1 GSB Evaluation Process and Scheme 16

C.2 Analysis of GSB Evaluation Results 16

D Further Dataset Evaluation in OOD **18**

E Further Analysis of Dataset **18**

E.1 Statistics 18

E.2 Dataset distribution information 18

F Connections and Differences with Existing RL-based Methods **20**

G Further Analysis **21**

G.1 When might PCTD fail? 21

G.2 Performance of PCTD Across Different Retrievers 23

G.3 Training Curves Analysis 24

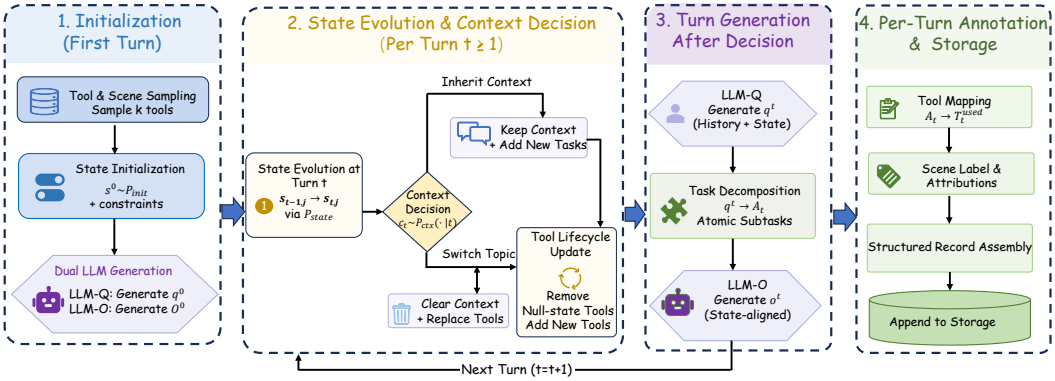


Figure 3: **Overview of the MTDTool construction pipeline.** The data generation follows a four-stage process. **(1) Initialization**: Tools and scenarios are sampled to establish initial constraints, and two LLM generators produce the initial user query q^0 and assistant output o^0. **(2) State Evolution & Context Decision**: At each turn $t \geq 1$, the system decides whether to inherit the previous context or switch to a new topic, and updates the pending intent $p_{t,j}$ and execution state $s_{t,j}$ for each active tool. **(3) Turn Generation**: Conditioned on the dialogue history H_t, current states, and active tools, the framework generates the user query q_t, derives the atomic task sequence $\mathcal{A}_t$, and produces the state-aligned assistant output o_t. **(4) Per-Turn Annotation & Storage**: The framework maps the atomic task sequence $\mathcal{A}_t$ to the used tool set $\mathcal{T}_t^{\text{used}}$, assigns scenario labels and attribution information, and stores the structured record. This pipeline preserves the coherence of multi-turn intent evolution while providing fine-grained annotations for task decomposition and tool retrieval.

A Baselines

A.1 Closed-Source LLMs

- **Qwen3 Series** (Yang et al., 2025) is a series of LLMs developed by the Alibaba team. It includes the general-purpose language model Qwen3, the advanced reasoning-specific model Qwen3-Reasoner, and the agent-oriented model Qwen3-Agent. The Qwen3 series of models have been pretrained on significantly larger and higher-quality datasets with extended context windows. In addition, it achieves state-of-the-art performance in complex tasks such as autonomous agent planning, multi-step mathematical reasoning, and precise instruction following.
- **ChatGPT-5.0** is a large-scale language model developed by OpenAI based on the Transformer architecture. It demonstrates strong capabilities in natural language understanding, reasoning, and text generation, enabling it to handle complex dialogue and knowledge-intensive tasks.
- **DeepSeek-V4 Series** is a state-of-the-art open-weight large language model built on a Mixture-of-Experts (MoE) architecture with auxiliary-loss-free load balancing and FP8 mixed-precision training. It achieves superior reasoning and instruction-following performance while maintaining high computational efficiency through sparse activation and optimized inference kernels.
- **MiniMax-M2.5** is a high-performance proprietary large language model featuring an optimized Transformer architecture with enhanced long-context processing and multi-modal alignment capabilities.

A.2 Prompting Methods

- **Q2E** (Wang et al., 2023) is a semantic query expansion mechanism based on Large Language Models (LLMs). For short and ambiguous user queries, Q2E leverages the reasoning capabilities of LLMs to reconstruct them into content-rich pseudo-documents, thereby improving the relevance ranking performance of retrieval systems without requiring additional supervision signals.
- **ReInvoke** (Chen et al., 2024) proposes a zero-shot tool retrieval paradigm based on Large LLMs, the essence of which is to decompose user intent into tool-executable semantic units. By reformulating raw queries into structured instructions aligned with tool descriptions, Re-invoke achieves fine-grained mapping from abstract requirements to specific tool parameters. This approach ef-

fectively bridges the semantic gap between user queries and tool interfaces without requiring fine-tuning, thereby enhancing tool retrieval performance in complex multi-task scenarios.

- **ToolReAGt** (Braunschweiler et al., 2025) constructs an iterative retrieval framework based on the ReAct paradigm, core to which is the mandatory embedding of tool retrieval within the Thought-Action-Observation loop. For each sub-task, the Agent generates a tailored query via Thought and mandatorily invokes the retriever to obtain candidate tools. Subsequently, in the Observation phase, it evaluates matching relevance; if no suitable tool is found, it leverages feedback to enter the next iteration for query refinement.
- **PLUTO** (Huang et al., 2024b) employs a post-retrieval planning and editing mechanism. It initially retrieves a candidate set of tools based on the user query, and subsequently refines the results through a two-step process leveraging LLMs. First, a planning module filters the most relevant tool subset from the candidates and determines their invocation order according to the current task context. Second, an editing module instantly corrects and adapts the documentation or parameter examples of the selected tools to eliminate semantic discrepancies between the original descriptions and specific task requirements.

A.3 RL-Based Method

- **ToolQP** (Fang & Glass, 2026) reformulates tool retrieval as an iterative planning process and incorporates Reinforcement Learning with Verifiable Rewards (RLVR) to optimize the planning policy. During inference, the model decomposes abstract intents into structured sub-goals with specific parameters and preconditions via sub-goal decomposition, generating targeted queries accordingly. During training, ToolQP employs a mixed reward consisting of retrieval quality, output-format correctness, and plan-level semantic consistency. Its retrieval reward is computed from Recall@K and NDCG@K over the final aggregated tool list.

