

Skill-CDPO: Evolving Agent Tool-Use via Critical Step Preference Optimization

Yuchen Li^{1*}, Jinghao Lin^{2*†}, Yuanqing Wang³

¹Tsinghua University ²Northeastern University, China

³The University of Texas at Dallas

linjinghao2k@gmail.com

Abstract

Compact open-source language models lag behind their larger counterparts in agentic tool-use reliability, yet standard remedies face fundamental obstacles: supervised fine-tuning suffers from exposure bias, while reinforcement learning is hampered by sparse credit assignment over long tool-interaction trajectories. We introduce **Skill-CDPO**, a progressive framework that first acquires tool-use skills at inference time through static tool analysis and dynamic strategy refinement, then distills the resulting error-correction signals into parameter updates via **Critical Step DPO (CDPO)**. CDPO identifies the specific trajectory steps where model capability is the bottleneck—through rollout divergence between a local policy and an expert model—and constructs distributional preference pairs from all cross-group rollouts at those steps, weighted by both step-level criticality and pair-level score gaps. This provides dense, fine-grained supervision without requiring a process reward model. We evaluate Skill-CDPO on three medical agent benchmarks—PubMed Search (a new PubMed-based deep research benchmark we contribute), CURE-Bench, and MedBrowseComp—using an 8B-parameter deep research model. Skill-CDPO substantially outperforms SFT and trajectory-level DPO baselines and achieves competitive or superior performance compared to GPT-5.2 on retrieval-intensive tasks. Our code and data are available at <https://github.com/Adam135792468/CDPO>.¹

1 Introduction

Large Language Models (LLMs) are undergoing a paradigm shift from passive chatbots to autonomous agents capable of interacting with exter-

nal environments to solve complex problems (Yao et al., 2023; Schick et al., 2023; Qin et al., 2024). Recent Agent Skills systems package procedural knowledge as reusable instructions, scripts, and resources that agents can load on demand (Anthropic, 2025). While large-scale models (e.g., >70B parameters) have demonstrated remarkable proficiency, their compact, deployable counterparts continue to struggle with reliability, particularly in specialized domains such as medical research where precision is paramount. Closing this gap through standard training approaches faces fundamental obstacles. SFT is trained on demonstration histories, whereas at inference time the policy conditions on its own previous outputs, creating a train–inference distribution mismatch under which early errors may compound (Ranzato et al., 2016). Moreover, demonstration-only SFT does not explicitly contrast preferred and rejected actions, motivating preference-based optimization (Rafailov et al., 2023). Reinforcement Learning (RL) methods (Schulman et al., 2017; Shao et al., 2024) are hampered by the *credit assignment problem* (Minsky, 1961): in a long chain of tool interactions, a final failure reward provides sparse supervision, making it difficult to identify which specific step caused the error. Neither approach can effectively enable compact models to match their larger counterparts.

In this work, we present two complementary findings that together address this challenge. First, we observe that compact models can **substantially improve tool-use reliability through inference-time skill synthesis**. By analyzing tool implementations to build accurate mental models of tool capabilities (*static optimization*) and by learning refined usage strategies from runtime execution logs (*dynamic optimization*), an 8B-parameter model

¹*Equal contribution. [†]Corresponding author.

achieves significant performance gains without any parameter updates. This training-free skill acquisition alone closes a notable portion of the gap, yet deeper deficiencies in the model’s decision-making persist and cannot be resolved without parameter updates.

Furthermore, we discover that a lightweight preference optimization approach can **further unlock the potential of newly acquired skills, enabling the compact model to surpass even larger closed-source systems**. We propose **Critical Step DPO (CDPO)**, which identifies the specific trajectory steps where model capability remains the bottleneck—through rollout divergence between the local policy and an expert model—and constructs distributional preference pairs from all $n \times m$ cross-group rollouts at those steps, weighted by both step-level criticality and pair-level score gaps. This provides dense, fine-grained supervision without requiring a process reward model, improving step-level credit assignment relative to trajectory-level methods.

Together, these two insights form **Skill-CDPO**, a progressive framework that first acquires tool-use skills at inference time, then distills the resulting error-correction signals into parameter updates via CDPO. We evaluate Skill-CDPO on three medical agent benchmarks—**PubMed Search** (a new PubMed-based deep research benchmark we contribute), **CURE-Bench** ([CURE-Bench Organizers, 2025](#)), and **MedBrowseComp** ([Chen et al., 2025b](#))—using an 8B-parameter deep research model. Our results reveal a clear progression: skill acquisition provides a strong foundation, but standard training methods (SFT, trajectory-level DPO) yield limited additional gains on top of it; CDPO, by contrast, fully unleashes the potential of skill-augmented agents, substantially outperforming all training baselines and achieving competitive or superior performance compared to GPT-5.2 on retrieval-intensive tasks.

Our contributions are: (1) **Skill-CDPO**, a progressive framework that bridges inference-time skill acquisition with critical step preference optimization for tool-use agents; (2) **Critical Step DPO (CDPO)**, which improves step-level credit assignment through distributional preference pairs at rollout-divergence-verified bottleneck steps, offering an efficient alternative to RL; and (3) **PubMed Search**, a new PubMed-based deep research benchmark, on which our 8B model achieves state-of-the-art open-source performance across all three

medical agent tasks.

2 Related Work

LLM Agents and Tool-Use Training. ReAct ([Yao et al., 2023](#)) and Toolformer ([Schick et al., 2023](#)) establish foundational paradigms for interleaving reasoning with tool invocations. On the training side, ETO ([Song et al., 2024](#)) constructs trajectory-level preference pairs from exploration rollouts, while Agent Q ([Putta et al., 2024](#)) integrates tree search with offline preference learning. In biomedical applications, retrieval-intensive agents must chain multiple search, reading, and synthesis steps ([Chen et al., 2025b](#)), where a single poorly formulated query can derail downstream reasoning. Our work targets this setting with a two-stage pipeline tailored to the compounding-error dynamics of retrieval-intensive agent tasks.

Preference Optimization at Different Granularities. DPO ([Rafailov et al., 2023](#)) casts preference alignment as a binary classification problem, spawning variants that modify the loss geometry ([Azar et al., 2024](#)), relax paired data requirements ([Ethayarajh et al., 2024](#)), or drop the reference model ([Meng et al., 2024](#))—all operating at the response level with uniform token weighting. Token-level DPO ([Zeng et al., 2024](#)) reweights tokens by marginal contribution, and Step-DPO ([Lai et al., 2024](#)) segments proofs into discrete steps. For agent trajectories, CSO ([Li et al., 2026](#)) trains a process reward model to score candidate steps, while IPR ([Xiong et al., 2024](#)) estimates per-step value through Monte Carlo rollouts at every position—yielding unbiased signals but scaling poorly with trajectory length. CDPO departs from both in three ways: it requires no process reward model, it leverages the full nm rollout cross-product rather than isolated pairs, and it weights steps by a normalized criticality score.

3 Method

In this section, we present **Skill-CDPO**, a progressive framework for evolving agent tool-use skills. As illustrated in Figure 1, our approach consists of two stages: (1) a *training-free* skill acquisition phase (§3.1), where the agent refines its tool-use behavior through static and dynamic optimization without parameter updates; and (2) a *training-based* preference optimization phase (§3.2), where critical step preference pairs are constructed from

error correction logs and used for direct preference optimization.