A.4 Re-ranking methods

- **Qwen3-Embedding series** (Zhang et al., 2025)is a high-performance text embedding model built upon the Qwen3 dense Transformer architecture, optimized specifically for semantic representation and retrieval tasks. It employs contrastive learning with hard negative mining and multi-stage training to achieve superior performance on MTEB benchmarks while maintaining efficient inference through knowledge distillation from larger teacher models.
- **gte-Qwen2-1.5B-instruct** (Li et al., 2023b) is a compact text embedding model based on the Qwen2-1.5B dense Transformer architecture, fine-tuned specifically for semantic retrieval and representation tasks. It achieves competitive MTEB performance through contrastive learning with hard negative mining and instruction-aware training, while maintaining high inference efficiency suitable for resource-constrained deployment scenarios.
- **e5-mistral-7b-instruct** (Wang et al., 2024a) is a text embedding model built upon the Mistral-7B dense Transformer architecture, fine-tuned with instruction-aware contrastive learning for semantic retrieval tasks. It achieves strong performance on MTEB benchmarks by leveraging the base model's reasoning capabilities and optimizing representation quality through hard negative mining and multi-stage training.

B Implementation Details

B.1 PCTD Implementation Details

Our PCTD framework is trained in two stages. We apply LoRA adaptation (rank=64, alpha=128) to Qwen3-4B and Qwen3-8B base models initialized from SFT checkpoints. The training configuration uses a batch size of 128, a PPO mini-batch size of 64, and a PPO micro-batch size of 8 per GPU. The maximum prompt length is set to 1024 and the maximum response length to 128.

During GRPO rollout, we sample 8 candidate responses per prompt ($n = 8$) and employ vLLM as the inference engine with a GPU memory utilization of 0.6. The multi-signal reward consists of two components: a preference reward with weight w_{pref} and a counterfactual reward with weight $1-w_{pref}$. The preference reward is provided by a Qwen2.5-1.5B reward model. We set $w_{\text{pref}} = 0.2$

to balance the preference reward and the counterfactual retrieval-gain reward. Within R_{cf}, we set $\lambda_{\mathrm{rank}} = \lambda_{\mathrm{cov}} = 1.0$ unless otherwise specified.

For the main experiments, we retrieve the top-10 tools for each query. The cross-retriever analysis in Table 10 is separately evaluated with $K = 15$. Following standard tool-recall practice, each decomposed subtask independently retrieves its top-K candidate tools, and the results are merged via round-robin ranking with deduplication.

B.2 Preference Reward Model Construction

We construct the preference data entirely through the automated MTDTool generation pipeline, without additional human annotation. For each dialogue context, the pipeline produces a preference pair containing a structurally correct decomposition y^+ as the preferred response and an automatically generated erroneous decomposition y^- as the rejected response. The negative decompositions cover violations of the target quality dimensions, including missing intents, incorrect context inheritance, unresolved coreferences, redundant subtasks, and inappropriate task granularity. We train Qwen2.5-1.5B as the PRM using the pairwise preference objective

$$\mathcal{L}_{\mathrm{PRM}} = -\mathbb{E}_{(x,y^+,y^-)} \left[\log \sigma \left(s_\phi(x, y^+) - s_\phi(x, y^-) \right) \right]. \tag{19}$$

The same trained PRM is used during PCTD policy optimization on both MTDTool and ToolRet; no dataset-specific PRM is trained for ToolRet. For MTDTool, $y^\star$ is produced directly by the automated generation pipeline; for ToolRet, it is obtained by applying the same automated reference-construction and filtering procedure to each training instance while keeping the PRM parameters fixed. Thus, PRM training, reference construction, and preference supervision rely on the automated pipeline rather than human-written decompositions. **PRM training details.** We initialize the PRM from Qwen2.5-1.5B-Instruct and attach a linear scalar reward head to the final hidden representation. The model is trained using LoRA with rank 32, scaling factor 64, and dropout 0.05. We optimize the pairwise Bradley–Terry loss using AdamW with a learning rate of 2×10^{-5}, weight decay of 0.01, a cosine learning-rate schedule, and a warmup ratio of 0.03. The effective batch size is 128 preference pairs, the maximum sequence length is 1536, and gradients are clipped to 1.0. Training is conducted in BF16 for up to three epochs, with early stopping based on pairwise accuracy on a context-disjoint validation set.

C Evaluation of Task Decomposition Quality

C.1 GSB Evaluation Process and Scheme

To comprehensively evaluate the model's performance in multi-turn intent rewriting and task decomposition, we established a General Scoring Benchmark (GSB) based on the *LLM-as-Judge* paradigm. Adopting an **absolute scoring** mode, we randomly sampled 300 instances each from In-Domain and Out-of-Domain datasets. Three state-of-the-art closed-source LLMs (Qwen3.6-Plus, GLM-5.1, and DeepSeek-v4-flash) served as independent judges, with their assessments aggregated via majority or weighted voting to mitigate individual model biases. The evaluation framework comprises two core dimensions, *Rewriting Quality* and *Decomposition Quality*, each assigned a 50% weight. Specifically, Rewriting Quality is fine-grainedly assessed based on conciseness-fidelity (0.2), context coherence (0.3), and multi-task handling capability (0.5); while Decomposition Quality focuses on granularity appropriateness (0.5), logical consistency (0.2), and completeness (0.3). Finally, we quantified the performance gap between our method (Ours) and mainstream closed-source baselines by aggregating the weighted 5-point scale scores across all dimensions.

C.2 Analysis of GSB Evaluation Results

As shown in Table 4, PCTD matches the strongest evaluated closed-source baseline in the In-Domain setting (4.36) and obtains a slightly higher score in the OOD setting (4.37 versus 4.34). These results indicate competitive multi-turn intent-rewriting and task-decomposition quality under the adopted LLM-as-Judge protocol. Furthermore, ablation studies support the contribution of the baseline-relative retrieval component: compared to the variant using only the preference reward (4.01/4.00), the complete PCTD method yields score improvements of 0.35 and 0.37 on In-Domain and OOD

Table 3: Evaluation Dimensions and Penalty Rules of the Preference Reward Model (PRM).

Dimension	**Description**	**Penalty Rule**
Completeness	Whether all valid intents, core entities, and constraints from the original utterance are included without omission.	Severe penalty for missing intents, entities, or constraints.
Accuracy	Whether the user's true intent is precisely restored without semantic deviation.	Severe penalty for semantic misinterpretation or non-executable tasks.
Coreference Resolution	Pronouns (e.g., "it", "that") must be replaced with explicit entities to ensure standalone readability.	Moderate penalty for unresolved coreferences.
Standardized Expression	Colloquialisms and filler words are removed; output should be concise and standardized instructions.	Minor penalty for non-standard or redundant expressions.
Contextual Consistency	Historical information is correctly inherited across turns; modifications and conflicts are properly handled.	Severe penalty for incorrect context handling.
Atomicity	Whether each subtask expresses one executable intent at an appropriate granularity.	Penalty for merging independent intents or fragmenting one intent into unnecessary subtasks.
Non-redundancy	Whether the decomposition avoids duplicated or semantically overlapping subtasks.	Penalty for repeated or overlapping subtasks.

Table 4: GSB evaluation results on In-Domain and Out-of-Domain datasets. Scores are on a 5-point scale. **Bold** indicates the best performance.

Model	**In-Domain**	**Out-of-Domain**
Closed-Source LLMs		
MiniMax-M2.5	4.19	4.17
DeepSeek-v4-pro	4.27	4.25
ChatGPT-5.0	**4.36**	4.34
Ours (RLVR)		
PCTD w/o Counterfactual Reward	4.01	4.00
PCTD	**4.36**	**4.37**

datasets, respectively. This gain supports the usefulness of baseline-relative retrieval feedback when combined with structural supervision; it should not be interpreted as evidence that the method identifies a causal effect or fully removes shortcut behavior.

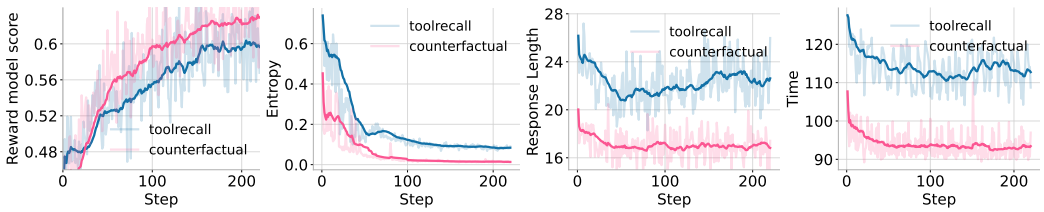
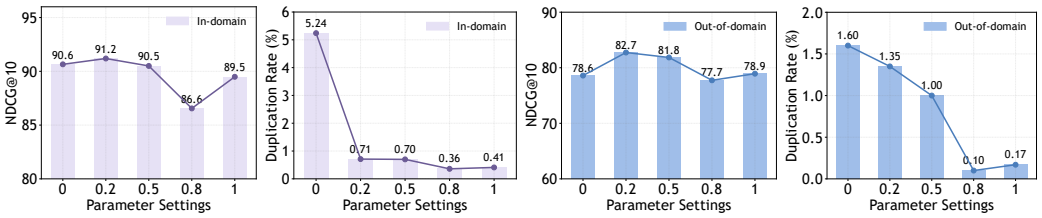


Figure 4: Training curves of different metrics.



(a) In-Domain (b) In-Domain (c) Out-of-Domain (d) Out-of-Domain

Figure 5: Hyperparameter analysis under different w_{pref} settings.

Table 5: Scenario-level difficulty and discriminative capability of MTDTool in the Out-of-Domain setting.

Scenario	**Type**	Out-of-Domain	
		N@10	C@10
Single-turn	Single Tool	81.95	95.36
	Same-category Multi-tool	77.77	63.78
	Cross-category Multi-tool	74.41	55.48
Multi-turn	Domain Switch	81.27	93.08
	Multi-turn Continuation	69.31	74.79
	Intent Selection	82.88	94.13
	User Adds New Task	84.64	89.51
	Unfinished Task Continuation	73.00	64.64
	Abnormal Scenario	77.89	81.65
	State Hybrid	76.79	53.36

D Further Dataset Evaluation in OOD

Out-of-Domain Performance and Robustness Challenges: As detailed in Table 5, the introduction of unseen tools leads to significant performance degradation, validating the effectiveness of MTDTool in evaluating OOD generalization capabilities. The overall average N@10 drops from 89.05 to 78.04. This regression is particularly pronounced in complex scenarios: the N@10 for the Cross-category Multi-tool scenario further declines to 74.41, and for the State Hybrid scenario, which demands rigorous state tracking, the C@10 drops sharply to 53.36. These results demonstrate that MTDTool provides a fine-grained and highly discriminative benchmark, effectively evaluating task decomposition and tool retrieval capabilities under the challenges of tool composition complexity, domain shifts, and multi-turn state evolution.

E Further Analysis of Dataset

E.1 Statistics

The constructed dataset covers seven vertical tool categories, including daily life, travel, utilities, device control (system or application control), reminders, communication, and audio-video entertainment. Based on domain relevance, the dataset is divided into In-Domain and Out-of-Domain subsets: the In-Domain subset includes three verticals (daily life, travel, and utilities) involving 36 tools, while the Out-of-Domain subset covers the remaining four verticals with 201 tools. The dataset comprises a total of 2,917 training samples, and the test set consists of 6,058 In-Domain samples and 5,201 Out-of-Domain samples. During the retrieval evaluation phase, the full tool pool is employed, and only the `tool_description` field is used for text embedding representation.

E.2 Dataset distribution information

We categorize the dataset into two major groups based on the dialogue interaction structure and task execution states: Single-turn and Multi-turn scenarios. On this basis, we further define several specific scenario types to systematically cover diverse interaction patterns that may arise during tool

Table 6: Scenario-level difficulty and discriminative capability of MTDTool in the In-Domain setting.