3.1 Training-Free Skill Acquisition

Before any parameter optimization, the agent first acquires tool-use skills through two complementary mechanisms that operate at inference time: *static optimization*, which deepens the agent’s understanding of tool specifications, and *dynamic optimization*, which refines tool-use strategies based on runtime feedback.

Static Optimization. When invoking external tools via the Model Context Protocol (MCP) (Anthropic, 2024), models are typically provided with tool descriptions that specify function signatures and usage guidelines. However, such descriptions are often overly brief and may omit edge-case behaviors, undocumented parameters, or input formatting constraints. To address this, we enable the agent to construct tool-specific skills by examining the concrete implementation of each MCP tool, building a more accurate mental model of tool capabilities, input requirements, and expected outputs.

Dynamic Optimization. Static analysis alone cannot capture tool-specific usage strategies—for instance, Google benefits from verbose queries while PubMed requires concise, keyword-focused terms. Through learning from historical invocation logs, the model acquires fine-grained strategies for each tool and identifies recurring failure patterns, enabling programmatic guardrails that proactively intercept known error modes (e.g., deduplication checks to block redundant API calls). The training-free phase produces refined tool-use behavior and, critically, generates the error correction logs that serve as the foundation for the subsequent training-based optimization stage. Implementation details and examples are provided in Appendix A.1.

3.2 Critical Step Direct Preference Optimization

While the training-free phase effectively addresses surface-level issues such as malformed tool calls and incorrect API parameters, deeper deficiencies—such as suboptimal query formulation, inefficient search-retrieval loops, or poor information synthesis—persist. These deficiencies are rooted in the model’s decision-making at specific trajectory steps, and cannot be resolved without parameter updates.

A natural approach is trajectory-level DPO (Rafailov et al., 2023), which treats entire trajectories as preference pairs. However, trajectory-level methods distribute a single binary outcome reward uniformly across dozens of steps, most of which the model already handles correctly. The result is noisy, slow training that wastes both data and compute. We instead propose **Critical Step DPO (CDPO)**, an offline preference optimization approach that identifies the specific steps where model capability is the bottleneck and provides fine-grained, distributional supervision at those steps, without the high cost and instability of online RL methods.

3.2.1 Notation and Problem Setup

We consider an agent operating over a task with query q . Following the ReAct paradigm (Yao et al., 2023), the agent produces a trajectory:

$$\tau = (s_1, a_1, o_1, s_2, a_2, o_2, \dots, s_T, a_T, o_T) \quad (1)$$

where $s_t = (q, a_1, o_1, \dots, a_{t-1}, o_{t-1})$ is the full history up to step t , $a_t \sim \pi_\theta(\cdot \mid s_t)$ is the policy action, and o_t is the environment observation. The trajectory terminates with an outcome score $r \in [0, A]$ assigned by a task-specific evaluation rubric (with A being the maximum score).

We maintain two models throughout:

- π_θ : the *local policy* being trained (e.g., an 8B open-source model);
- π_{exp} : a fixed *expert model* used only for generating positive alternatives (e.g., a frontier closed-source model).

The central insight of CDPO is that a step t is *critical* if and only if model capability is the bottleneck at that point: the expert model succeeds from state s_t where the local model fails. Steps that are easy for both models, or hard for both, carry no discriminative learning signal and should be excluded from training.

3.2.2 Critical Step Identification

We adopt a two-stage pipeline to identify critical steps: a cheap candidate detection stage followed by a rigorous rollout-based verification stage.

Stage 1: Candidate Detection. To avoid the prohibitive cost of running rollouts at every step of every trajectory, we first apply lightweight filters to identify candidate critical steps. Identifying critical

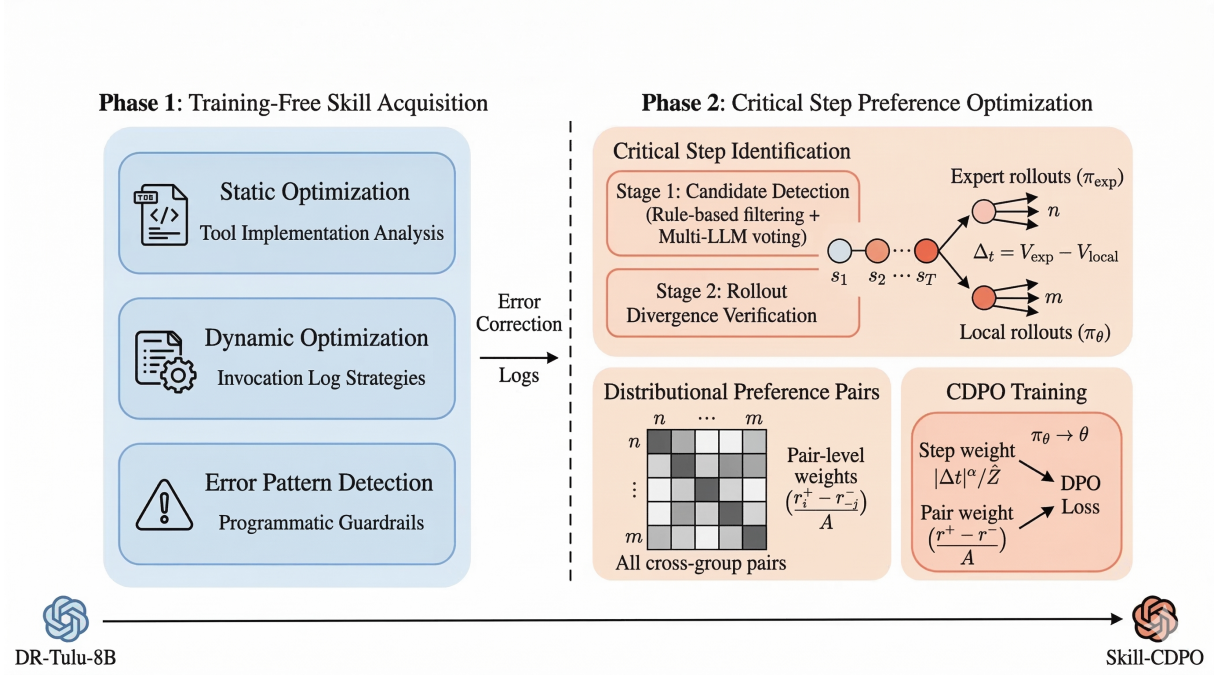


Figure 1: Overview of the Skill-CDPO framework. **Phase 1** (left): Training-free skill acquisition refines tool-use behavior through static tool analysis, dynamic strategy learning from invocation logs, and automated error pattern detection. **Phase 2** (right): Critical Step DPO identifies bottleneck steps via rollout divergence between expert and local policies, constructs distributional preference pairs from all $n \times m$ cross-group rollouts, and trains the local model with criticality-weighted, pair-weighted DPO.

steps typically relies on a Process Reward Model (PRM) to judge step quality (Lightman et al., 2024). However, training a dedicated PRM for agentic tool-use trajectories is costly, and directly using an LLM as a PRM judge suffers from instability, as current LLMs have limited exposure to multi-step tool-use evaluation during training. We instead combine two complementary signals:

1. **Rule-based filtering.** Observable features such as tool call failures, empty search results, repeated queries, and timeout errors serve as high-recall indicators of potential critical steps. For multiple-choice tasks with definitive answers, we additionally apply term-frequency analysis over rollout outputs to pinpoint the step at which an incorrect answer is first introduced or where the frequency of incorrect answers begins to exceed that of the correct one. The detailed algorithm is provided in Appendix B.
2. **Multi-LLM voting.** Candidate steps from the rule-based filter are further assessed by multiple LLMs that independently evaluate whether the step represents a genuine capability bottleneck. This ensemble-based approach provides

precision without requiring a dedicated PRM.