Scenario	Type	In-Domain	
		N@10	C@10
Single-turn	Single Tool	95.48	100.00
	Same-category Multi-tool	93.99	95.42
	Cross-category Multi-tool	86.02	73.27
Multi-turn	Domain Switch	91.78	95.89
	Multi-turn Continuation	81.29	83.83
	Intent Selection	95.09	99.29
	User Adds New Task	94.31	98.58
	Unfinished Task Continuation	90.71	87.28
	Abnormal Scenario	90.20	92.68
	State Hybrid	88.48	70.35

usage. The detailed classification results are presented in Table 8. As illustrated in Figure 6, the proportions of different categories in the training set and test set are presented in detail. Figure 7 further shows the distribution of different categories across the entire dataset.

Algorithm 1 Counterfactual and Preference Rewards for Task Decomposition

Input: SFT policy π_θ, old policy $\pi_{\theta_{\text{old}}}$, reference policy π_{ref}, preference model s_ϕ, tool retrieval service $\mathcal{S}_{\text{tr}}$, weights $w_{\text{pref}}, \lambda_{\text{rank}}, \lambda_{\text{cov}}$

Output: Optimized policy π_θ

for each training step **do**

  Sample n decompositions $\{y_t^{(j)}\}_{j=1}^n \sim \pi_{\theta_{\text{old}}}(\cdot \mid x_t)$ for each prompt $x_t = (H_t, q_t)$

  Raw-query baseline (computed once per prompt): $\mathcal{L}_{\text{raw}}^K \leftarrow \mathcal{S}_{\text{tr}}(x_t, K)$

  Counterfactual reward: for each $y_t^{(j)}$,

    $\mathcal{A}_t^{(j)} \leftarrow \text{Parse}(y_t^{(j)})$

    Retrieve each subtask independently, then merge, deduplicate, and truncate the lists to obtain $\mathcal{L}_{\text{atom}}^{K,(j)}$

    $\Delta_{\text{rank}}^{(j)} \leftarrow \text{NDCG@}K(\mathcal{L}_{\text{atom}}^{K,(j)}, \mathcal{G}_t) - \text{NDCG@}K(\mathcal{L}_{\text{raw}}^K, \mathcal{G}_t)$

    $\Delta_{\text{cov}}^{(j)} \leftarrow \mathbb{I}[\mathcal{G}_t \subseteq \mathcal{L}_{\text{atom}}^{K,(j)}] - \mathbb{I}[\mathcal{G}_t \subseteq \mathcal{L}_{\text{raw}}^K]$

    $R_{\text{cf}}^{(j)} \leftarrow \lambda_{\text{rank}} \max(0, \Delta_{\text{rank}}^{(j)}) + \lambda_{\text{cov}} \Delta_{\text{cov}}^{(j)}$

  Preference reward: $R_{\text{pref}}^{(j)} \leftarrow \sigma\Big(s_\phi(x_t, y_t^{(j)}) - s_\phi(x_t, y_t^\star)\Big)$

  Multi-signal fusion: $R^{(j)} \leftarrow w_{\text{pref}} \cdot R_{\text{pref}}^{(j)} + (1 - w_{\text{pref}}) \cdot R_{\text{cf}}^{(j)}$

  GRPO advantage: normalize within the n-candidate group,

  $$\hat{A}^{(j)} \leftarrow \frac{R^{(j)} - \frac{1}{n}\sum_{k=1}^n R^{(k)}}{\text{std}(\{R^{(k)}\}_{k=1}^n) + \epsilon_{\text{std}}}$$

  Policy update: minimize

  $$L \leftarrow -\mathbb{E}\Big[\min\big(\rho^{(j)}\hat{A}^{(j)},\ \text{clip}(\rho^{(j)}, 1-\epsilon_{\text{clip}}, 1+\epsilon_{\text{clip}})\hat{A}^{(j)}\big)\Big] + \beta_{\text{KL}} \cdot \text{KL}\big(\pi_\theta \,\|\, \pi_{\text{ref}}\big)$$

  where $\rho^{(j)} = \pi_\theta(y_t^{(j)} \mid x_t) / \pi_{\theta_{\text{old}}}(y_t^{(j)} \mid x_t)$

end for

return π_θ

In single-turn scenarios, the user request is completed within a single dialogue turn. This category includes three types: Single-turn Single Tool, Single-turn Multi-tool (Same Category), and Single-turn Cross-category Multi-tool. Specifically, Single-turn Single Tool refers to tasks that can be completed by invoking only one tool in a single interaction, representing the most basic form of tool usage. Single-turn Multi-tool (Same Category) describes situations where multiple tools are required within a single turn, but these tools belong to the same functional category, such as multiple tools related to travel services. Single-turn Cross-category Multi-tool refers to cases where multiple

tools from different domains or categories are required within one turn, for example, simultaneously checking the weather and booking a hotel.

In multi-turn scenarios, the dialogue consists of multiple interaction turns, and the execution state of tasks may evolve or be inherited across turns. We further define several common multi-turn interaction types, including Domain Switching, Exception Scenario, Unexecuted Task Inheritance, Follow-up Selection, User-added Task, Mixed State Type, and Multi-turn Inheritance. Domain Switching refers to situations where the current turn abruptly shifts to a completely new topic or domain that is unrelated to the previous turn. Exception Scenario describes cases where abnormal execution states occur during task execution, such as failures, cancellations, re-executions, or rollbacks. Unexecuted Task Inheritance indicates that a silent task from the previous turn is still running in the background and its state must be tracked or inherited in the current turn. Follow-up Selection refers to interactions where the assistant asks the user for missing information or clarification, or where the user selects among multiple options. User-added Task describes situations in which the user introduces a new task during the dialogue that is unrelated to the previous task. Mixed State Type refers to dialogues that simultaneously involve three or more of the above multi-turn interaction types (excluding multi-turn inheritance). Finally, Multi-turn Inheritance represents the normal continuation of an unfinished task from the previous turn into the current turn. Through this categorization, the dataset aims to comprehensively cover interaction patterns ranging from basic tool invocation to complex multi-turn task management, thereby providing a more fine-grained foundation for evaluating and analyzing model capabilities.

F Connections and Differences with Existing RL-based Methods

Although both PCTD and ToolQP employ reinforcement learning to optimize tool retrieval, they differ substantially in their optimization objectives and training paradigms. ToolQP focuses on interactive query planning, iteratively generating subsequent queries through multiple rounds of planning, retrieval, and feedback, and primarily performs sequence-level optimization based on the absolute retrieval performance of the final aggregated tool list. This paradigm requires multi-step interactions with the retrieval environment. Moreover, absolute retrieval rewards cannot distinguish the performance already attainable from the raw input from the additional benefit introduced by task decomposition, potentially encouraging reward-hacking behaviors and impairing OOD generalization. In contrast, PCTD shifts the optimization target to contextual task atomization before retrieval. Under the same retrieval environment, it uses the retrieval performance of the raw query as an instance-dependent baseline and applies a prompt-dependent hinge gate, assigning ranking credit only to decompositions that outperform direct retrieval. Meanwhile, a PRM provides structure-aware assessments of decomposition completeness, atomicity, and non-redundancy. By jointly optimizing baseline-relative retrieval gains and decomposition quality, PCTD avoids iterative planning conditioned on multi-round retrieval feedback and improves retrieval robustness in the evaluated OOD settings.

Table 7: Comparison of MTDTool with existing tool-use benchmarks across different capability dimensions.

Dimension	Ours	Hammer	ToolRet	APIBench	ToolBench	API-Bank
Single-turn Single Tool	✓	✓	✓	✓	✓	✓
Single-turn Multiple Tools	✓	×	✓	×	✓	✓
Single-turn Cross-domain Multiple Tools	✓	×	✓	×	✓	✓
Multi-turn Inheritance	✓	✓	×	×	✓	✓
Switching	✓	✓	×	×	×	×
Exception Scenarios	✓	×	×	×	×	×
User Follow-up Tasks	✓	×	×	×	×	×
Unfinished Task Inheritance	✓	×	×	×	×	×
Follow-up Selection	✓	✓	×	×	×	×
Mixed State Types	✓	×	×	×	×	×

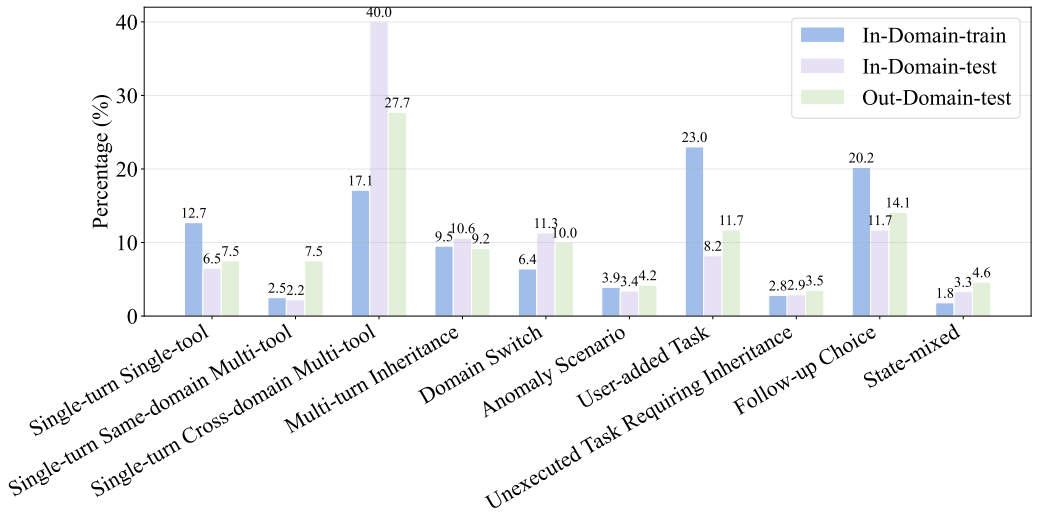


Figure 6: Dataset distribution (grouped percentage).

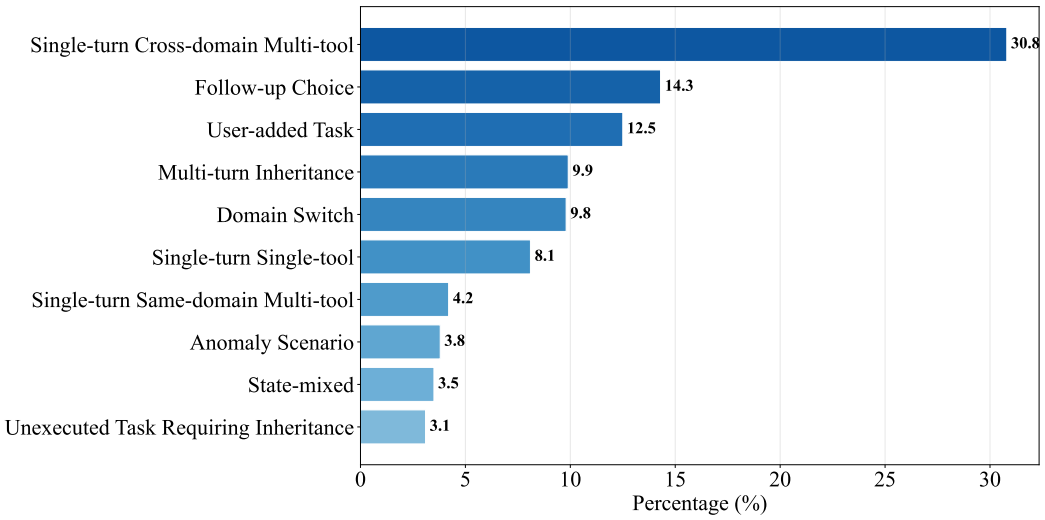


Figure 7: Overall dataset distribution.

G Further Analysis

G.1 When might PCTD fail?

Although PCTD improves performance through the joint reward mechanism, its effectiveness depends on the balance between the counterfactual reward R_{cf} and the preference reward R_{pref}. Table 9 reports the effect of the preference weight w_{pref} on retrieval performance and repetition rate, revealing the potential failure modes of PCTD under extreme reward configurations. When $w_{\text{pref}} = 0$, the model is optimized solely by the counterfactual reward. Although it achieves a high In-Domain N@10 of 90.65, its Out-of-Domain performance drops to 78.59, and the repetition rate increases to 5.24. This result suggests that counterfactual retrieval feedback alone is insufficient to guarantee high-quality task decomposition. Without structural preference constraints, the policy may still exploit retrieval metrics through redundant or repetitive decomposition patterns, leading to weaker OOD generalization.