Stage 1 only needs to function as a rough filter: false negatives are acceptable since they merely reduce the training set, while false positives are caught by the subsequent verification stage.

Stage 2: Rollout Divergence Verification. At each candidate step t identified by Stage 1, we verify its criticality through *rollout divergence*—a direct, outcome-based measure that requires no learned reward model. From state s_t , we generate n expert rollouts and m local rollouts, each continuing to task completion:

$$\{a_{t,1}^+, \dots, a_{t,n}^+\} \sim \pi_{\text{exp}}(\cdot | s_t) \quad (2)$$

$$\{a_{t,1}^-, \dots, a_{t,m}^-\} \sim \pi_{\theta}(\cdot | s_t) \quad (3)$$

Each rollout yields an outcome score $r_{t,i}^+$ or $r_{t,j}^-$ from the task evaluation rubric. We define the empirical value estimates as:

$$\hat{V}_{\text{exp}}(s_t) = \frac{1}{n} \sum_{i=1}^n r_{t,i}^+, \quad \hat{V}_{\text{local}}(s_t) = \frac{1}{m} \sum_{j=1}^m r_{t,j}^- \quad (4)$$

These are unbiased Monte Carlo estimates of the true state values $V_{\pi_{\text{exp}}}(s_t)$ and $V_{\pi_{\theta}}(s_t)$.

Definition 1 (Criticality Score). *The criticality score of step t is the empirical advantage of the expert policy over the local policy from state s_t :*

$$\Delta_t = \hat{V}_{\text{exp}}(s_t) - \hat{V}_{\text{local}}(s_t) \in [-A, +A] \quad (5)$$

This score has a natural interpretation: $\Delta_t \approx 0$ indicates both models handle this step similarly (not critical); $\Delta_t \gg 0$ indicates the expert consistently outperforms the local model (highly critical); $\Delta_t < 0$ indicates the local model happens to perform better (rare, ignored).

Definition 2 (Verified Critical Step). *Step t is a verified critical step if the rubric-normalized gap Δ_t/A exceeds a dimensionless threshold $\epsilon > 0$. The set of critical steps in trajectory τ is:*

$$\mathcal{C}_\tau = \{t \in [T] : \Delta_t/A > \epsilon\} \quad (6)$$

Crucially, Δ_t is computed entirely from observed task outcomes with no reliance on a process reward model. This eliminates PRM approximation error as a noise source and ensures that criticality estimates converge to the true advantage gap as the number of rollouts increases.

Dual-Context Rollout Construction. Following a *proactive prevention and post-hoc remedy* paradigm, we construct two rollout contexts for each critical step: (1) a context truncated before the critical tool call, allowing the model to learn to avoid the error proactively by choosing a better action; and (2) a context that includes the critical tool call and its observation, allowing the model to learn corrective alternatives after encountering the error. Both the local and expert models generate rollouts from these contexts, yielding preference pairs that capture complementary aspects of the error—prevention and recovery.

3.2.3 From Standard DPO to Distributional CDPO

Standard DPO (Rafailov et al., 2023) operates on a single preference pair (s, a^+, a^-) and maximizes:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma \left(\beta \log \frac{\pi_\theta(a^+ | s)}{\pi_{\text{ref}}(a^+ | s)} - \beta \log \frac{\pi_\theta(a^- | s)}{\pi_{\text{ref}}(a^- | s)} \right) \quad (7)$$

This uses exactly one bit of preference information per step. However, at each critical step our rollout procedure yields $n \times m$ cross-group preference observations. We leverage all of these observations through a distributional formulation.

All-Pairs Preference Aggregation. At each verified critical step t , the n expert and m local rollouts provide $n \times m$ pairwise comparisons. Rather than selecting a single “best” pair, we aggregate over all pairs to form a distributional training signal.

For rubric-scored tasks, each pair (i, j) carries a score gap $d_{t,ij} = r_{t,i}^+ - r_{t,j}^-$. We orient the comparison so that the higher-scoring action is preferred, and use the absolute gap as a dimensionless pair-level weight:

$$q_{t,ij} = \frac{|d_{t,ij}|}{A} \in [0, 1] \quad (8)$$

Specifically, let $(a_{t,ij}^*, a_{t,ij}^o) = (a_{t,i}^+, a_{t,j}^-)$ when $d_{t,ij} \geq 0$, and swap the order otherwise. Ties have $q_{t,ij} = 0$ and therefore contribute no gradient. This retains the fixed all-pairs average while ensuring a valid preference direction (Appendix E.4).

Criticality-Weighted Step Importance. Not all critical steps are equally informative. A step where the expert scores 30 points higher on average than the local model should receive more gradient update than one with only a 5-point gap. We encode this through a *criticality weight* $w(\Delta_t)$.

The most obvious choice, using Δ_t directly as the weight, fails for two reasons. First, Δ_t is measured in rubric points and is therefore not dimensionless. Second, without knowing what a typical divergence looks like across the selected steps, the raw value provides no relative scale.

We resolve both issues through a normalization constant $\hat{Z}$, computed once before training over the selected critical steps $\mathcal{D}_{\text{crit}}$:

$$\hat{Z} = \frac{1}{|\mathcal{D}_{\text{crit}}|} \sum_{t \in \mathcal{D}_{\text{crit}}} \Delta_t^\alpha \quad (9)$$

The normalized step weight is then:

$$w(\Delta_t) = \frac{\Delta_t^\alpha}{\hat{Z}} \quad (10)$$

By construction, the weights average to 1 over $\mathcal{D}_{\text{crit}}$. Since standard DPO implicitly assigns weight 1 to every selected step, CDPO preserves the average loss scale while redistributing weight toward larger verified gaps. The normalization also provides *scale invariance*—changing the rubric scale leaves $w(\Delta_t)$ unchanged (Appendix E.2)—and the exponent α controls the concentration of weight ($\alpha = 1$ is our default; Appendix E.3).

The Complete CDPO Loss. Combining the step-level criticality weight, pair-level score gap, and standard DPO log-likelihood, we arrive at the full CDPO objective:

$$\mathcal{L}_{\text{CDPO}}(\theta) = -\mathbb{E}_{\{t: \Delta_t/A > \epsilon\}} \left[\underbrace{\frac{\Delta_t^\alpha}{\hat{Z}}}_{\text{step weight}} \cdot \frac{1}{nm} \sum_{i=1}^n \sum_{j=1}^m \underbrace{q_{t,ij}}_{\text{pair weight}} \cdot \log \sigma \left(\beta \log \frac{\pi_\theta(a_{t,ij}^* | s_t)}{\pi_{\text{ref}}(a_{t,ij}^* | s_t)} - \beta \log \frac{\pi_\theta(a_{t,ij}^\circ | s_t)}{\pi_{\text{ref}}(a_{t,ij}^\circ | s_t)} \right) \right] \quad (11)$$

The loss combines a mean-normalized step weight, a non-negative score-gap weight, and the standard DPO log-likelihood over a score-oriented pair. Thus all nm comparisons remain in the fixed average, ties contribute zero, and higher-scoring actions define the preference direction.