Table 8: Definitions of dialogue scenario labels used in the dataset.

Scenario	Label	Brief Description
Single-turn	Single-turn Single Tool	A single-turn dialogue that completes a simple task using only one tool.
	Single-turn Multi-tool (Same Category)	A single-turn dialogue that uses multiple tools within the same category (e.g., multiple "travel" tools).
	Single-turn Cross-category Multi-tool	A single-turn dialogue that uses multiple tools across different categories (e.g., checking weather and booking a hotel).
Multi-turn	Domain Switching	In a multi-turn dialogue, the current turn switches to a completely new topic or domain unrelated to the previous turn.
	Exception Scenario	A multi-turn dialogue containing abnormal execution states such as failure, cancellation, re-execution, or rollback.
	Unexecuted Task Inheritance	A silent task from the previous turn is still running in the background and its state must be inherited in the current turn.
	Follow-up Selection	The assistant asks follow-up questions for missing details, or the user makes a selection during multi-turn dialogue.
	User-added Task	The user introduces a new task unrelated to the task from the previous turn during multi-turn dialogue.
	Mixed State Type	A multi-turn dialogue simultaneously including three or more of the above scenarios (excluding multi-turn inheritance).
	Multi-turn Inheritance	The current turn normally continues executing an unfinished task from the previous turn in multi-turn dialogue.

Table 9: Hyperparameter analysis of reward weights w_{pref}. Performance metrics (N@10, C@10) and repetition rates are reported as percentages (%).

w_{pref}	$1 - w_{pref}$	In Domain			Out Domain		
		N@10	C@10	Duplication Rate (%)	N@10	C@10	Duplication Rate (%)
0	1	90.65	86.18	5.24	78.59	75.79	1.60
0.2	**0.8**	**91.19**	**87.08**	0.71	**82.74**	**79.77**	1.35
0.5	0.5	90.50	86.63	0.70	81.85	79.50	1.00
0.8	0.2	86.55	80.59	0.36	77.74	73.24	0.10
1	0	89.49	85.70	0.41	78.93	75.77	0.17

As w_{pref} increases, the preference reward gradually provides stronger structural regularization. The repetition rate decreases substantially from 5.24 to 0.17, while the Out-of-Domain N@10 reaches its best value of 82.74 when $w_{\text{pref}} = 0.2$. However, when w_{pref} becomes too large, e.g., $w_{\text{pref}} = 1$, the model relies only on preference supervision. Although this setting effectively suppresses repetition, the In-Domain N@10 decreases to 89.49, indicating that overly strong structural regularization may weaken the model's ability to optimize retrieval-specific objectives.

These results show that the main failure risk of PCTD comes from an imbalance between the two reward signals. When $w_{\text{pref}} \to 0$, the lack of preference supervision makes the model prone to repetitive decomposition and OOD degradation. When $w_{\text{pref}} \to 1$, excessive preference supervision may over-regularize the policy and sacrifice retrieval accuracy. Therefore, PCTD performs best when the preference reward and the counterfactual reward are properly balanced, enabling the model to learn decompositions that are both structurally well-formed and effective for tool retrieval.

Table 10: Retrieval performance on the MTDTool benchmark using different retrievers.

	In-Domain		Out-of-Domain	
Method	N@15	C@15	N@15	C@15
Qwen3-Embedding-4B				
Base Retriever	68.35	61.79	50.57	42.22
Ours	91.19	87.08	82.74	79.77
Δ (points)	22.84	25.29	32.17	37.55
gte-Qwen2-1.5B-instruct				
Base Retriever	61.93	54.74	48.01	38.20
Ours	87.01	83.93	81.40	80.97
Δ	25.08	29.19	33.39	42.77
bge-large-zh-v1.5				
Base Retriever	52.06	39.58	40.98	29.69
Ours	81.67	76.01	79.89	78.10
Δ	29.61	36.43	38.91	48.41
Qwen3-Embedding-0.6B				
Base Retriever	64.42	58.26	48.43	39.84
Ours	88.54	86.60	84.27	82.43
Δ	24.12	28.34	35.84	42.59
e5-mistral-7b-instruct				
Base Retriever	55.73	48.09	40.58	34.07
Ours	84.10	80.44	71.45	65.87
Δ	28.37	32.35	30.87	31.80
Fine-tuned Qwen3-Embedding-4B				
Base Retriever	93.37	87.97	72.20	64.10
Ours	93.31	89.81	82.85	84.06
Δ	-0.06	1.84	10.65	19.96

G.2 Performance of PCTD Across Different Retrievers

Table 10 presents the zero-shot generalization performance of PCTD across different retrievers after training on MTDTool. This cross-retriever analysis uses $K = 15$, whereas the main experiments report results at $K = 10$. The experiments cover a range of mainstream embedding models from lightweight (0.6B) to large-scale (7B+), as well as a Qwen3-4B model fine-tuned on downstream tasks. Key analyses and conclusions are as follows:

(1) **Strong Retriever-Agnostic Capability:** PCTD achieves consistent improvements across the evaluated base retrievers. Notably, for the smaller-parameter models `Qwen3-Embedding-0.6B` and `bge-large-zh-v1.5`, In-Domain N@15 increases by 24.12 and 29.61 points, respectively, while Out-of-Domain C@15 improves by 42.59 and 48.41 points. These results indicate that PCTD's atomic subtasks provide useful retrieval inputs across retrievers of different scales, including lightweight models. (2) **More Pronounced Robustness Enhancement in OOD Scenarios:** Compared to In-Domain performance, the improvement margin of PCTD in OOD scenarios is generally 10–20 percentage points higher. For instance, on `gte-Qwen2-1.5B`, OOD C@15 improves by 42.77 points, exceeding the In-Domain gain of 29.19 points. This pattern is consistent with improved robustness to unseen tools and reduced dependence on training-set co-occurrences, but cross-retriever gains alone do not establish causal invariance. (3) **Complementary Advantage over Fine-Tuned Retrievers:** Even against the downstream fine-tuned `Fine-tuned Qwen3-Embedding-4B`, PCTD remains competitive. Although the absolute gain narrows on this specific fine-tuned model due to its high adaptation to the task distribution (In-Domain N@15 drops slightly by 0.06 points, but C@15 still rises by 1.84 points; OOD N@15 increases by 10.65 points), this precisely highlights that the core value of PCTD lies in **optimizing the semantic quality at the input end**. When the retriever is already sufficiently strong, PCTD shifts from "score boosting" to "tackling hard cases," continuing to unlock potential in OOD difficult examples; whereas for un-tuned generic retrievers, PCTD plays a decisive role in achieving substantial performance leaps.

Table 11: Definition of Tool Execution States and Typical Transition Logic. The transition logic follows the two-stage process $d \sim P_{\text{todo}}(s) \rightarrow s' \sim P_{\text{task}}(d)$.

Category	**ID**	**State Name**	**Core Semantics**	**Typical Transition Logic ($P_{\text{todo}} \rightarrow P_{\text{task}}$)**
Regular Execution	s_1	Success	Task completed	Null (0.95) / Revocation (0.05) → Remove / Reverse Op
	s_2	In Progress	Task ongoing	Retry (0.8) / Cancel (0.2) → Success / Failure
	s_3	Failure	Abnormal termination	Retry (0.9) / Cancel (0.1) → Retry / Terminate
Interaction Clarification	s_4	Clarification	Missing key info	Supplementation (1.0) → Re-execute after user input
	s_5	Selection	Multi-option confirm	Selection Confirm (1.0) → Execute corresponding branch
	s_6	Silence	Demand not triggered	Maintain silence / Transition to new task
Intent Management	s_7	Cancellation	Explicit refusal	Null (1.0) → Tool exits context
	s_8	Revocation	Rollback operation	Null (1.0) → Execute reverse tool
	s_9	Retry	Retry request	Success (0.95) / Failure (0.05)
	s_{10}	Supplementation	Active completion	Success (1.0) → Continue subsequent tasks
	s_{11}	Null	Exit dialogue	–

In summary, consistent gains across the evaluated retrievers indicate that PCTD is not tied to a single embedding model. Its atomic subtasks improve retrieval inputs for both lightweight and larger retrievers, particularly in the evaluated OOD settings.

G.3 Training Curves Analysis

We further analyze the training dynamics of various metrics, comparing the existing reinforcement learning baseline relying on traditional tool retrieval rewards with our PCTD framework, as illustrated in Figure 4. Specifically, as shown in Figure 4(a), the reward model score of PCTD rises more rapidly and reaches a final value of 0.62, significantly outperforming the baseline value of 0.59. This indicates that the preference reward R_{pref} effectively guides the model to generate high-quality decomposition results that are more logically coherent, highly atomic, and better aligned with automatically constructed references, thereby compensating for the lack of structural supervision in existing methods. In terms of policy exploration, as depicted in Figure 4(b), the policy entropy of PCTD rapidly decreases and stabilizes at a minimal level of 0.05, substantially lower than that of the baseline. This shows that the joint reward is associated with faster entropy reduction in this training run; entropy dynamics do not establish a causal gradient or convergence to a globally optimal decomposition strategy. More crucially, as demonstrated in Figure 4(c), the response length of PCTD is significantly shortened to approximately 17 tokens, whereas the baseline remains at 22 tokens. This shorter response length is consistent with reduced decomposition redundancy under structural supervision, although response length alone is not a direct measure of reward hacking.We therefore interpret the response-length result as evidence about decomposition conciseness rather than as a comprehensive comparison of training efficiency.

System: You are a real human phone user interacting with an AI voice assistant. Generate a natural spoken command.

User:

Below is the list of tools available to the assistant (for your reference only; you must trigger them implicitly):

```
available_tools:
[
{"tool_name": "create_reminder",
"description": "Create, query, delete and modify reminders",
"actions": ["create", "query", "delete"],
"objects": ["time", "title", "repeat rule"]},
{"tool_name": "weather_service",
"description": "Query weather for a specified city and date",
"actions": ["query weather"],
"objects": ["city", "date"]},
{"tool_name": "calendar_service",
"description": "Query, create, modify and delete calendar events",
"actions": ["query", "create", "delete"],
"objects": ["title", "start time", "end time", "location"]}
]
```

Conceive a natural human voice command, strictly following these rules:

(1) **Order**: express demands in the exact order of the tool list.

(2) **One-tool-one-request**: exactly 3 tools ⇒ exactly 3 demands.

(3) **Colloquial conversion**: do not copy official terminology verbatim.

(4) **No long sentences**: do not cram all demands into one sentence.

State guidance (injected when a tool is assigned the *Ask-Detail* state):

> Tool `weather_service` is in state **Ask-Detail**. You must **deliberately omit exactly one** critical information slot (e.g., city) in the corresponding demand. The omission must be achieved through natural colloquial phrasing, with no explanatory language.

Linguistic style (one feature randomly sampled per turn):

> Use *fuzzy quantifiers and relative time*—avoid exact numbers or absolute timestamps; use expressions like "a bit", "later", "in a couple of days" in exactly one demand.

Expected output: Set a reminder for me to pick up the kid this afternoon. Oh, by the way, what's the weather like tomorrow? And while you're at it, check if I've got any meetings on Wednesday.

First-turn user prompt for instruction generation. Variable blocks (tool list, state guidance, linguistic style) are dynamically injected based on the state-machine configuration.

System: You are an AI voice assistant. Generate a natural, helpful response based on the user's command and backend execution states.

User: "Set a reminder for me to pick up the kid this afternoon. Oh, by the way, what's the weather like tomorrow? And while you're at it, check if I've got any meetings on Wednesday."

Assistant:

Backend execution states:

```
- Tool: Create, query, delete and modify reminders
  State: [Success]
- Tool: Query weather for a specified city and date
  State: [Ask-Detail]
  Follow-up direction: city
- Tool: Query, create, modify and delete calendar events
  State: [Silent]
```

(2) **Coherent delivery**: blend multiple task feedback into one fluent response; no bullet-point reporting.

(3) **No tool leakage**: never reveal tool names, codes, or backend terminology.

Expected output: "Your reminder to pick up the kid this afternoon has been set. By the way, which city would you like me to check the weather for tomorrow?"

First-turn assistant prompt. Responses are strictly driven by execution states; the *Silent* tool is completely absent from the output.

System: You are a real human phone user. Generate the next-turn spoken command, building on the prior conversation.