Under the Bradley–Terry preference model, the inner log-sigmoid term models the probability that the higher-scoring action is preferred. The pair average uses the full cross-group comparison matrix rather than selecting a single pair.

3.2.4 Theoretical Properties

We state two statistical properties of CDPO. Proofs are provided in Appendix C.2.

Proposition 1 (Two-Sample Variance Decomposition). *Let $h(X_i, Y_j)$ be a square-integrable per-pair gradient built from independent expert rollouts X_i and local rollouts Y_j , and let $\hat{g}_{n,m} = \frac{1}{nm} \sum_{i,j} h(X_i, Y_j)$. Then $\text{Var}(\hat{g}_{n,m}) = \sigma_X^2/n + \sigma_Y^2/m + \sigma_R^2/(nm)$ for non-negative Hoeffding-decomposition components. Thus more rollouts reduce variance, but shared-rollout pairs are not generally independent.*

Proposition 2 (Unbiased Criticality Estimation). *The empirical criticality score $\hat{\Delta}_t$ is an unbiased estimator of the true advantage gap $\Delta_t^* = V_{\pi_{\text{exp}}}(s_t) - V_{\pi_\theta}(s_t)$, with standard error $\text{SE}(\hat{\Delta}_t) = \sqrt{\text{Var}[r_t^+]/n + \text{Var}[r_t^-]/m}$.*

Together, these results justify averaging cross-group comparisons and establish unbiased criticality estimation without treating the nm comparisons as independent or claiming an nm -fold sample-efficiency guarantee.

4 Experiments

4.1 Experimental Setup

Policy Model and Tools. We adopt DR-Tulu-8B (Shao et al., 2025) as the local policy model π_θ , an open-source deep research model built on Qwen3-8B (Yang et al., 2025) that has been trained for multi-step information retrieval and long-form answer synthesis. The model interacts with external tools through the MCP-based agent infrastructure (Anthropic, 2024) provided by DR-Tulu, which includes a suite of search and web-reading tools (e.g., web search, PubMed API search, page reading, and content extraction) implemented via FastMCP (Lowin, 2024). We use the default dr-agent tool configuration throughout the primary experiments and GPT-5.2 (OpenAI, 2025b) as the fixed expert. Additional model-scale and general-domain results are reported in Appendix A.2.

Benchmarks. We evaluate on three medical benchmarks: the Phase 1 validation split of **CURE-Bench** (CURE-Bench Organizers, 2025), evaluated by exact string matching with LLM-judge fallback; **MedBrowseComp** (Chen et al., 2025b), a closed-form QA benchmark with the same evaluation protocol; and **PubMed Search**, a new PubMed-based deep research benchmark we introduce for biomedical literature retrieval and open-ended QA, evaluated via rubric-based scoring on factual accuracy, completeness, and attribution. All benchmarks use GPT-4.1-mini (OpenAI, 2025a) as the judge model. Split composition is reported in Appendix A.5.

Training Data. We construct task-specific data synthesis pipelines for each benchmark, generating training trajectories that capture the diversity of tool-use patterns and failure modes encountered in medical research tasks. The resulting dataset contains 16,951 trajectory samples: 5,672 from CURE-Bench, 3,124 from MedBrowseComp, and 8,155 from PubMed Search. Critical step identification and preference pair construction follow the pipeline described in §3.2.2, with $n = m = 4$ rollouts per group and criticality threshold $\epsilon = 0.3$. Dataset-quality and decontamination details are provided in Appendix A.5.

Training Details. For DPO training, we set the KL penalty coefficient $\beta = 0.5$ and use the $\alpha = 1$ (linear) weighting scheme as default. We train for 3 epochs with a learning rate of 5×10^{-6} and batch

Table 1: Main results on three medical benchmarks. PubMed Search reports rubric-based scores; CURE-Bench and MedBrowseComp report accuracy (%). Best results are **bolded**; second best are underlined. [†]Closed-source model with equivalent tool access.

Method	PubMed Search $\uparrow$	Cure Bench $\uparrow$	MedBrowse Comp $\uparrow$
GPT-5.2 [†]	<u>19.08</u>	71.05	27.93
DR-Tulu-8B	15.48	59.30	25.12
+ Skill	17.75	62.41	26.61
+ SFT	17.93	63.42	27.44
+ Traj. DPO	18.47	63.05	27.19
+ Step DPO	18.83	64.25	<u>28.06</u>
Skill-CDPO	19.58	<u>66.63</u>	29.09

size of 32. The reference model π_{ref} is initialized as the base DR-Tulu-8B checkpoint. All experiments are conducted on 8 NVIDIA A100 80GB GPUs.

Baselines. We compare against: **DR-Tulu-8B** (base model), **+ Skill** (training-free skill acquisition only), **+ SFT** (behavioral cloning on expert trajectories), **+ Trajectory DPO** (trajectory-level preference pairs), and **+ Step DPO** (step-level DPO with uniform weighting and single pairs). All training-based methods include the Skill phase for fair comparison. We additionally report GPT-5.2 with equivalent tool access as a closed-source reference.

4.2 Main Results

Table 1 presents the main results. We highlight the following findings.

Skill synthesis is necessary and provides an essential foundation. The training-free Skill phase alone yields substantial gains over the base model (+2.27 on PubMed Search, +3.11% on CURE-Bench, +1.49% on MedBrowseComp) without any parameter updates, confirming that inference-time skill synthesis provides a necessary foundation for subsequent optimization.

Standard training methods cannot fully exploit skill-augmented agents. On top of the Skill phase, SFT provides only marginal additional improvement (+0.18 on PubMed Search, +1.01% on CURE-Bench, +0.83% on MedBrowseComp), confirming that behavioral cloning fails to teach the model *when and how* to avoid errors. Trajectory DPO shows similarly limited gains, as it distributes outcome rewards uniformly across all steps rather than targeting specific capability bottlenecks. Even

Table 2: Ablation study. Each row removes one component from the full Skill-CDPO model.

Variant	PubMed	Cure	MedBrowse
Skill-CDPO (Full)	19.58	66.63	29.09
– Training-free Skill	18.36	64.71	27.68
– Critical step select.	18.41	64.15	27.53
– Criticality weight	19.14	65.89	28.51
– Pair-level weight	18.71	66.25	28.82
– All-pairs (single pair)	19.07	65.62	28.38

Step DPO, which operates at the step level with uniform weighting and single pairs, offers only modest improvements over Trajectory DPO.

CDPO fully unlocks the potential of skill-augmented agents. Skill-CDPO achieves substantial improvements over all training baselines (+0.75 over Step DPO on PubMed Search, +2.38% on CURE-Bench, +1.03% on MedBrowseComp), confirming that criticality-weighted distributional supervision captures learning signals that other methods miss. Crucially, Skill-CDPO outperforms GPT-5.2 on PubMed Search (+0.50) and MedBrowseComp (+1.16%), demonstrating that targeted critical step optimization enables a compact 8B-parameter model to surpass a larger closed-source system on retrieval-intensive tasks. On CURE-Bench, GPT-5.2 retains a lead (71.05% vs. 66.63%), likely due to broader parametric knowledge, but Skill-CDPO closes the gap from 11.75% to 4.42%.