Dialogue history:

> **User:** Set a reminder to pick up the kid this afternoon. Oh, by the way, what's the weather like tomorrow? And check if I've got any meetings on Wednesday."
> **Assistant:** Your afternoon reminder is set. Which city would you like me to check the weather for?"

User (next turn):

> *This turn requires:*
> - Answer the assistant's follow-up question about the missing city.
> - Introduce new demands corresponding to the tools below.
>
> *Tools and target states:*
>
> ```
> ["weather_service", "calendar_service", "wifi_service"]
> States: [Ask-Detail, Select, Silent]
> ```
>
> **Rules:**
> 1. **Supplement**: provide the previously omitted city name naturally; do not repeat the full original instruction.
> 2. **Selection**: choose one of the assistant's offered options; output only the option itself.
> 3. **Silent**: completely skip this tool—no mention, hint, or summary.
> 4. **Order**: demands must follow the tool list order.
> 5. **One-tool-one-request**: exactly 3 tools $\Rightarrow$ 3 demands.
> 6. **No confirmation words**: no "okay", "thanks", etc.
>
> *Linguistic style (one demand only):* Use *inverted sentence*—front-load the object before the action, e.g., "That QQ Music, open it for me."

Expected output: "Beijing. The first meeting on Wednesday, cancel that. And turn on the Wi-Fi."

Next-turn user prompt (context-inheritance mode). The user must simultaneously answer the assistant's prior follow-up, make a selection, introduce new demands, and silently skip one tool—all in a single utterance.

System: You are an AI voice assistant. Generate a natural response based on the user's next-turn command and backend execution states.

User: "Beijing. The first meeting on Wednesday, cancel that. And turn on the Wi-Fi."

Assistant:

Backend execution states:

```
Tool: Query weather for a specified city and date State:
     [Success]
Tool: Query, create, modify and delete calendar events
    State: [Ask-Detail] Follow-up direction: specific
    date to cancel
Tool: Toggle Wi-Fi connectivity State: [Executing]
```

Response rules:

1. **State-driven**:
   - *Success*: confirm completion.
   - *Ask-Detail*: ask a specific question for the missing information.
   - *Executing*: inform the user the task is being processed.
   - *Silent*: completely skip, as if the demand never existed.
2. **Coherent**: blend all feedback into one fluent response.
3. **No tool leakage**: never reveal tool names or backend terms.

Expected output: "I've checked the weather for Beijing—it looks clear tomorrow. Which specific meeting on Wednesday would you like me to cancel? Meanwhile, I'm turning on the Wi-Fi for you."

Next-turn assistant prompt. Identical rule structure to the first-turn version; only the injected user query and tool states differ.

System: You are a multi-turn rewriting and task decomposition expert. Decompose user commands into single-turn, single-task units.

Dialogue history:

> **User:** Set a reminder to pick up the kid this afternoon. Oh, by the way, what's the weather like tomorrow? And check if I've got any meetings on Wednesday."
> **Assistant:** Your afternoon reminder is set. Which city would you like me to check the weather for?"

Current query: "Beijing. The first meeting on Wednesday, cancel that. And turn on the Wi-Fi."

Task definition:

1. **Rewriting**: resolve pronouns and omissions using context. E.g., (Q: open Bluetooth)(A: done)(Q: turn it off)" $\rightarrow$ turn off Bluetooth".
2. **Multi-task splitting**: separate compound commands into atomic tasks. E.g., (Q: open Wi-Fi, Bluetooth and dark mode)" $\rightarrow$ open Wi-Fi$<$ br $>$open Bluetooth$<$ br $>$enable dark mode".

Output format: separate tasks with `<br>`.

Expected output:

> "Query weather for Beijing$<$ br $>$cancel the first Wednesday meeting$<$ br $>$turn on Wi-Fi"

Task decomposition prompt. Invoked independently at each turn to produce an atomic task sequence $\mathcal{A}_t$, forming the middle tier of the three-level annotation structure (context $\rightarrow$ atomic tasks $\rightarrow$ candidate tools).

Round 1 (Initial) — Tools: `create_reminder`, `weather_service`, `calendar_service`
States: [Success, Ask-Detail, Silent]

User: "Set a reminder to pick up the kid this afternoon. Oh, by the way, what's the weather like tomorrow? And check if I've got any meetings on Wednesday."

Task split:
"Set reminder to pick up kid this afternoon"
"Query tomorrow's weather"
"Check Wednesday meetings"

Assistant: "Your afternoon reminder is set. Which city would you like me to check the weather for tomorrow?" *(note: calendar_service → Silent, completely absent)*

Round 2 (Inherit) — Tools: `weather_service`, `calendar_service`, `wifi_service`
States: [Ask-Detail, Select, Executing]

User: "Beijing. The first meeting on Wednesday, cancel that. And turn on the Wi-Fi."

Task split:
"Query weather for Beijing"
"Cancel first Wednesday meeting"
"Turn on Wi-Fi"

Assistant: "Got it—tomorrow in Beijing will be partly cloudy, around 18 degrees. Your Wednesday meeting has been cancelled. Meanwhile, I'm enabling Wi-Fi for you."

Round 3 (Inherit) — Tools: `navigation_service`, `calendar_service`, `create_reminder`
States: [Success, Retry, Ask-Detail]

User: "Navigate to the company. That meeting thing, try again. Also remind me to bring my laptop—oh wait, make it two reminders, one for the laptop and one for the charger."

Task split:
"Navigate to company"
"Retry creating Wednesday meeting"
"Remind to bring laptop"
"Remind to bring charger"

Assistant: "Route to the company is ready, about 25 minutes. I'm retrying the meeting creation now. What time should I set the laptop reminder for?"

Round 4 (Switch) — Tools: `music_service`, `bluetooth_service`
States: [Success, Success]

User: "Play some light music. Oh, and connect the headphones."

Task split:
"Play light music"
"Connect Bluetooth headphones"

Assistant: "Playing a light music playlist. Your Bluetooth headphones are now connected."

End-to-end multi-turn dialogue construction example. Round 1 initializes states; Rounds 2–3 inherit context (answering follow-ups, retrying failed tasks, introducing new tools); Round 4 switches to a new topic. Each round independently invokes user prompt, task decomposition, and assistant prompt.