4.3 Analysis

Ablation Study. To understand the contribution of each component, we remove one at a time from the full model. Results in Table 2 yield the following observations.

(1) **Critical step selection is the most impactful component.** Removing the criticality threshold and training on all steps produces the largest average degradation (−1.17 on PubMed Search, −2.48% on CURE-Bench, −1.56% on MedBrowseComp), confirming that including non-critical steps dilutes the gradient with noise.

(2) **Training-free skill acquisition provides an essential foundation.** Removing the Skill phase leads to substantial drops, indicating that it not only improves inference-time behavior but also produces higher-quality trajectories for preference pair construction.

(3) **Pair-level score weighting disproportionately affects rubric-scored tasks.** Reverting to

Table 3: Sensitivity to the weighting exponent α . Default setting ($\alpha = 1.0$) shown in **bold**.

	PubMed	Cure	MedBrowse
$\alpha = 0.5$	19.32	65.91	28.87
$\alpha = 1.0$	19.58	66.63	29.09
$\alpha = 2.0$	19.41	65.38	28.64

binary preference pairs causes the largest drop on PubMed Search (-0.87), where continuous score gaps are most informative, while the effect is minimal on binary-outcome CURE-Bench (-0.38%).

(4) **The all-pairs formulation provides consistent empirical gains.** Using a single randomly sampled pair instead of all nm pairs degrades performance across all benchmarks (-0.51 , -1.01% , -0.71%), consistent with the variance characterization in Proposition 1.

(5) **Criticality weighting offers a moderate but consistent benefit**, indicating that the degree of criticality carries useful signal beyond the binary selection decision.

Critical Step Statistics. Only 14.3% of steps are verified critical ($\Delta_t/A > \epsilon$), indicating that the selected bottlenecks are sparse. For the multiple-choice subset of the CURE-Bench training set—which provides both an option set and a definitive answer—we further classify wrong-answer errors via the algorithm in Appendix B: **misled by evidence** (39.3%, correct reasoning derailed by a tool failure), **final reasoning error** (36.9%, correct reasoning but wrong extracted letter; 42% of these have the correct answer stated verbatim in the reasoning text), **evidence ignored** (13.1%, correct evidence retrieved but dismissed), and **insufficient search** (10.7%, correct evidence never retrieved). The dominant failure mode is *evidence integration under uncertainty*—abandoning correct conclusions after tool failures or failing to act on retrieved evidence—accounting for over half of all errors and explaining CDPO’s largest gains at these steps. Representative cases are in Appendix I; threshold sensitivity is reported in Appendix A.3.

Sensitivity to α . Table 3 reports the sensitivity of CDPO to the weighting exponent α . The linear weighting ($\alpha = 1$) performs best on average across all three benchmarks. The concave choice $\alpha = 0.5$ is competitive, particularly on MedBrowseComp where noisier rollout estimates benefit from not over-emphasizing steps with large but potentially noisy $\hat{\Delta}_t$ values. The convex choice $\alpha = 2$ con-

centrates too aggressively on a few extreme steps, leading to slight underfitting on the remaining critical steps.

5 Conclusion

We presented Skill-CDPO, a progressive framework for evolving the tool-use capabilities of compact language models. Our experiments reveal two key findings. First, inference-time skill synthesis—through static tool analysis and dynamic strategy refinement—provides a necessary foundation that substantially narrows the gap between compact and large-scale models without any parameter updates. Second, and more importantly, Critical Step DPO (CDPO) further unlocks the potential of these newly acquired skills by concentrating preference learning on the specific trajectory steps where model capability is the bottleneck. Through rollout-divergence-based step identification, all-pairs distributional supervision, and criticality-weighted training, CDPO improves step-level credit assignment without a process reward model or online RL. On three medical agent benchmarks, an 8B-parameter Skill-CDPO model not only substantially outperforms SFT and trajectory-level DPO baselines, but also surpasses GPT-5.2 on retrieval-intensive tasks—demonstrating that targeted critical step optimization can compensate for orders-of-magnitude fewer parameters. We additionally contribute PubMed Search, a new PubMed-based deep research benchmark for biomedical literature retrieval.

Limitations

Dependence on an expert model. CDPO requires access to a frontier closed-source model π_{exp} for generating positive rollouts at critical steps. This introduces both a cost overhead and a capability ceiling: the local model can only learn to match behaviors that the expert can demonstrate. In settings where no substantially stronger model is available, the rollout divergence signal may be weak or absent.

Domain and model coverage. Our primary experiments use medical benchmarks and a single 8B-parameter base model (DR-Tulu-8B). Additional results on Qwen3-4B-Thinking and xbench-DeepSearch provide preliminary evidence beyond this setting, but broader validation remains necessary (Appendix A.2).

Data synthesis cost. Although CDPO achieves higher data utilization efficiency than trajectory-level DPO by concentrating learning on critical steps, the upstream cost of synthesizing the preference data is substantial: each candidate critical step requires $n + m$ complete trajectory rollouts from both the expert and local models. These costs are incurred once offline before training, but may be prohibitive when the tool-interaction environment is slow or expensive to query. Detailed measurements are provided in Appendix A.4.

Single-round training. Although we discuss iterative refinement in Appendix F, all reported results are from a single round of CDPO training. The benefits and potential instabilities of multi-round iterative CDPO remain empirically unvalidated.

Ethics Statement

All benchmarks used in this work are constructed from publicly available biomedical literature and do not involve private patient data or protected health information. PubMed Search, the benchmark we introduce, consists of open-ended research questions paired with information retrievable through the public PubMed API; no human subjects are involved in its construction or evaluation. Our framework is designed to improve tool-use reliability of language model agents and does not generate medical advice intended for clinical decision-making. We caution that outputs from any agent system—including those trained with Skill-CDPO—should not be used as a substitute for professional medical judgment. The expert model used for rollout generation is accessed through standard commercial APIs in compliance with the provider’s terms of service.

Acknowledgments

We thank the anonymous reviewers and area chairs for their constructive feedback.

AI Assistance Statement

Generative AI tools were used for language polishing and code assistance during manuscript preparation. The authors reviewed and edited all generated content and remain fully responsible for the paper’s claims, experiments, and presentation.

References

- Anthropic. 2024. [Introducing the model context protocol](#).
- Anthropic. 2025. [Equipping agents for the real world with agent skills](#).
- Mohammad Gheshlaghi Azar, Mark Rowland, Bilal Piot, Daniel Guo, Daniele Calandriello, Michal Valko, and Rémi Munos. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*.
- Kaiyuan Chen, Yixin Ren, Yang Liu, Xiaobo Hu, Haotong Tian, Tianbao Xie, Fangfu Liu, Haoye Zhang, Hongzhang Liu, Yuan Gong, Chen Sun, Han Hou, Hui Yang, James Pan, Jianan Lou, Jiayi Mao, Jizheng Liu, Jinpeng Li, Kangyi Liu, et al. 2025a. [xbench: Tracking agents productivity scaling with profession-aligned real-world evaluations](#). *arXiv preprint arXiv:2506.13651*.
- Shan Chen, Pedro Moreira, Yuxin Xiao, Sam Schmidgall, Jeremy Warner, Hugo Aerts, Thomas Hartvigsen, Jack Gallifant, and Danielle S. Bitterman. 2025b. [MedBrowseComp: Benchmarking medical deep research and computer use](#). *arXiv preprint arXiv:2505.14963*.
- Gheorghe Comanici et al. 2025. [Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities](#). *arXiv preprint arXiv:2507.06261*.
- CURE-Bench Organizers. 2025. [CURE-Bench at NeurIPS 2025: Benchmarking AI reasoning for therapeutic decision-making at scale](#).
- Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. KTO: Model alignment as prospect theoretic optimization. In *International Conference on Machine Learning*.
- Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. PubMedQA: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577. Association for Computational Linguistics.
- Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. [Step-DPO: Step-wise preference optimization for long-chain reasoning of LLMs](#). *arXiv preprint arXiv:2406.18629*.
- Mukai Li, Qingcheng Zeng, Tianqing Fang, Zhenwen Liang, Linfeng Song, Qi Liu, Haitao Mi, and Dong Yu. 2026. [Verified critical step optimization for LLM agents](#). In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 39627–39639, San Diego, California, United States. Association for Computational Linguistics.

- Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let’s verify step by step. In *International Conference on Learning Representations*.
- Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2025. [Critical tokens matter: Token-level contrastive estimation enhances LLM’s reasoning capability](#). In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 37906–37918. PMLR.
- Jonathan Lowin. 2024. [FastMCP: The fast, pythonic way to build MCP servers and clients](#).
- Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. SimPO: Simple preference optimization with a reference-free reward. *Advances in Neural Information Processing Systems*.
- Marvin Minsky. 1961. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30.
- OpenAI. 2025a. [Introducing GPT-4.1 in the API](#).
- OpenAI. 2025b. [Introducing GPT-5.2](#).
- Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. [Agent Q: Advanced reasoning and learning for autonomous AI agents](#). *arXiv preprint arXiv:2408.07199*.
- Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. [ToolLLM: Facilitating large language models to master 16000+ real-world APIs](#). In *International Conference on Learning Representations*.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*.
- Marc’Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen-tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, et al. 2025. [DR Tulu: Reinforcement learning with evolving rubrics for deep research](#). *arXiv preprint arXiv:2511.19399*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. [DeepSeekMath: Pushing the limits of mathematical reasoning in open language models](#). *arXiv preprint arXiv:2402.03300*.
- Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. 2024. [Trial and error: Exploration-based trajectory optimization of LLM agents](#). In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7584–7600. Association for Computational Linguistics.
- Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. 2024. [Watch every step! LLM agent learning via iterative step-level process refinement](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1556–1572. Association for Computational Linguistics.
- An Yang et al. 2025. [Qwen3 technical report](#). *arXiv preprint arXiv:2505.09388*.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.
- Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gao-hong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, et al. 2025. [DAPO: An open-source LLM reinforcement learning system at scale](#). *arXiv preprint arXiv:2503.14476*.
- Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. 2024. Token-level direct preference optimization. In *International Conference on Machine Learning*.

A Additional Experiments and Implementation Details

This appendix incorporates the experiments and clarifications reported during author response.

A.1 Skill Acquisition Details

Each tool is represented as a Markdown skill card containing its signature, docstring, argument

schema, and approximately 50–200 lines of implementation. We collect roughly 500 historical invocations per tool and cluster failures by error type. An LLM then summarizes tool-specific strategies from these clusters; recurring failure patterns are compiled automatically into pre- and post-call assertions rather than hand-written benchmark rules. Examples include: decomposing PubMed queries into 2–4 core keywords and simplifying a query after a zero-result response; routing from `get_trial_info` to `get_drug_patents` while copying identifiers exactly; and prioritizing `fda_drug_search` with the exact drug name. No benchmark labels or answers enter this process.

A.2 Cross-Model and Cross-Domain Evaluation

Table 4 evaluates Qwen3-4B-Thinking (Yang et al., 2025) on CURE-Bench (CURE-Bench Organizers, 2025) and the general-domain xbench-DeepSearch benchmark (Chen et al., 2025a). GPT-5.2 (OpenAI, 2025b) supplies expert trajectories for both SFT distillation and CDPO. The LLM-judge variant selects critical steps with an independent judge; across five repeated evaluations of the same trajectory, its selected-step overlap with the rollout-divergence selector is below 35%, indicating that the two selectors are not interchangeable.

Table 4: Qwen3-4B-Thinking results on CURE-Bench and xbench-DeepSearch (accuracy, %).

Method	CURE-Bench	xbench-DeepSearch
Base	41.30	4.0
SFT (GPT-5.2 distillation)	48.20	10.0
IPR	52.40	11.0
ETO	53.00	13.0
LLM-Judge CDPO	53.60	14.0
GRPO (Shao et al., 2024)	54.30	16.0
DAPO (Yu et al., 2025)	54.50	17.0
CSO (Li et al., 2026)	54.80	17.0
Step-DPO (Lai et al., 2024)	55.40	19.0
cDPO (Lin et al., 2025)	56.70	15.0
Skill-CDPO	59.10	21.0

A.3 Pair Selection and Criticality Threshold

Table 5 compares the default cross-group construction with two single-pair controls. Table 6 varies the criticality threshold. The default $\epsilon = 0.3$ retains 15.7% of candidates, with 2.67 verified steps per trajectory on average, and gives the best CURE-Bench result. Very permissive thresholds retain noisy steps; very strict thresholds leave too little training signal.

Table 5: Pair-construction comparison.

Construction	PubMed	Cure	MedBrowse
All cross-group pairs	19.58	66.63	29.09
Maximum-reward-gap pair	18.62	65.88	28.45
Random pair	19.07	65.62	28.38

Table 6: Criticality-threshold sensitivity on CURE-Bench.

ϵ	Retained	Steps/traj.	Score
0.1	30.8%	3.74	62.15
0.2	26.1%	3.21	64.82
0.3	15.7%	2.67	66.63
0.4	4.9%	1.61	61.40
0.5	2.6%	0.96	60.05

A.4 Cost and Latency

Table 7 reports measured end-to-end resource use. CDPO’s cost is front-loaded into offline suffix synthesis: $16,951 \times 4.5 \times 8 \approx 610\text{K}$ completions. GRPO uses little external API budget but substantially more accelerator time.

Table 7: Approximate resource profile of CDPO and GRPO. Dollar totals use the experiment-time API and compute prices.

Method	API	A100-h	Total	Wall time
CDPO	\$1,027	48	~\$1.15K	~6 h
GRPO	Low	~1,100	~\$2.8K	~3–5 d

A.5 Benchmark Quality and Decontamination

The CURE-Bench evaluation uses its 459-item Phase 1 validation split: 183 `multi_choice`, 230 `open_ended_multi_choice`, and 46 `open_ended` questions.

PubMed Search contains a 711-question Full set and a 147-question Lite set spanning 50 biomedical topic buckets and six question types: efficacy comparison, mechanism, prognosis, epidemiology, review, and drug interaction. Full has 5.31 content rubrics and 9.31 total rubrics per item on average; Lite has 5.78 and 7.78, respectively. In Full, 3,757 of 3,777 content-rubric items include source and confidence annotations.

To reduce evaluation leakage, we remove benchmark questions, evaluation rubrics, and PMID entries from training data. We additionally filter overlapping NCT identifiers, drug and disease names, and key relation tuples. For MedBrowseComp, we exclude HemOnc.org from all training-time retrieval. These filters reduce direct overlap but can-

not rule out indirect semantic overlap in public pretraining corpora.

B Discriminative Critical Turn Localization Algorithm

For the multiple-choice items of CURE-Bench, which provide both an option set and a definitive ground-truth answer, we supplement the observable-feature heuristics with a *discriminative critical turn localization* procedure that identifies the specific reasoning turn where the agent’s trajectory diverges from the correct answer, and classifies the error into one of four mechanistic subtypes. The entire pipeline is rule-based with no LLM calls.

Stage 1: Discriminative Keyword Extraction

Given the option set $\mathcal{O} = \{o_1, \dots, o_K\}$ for a multiple-choice question, we extract a weighted keyword set for each option. For option o_i with text t_i , we tokenize t_i into lowercase alphanumeric tokens and remove a standard English stopword list. Crucially, we impose *no minimum token length*, since short but highly discriminative medical abbreviations (e.g., “HIV”, “CT”, “CRP”) would otherwise be discarded.

We then partition each option’s keyword set $\mathcal{K}_i$ into:

$$\mathcal{U}_i = \mathcal{K}_i \setminus \bigcup_{j \neq i} \mathcal{K}_j \quad (\text{unique keywords}) \quad (12)$$

$$\mathcal{S}_i = \mathcal{K}_i \setminus \mathcal{U}_i \quad (\text{shared keywords}) \quad (13)$$

Unique keywords appear only in option o_i and carry high discriminative value; shared keywords appear in multiple options and are down-weighted. Each keyword k is assigned weight $w(k) = 1.0$ if $k \in \mathcal{U}_i$ and $w(k) = 0.3$ if $k \in \mathcal{S}_i$.

Stage 2: Per-Turn Option Tendency Scoring

The agent trajectory is parsed into a sequence of turns $\mathcal{T} = [t_1, \dots, t_N]$, where each turn contains a reasoning segment think_j (content within `<think>` tags), one or more tool calls, and their corresponding outputs. For each turn t_j and option o_i , we compute a tendency score:

$$\begin{aligned} \text{score}(t_j, o_i) = & \sum_{k \in \mathcal{U}_i} \text{count}(k, \text{think}_j) \\ & + 0.3 \sum_{k \in \mathcal{S}_i} \text{count}(k, \text{think}_j) \end{aligned} \quad (14)$$

where $\text{count}(k, \cdot)$ counts case-insensitive occurrences of keyword k .

Stage 3: Trajectory-Based Critical Turn Localization

Rather than simply finding the first turn where the wrong option dominates, we analyze the full score trajectory via a *tendency gap*:

$$\text{gap}[j] = \text{score}(t_j, o_{\text{wrong}}) - \text{score}(t_j, o_{\text{correct}}) \quad (15)$$

where o_{wrong} is the model’s (incorrect) answer and o_{correct} is the ground truth. A negative gap indicates the model leans toward the correct answer at turn j ; a positive gap indicates a lean toward the wrong answer.

We compute two key trajectory anchors:

- **Last Correct Turn (LCT)**: the last index j where $\text{gap}[j] < 0$, i.e., the correct answer was still favored. If no such turn exists, $\text{LCT} = \emptyset$.
- **Maximum Shift Turn (MST)**: the index $j \geq 1$ maximizing the positive delta $\text{gap}[j] - \text{gap}[j-1]$, i.e., the turn where the model’s tendency shifted most sharply toward the wrong answer. If all deltas ≤ 0 , $\text{MST} = \emptyset$.

The critical turn interval is then:

- $\text{LCT} \neq \emptyset$ and $\text{MST} \neq \emptyset$: interval $[\text{LCT}, \text{MST}]$ (the full transition zone).
- Only $\text{MST} \neq \emptyset$: interval $[\text{MST} - 1, \text{MST}]$.
- Only $\text{LCT} \neq \emptyset$: interval $[\text{LCT}, \text{LCT} + 1]$.
- Neither: last 2 turns (fallback).

Stage 4: Error Sub-Classification

We additionally detect tool-output evidence signals: for each turn t_j , let $\text{cev}[j] = \mathbf{1}[\exists k \in \mathcal{U}_{\text{correct}} : k \in \text{tool_output}_j]$ indicate whether the tool output contains a unique keyword of the correct option. Define $\text{AnyCorrectEv} = \bigvee_j \text{cev}[j]$ and $\text{CorrectEverFavored} = (\text{LCT} \neq \emptyset)$.

The four subtypes are determined by the following decision rule:

Correct Ever Favored	MST	Correct Ev.	Subtype
True	$\neq \emptyset$	—	MISLED-BY-EVIDENCE
True	$= \emptyset$	—	FINAL-REASONING-ERROR
False	—	True	EVIDENCE-IGNORED
False	—	False	INSUFFICIENT-SEARCH

MISLED-BY-EVIDENCE: the model initially favored the correct answer but was derailed at MST, typically by a failed or misleading tool call. Critical turns: $[\text{LCT}, \text{MST}]$.

FINAL-REASONING-ERROR: the model consistently favored the correct answer (no sharp shift) but output the wrong letter. Critical turns: last 2 turns.

EVIDENCE-IGNORED: correct-answer evidence appeared in tool outputs but was never incorporated into reasoning. Critical turns: the turn(s) where $\text{cev}[j] = 1$ and the immediately following turn.

INSUFFICIENT-SEARCH: no correct-answer evidence was retrieved at any point. Critical turns: first turn and the turn with the highest wrong-option evidence signal.

C Additional Theoretical Analysis

C.1 False Positive Bound for Critical Step Selection

A natural concern is whether the thresholding procedure $\Delta_t/A > \epsilon$ may incorrectly flag non-critical steps. We provide a formal bound on this false positive rate.

Proposition 3 (False Positive Bound). *Suppose the true criticality gap is $\Delta_t^* = 0$ (step t is not critical). With $n = m = k$ rollouts per group, the probability of a false positive detection $\hat{\Delta}_t/A > \epsilon$ is bounded by:*

$$P(\hat{\Delta}_t/A > \epsilon \mid \Delta_t^* = 0) \leq \exp\left(-\frac{k\epsilon^2}{2}\right) \quad (16)$$

Proof. Under $\Delta_t^* = 0$, we have $\mathbb{E}[\hat{\Delta}_t] = 0$. Define paired differences $D_i = r_{t,i}^+ - r_{t,i}^- \in [-A, +A]$, so that $\hat{\Delta}_t = \frac{1}{k} \sum_{i=1}^k D_i$. By Hoeffding’s inequality:

$$\begin{aligned} P(\hat{\Delta}_t/A > \epsilon) &\leq \exp\left(-\frac{2k^2(A\epsilon)^2}{k \cdot (A - (-A))^2}\right) \\ &= \exp\left(-\frac{k\epsilon^2}{2}\right) \quad \square \end{aligned} \quad (17)$$

Corollary 1. *For binary-outcome tasks ($A = 1$) with $k = 4$ rollouts per group and threshold $\epsilon = 0.5$: false positive rate $\leq e^{-0.5} \approx 0.61$. With $k = 8$: $\leq e^{-1.0} \approx 0.37$. With $k = 16$: $\leq e^{-2.0} \approx 0.14$.*

This conservative bound quantifies how the worst-case false-positive probability decreases as k or ϵ increases; for small rollout budgets it can remain loose.

C.2 Full Proofs

Proof of Proposition 1 (Two-Sample Variance Decomposition). Let $h(X_i, Y_j)$ be the per-pair

gradient and $\mu = \mathbb{E}[h(X, Y)]$. Its two-sample Hoeffding decomposition is

$$h(X, Y) - \mu = h_X(X) + h_Y(Y) + h_R(X, Y), \quad (18)$$

where $h_X(X) = \mathbb{E}[h(X, Y) \mid X] - \mu$, $h_Y(Y) = \mathbb{E}[h(X, Y) \mid Y] - \mu$, and the residual has zero conditional expectation given either argument. Averaging over the cross product gives

$$\begin{aligned} \hat{g}_{n,m} - \mu &= \frac{1}{n} \sum_i h_X(X_i) + \frac{1}{m} \sum_j h_Y(Y_j) \\ &\quad + \frac{1}{nm} \sum_{i,j} h_R(X_i, Y_j). \end{aligned} \quad (19)$$

The three terms are uncorrelated by construction. Independence within each rollout sample and conditional degeneracy of h_R therefore yield

$$\begin{aligned} \text{Var}(\hat{g}_{n,m}) &= \frac{\text{Var}(h_X)}{n} + \frac{\text{Var}(h_Y)}{m} \\ &\quad + \frac{\text{Var}(h_R)}{nm}. \end{aligned} \quad (20)$$

Setting these components to $\sigma_X^2, \sigma_Y^2, \sigma_R^2$ proves the claim. $\square$

Proof of Proposition 2 (Unbiased Criticality Estimation). Since $r_{t,i}^+$ are i.i.d. draws from the expert rollout distribution with mean $V_{\pi_{\text{exp}}}(s_t)$, the sample mean $\hat{V}_{\text{exp}}(s_t) = \frac{1}{n} \sum_{i=1}^n r_{t,i}^+$ is unbiased by linearity of expectation. The same holds for $\hat{V}_{\text{local}}(s_t)$. Since the two groups of rollouts are independent:

$$\begin{aligned} \mathbb{E}[\hat{\Delta}_t] &= \mathbb{E}[\hat{V}_{\text{exp}}(s_t)] - \mathbb{E}[\hat{V}_{\text{local}}(s_t)] \\ &= V_{\pi_{\text{exp}}}(s_t) - V_{\pi_{\theta}}(s_t) = \Delta_t^* \end{aligned} \quad (21)$$

The variance formula follows from independence and the standard variance of a sample mean: $\text{Var}[\hat{\Delta}_t] = \text{Var}[r_t^+]/n + \text{Var}[r_t^-]/m$. $\square$

D Comparison with Step-Level Preference Methods

Table 8 provides a systematic comparison of CDPO with existing step-level preference optimization methods.

Key advantages of CDPO over prior step-level methods:

1. **No PRM required.** Criticality is measured directly from task outcomes via rollout divergence, eliminating the need to train or maintain a separate process reward model.

Table 8: Comparison of step-level preference learning methods. CDPO is the only method that combines outcome-verified step selection (no PRM), distributional supervision from multiple rollouts, and continuous criticality weighting.

Method	Step Selection	Signal Source	Training Signal	Reward Model
Trajectory DPO (ETO)	None (full trajectory)	Outcome	Coarse, uniform	Not required
Step-DPO	PRM score	PRM estimate	Dense, noisy	Required
IPR	Outcome-verified	MC rollout	Step-level, binary	Not required
CSO	PRM + verification	Outcome	Binary weight	Required
CDPO (Ours)	Rollout divergence	Outcome (verified)	Distributional, weighted	Not required

- Distributional supervision.** All nm rollout outcomes contribute to the training signal through the all-pairs formulation, rather than using a single “best” pair.
- Continuous weighting.** The criticality weight $w(\Delta_t)$ encodes the degree of criticality, unlike binary select/reject schemes.
- Statistical characterization.** Propositions 1 and 2 characterize the two-sample variance and criticality estimator.

E CDPO Weight Design: Detailed Examples

E.1 Step-Level Weight Allocation Example

To illustrate how the normalized step weight $w(\Delta_t) = \Delta_t^\alpha / \hat{Z}$ redistributes gradient budget, consider a concrete example with $\alpha = 1$ and rubric maximum $A = 40$. Suppose critical-step selection retains five steps with the following empirical criticality scores:

Table 9: Step-level weight allocation example ($\alpha = 1$, $A = 40$). The normalization constant $\hat{Z} = (2 + 8 + 15 + 3 + 12)/5 = 8.0$ ensures weights average to 1. Steps with larger divergence receive proportionally more gradient update.

Step	Δ_t (pts)	$\hat{Z}$	$w(\Delta_t)$	Verdict
Step 1	2	8.0	$0.25\times$	Downweighted ↓
Step 2	8	8.0	$1.00\times$	Average
Step 3	15	8.0	$1.875\times$	Upweighted ↑↑
Step 4	3	8.0	$0.375\times$	Downweighted ↓
Step 5	12	8.0	$1.50\times$	Upweighted ↑

The sum of weights is $0.25 + 1.00 + 1.875 + 0.375 + 1.50 = 5.0$, confirming the mean-1 property ($5.0/5 = 1.0$). Step 3 has the largest divergence—the local model fails badly while the expert succeeds—and receives the strongest gradient update ($1.875\times$ standard DPO). Step 1 has nearly no divergence between models and is effectively downweighted ($0.25\times$).

Figure 2 visualizes this redistribution. The left panel shows raw criticality scores with the normalization constant $\hat{Z} = 8.0$ (dashed), while the right panel shows the resulting normalized weights centered around the mean-1 baseline.

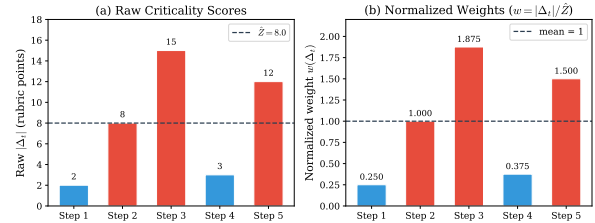


Figure 2: Step-level weight allocation. (a) Raw criticality scores Δ_t for the selected steps with the normalization constant $\hat{Z} = 8.0$ (dashed). (b) Normalized weights $w(\Delta_t) = \Delta_t / \hat{Z}$, averaging to 1 (dashed). Red bars indicate upweighted steps; blue bars indicate downweighted steps.

E.2 Scale Invariance of Normalized Weights

A key property of the $\hat{Z}$ -normalized weights is that they are completely unaffected by the rubric scale A . If the same task used $A' = 2A$ instead of A , every criticality score doubles, but so does $\hat{Z}$:

$$\begin{aligned} \Delta_t &\rightarrow 2\Delta_t, \quad \hat{Z} \rightarrow 2\hat{Z} \\ \Rightarrow w(\Delta_t) &= \frac{\Delta_t^\alpha}{\hat{Z}} \text{ unchanged} \end{aligned} \quad (22)$$

This means one can switch rubrics (e.g., from 40-point to 100-point grading) without retuning any hyperparameters. $\hat{Z}$ acts as an automatic unit converter, absorbing the rubric scale.

Figure 3 illustrates this property. Although the raw scores double when switching from $A = 40$ to $A = 80$, the normalized weights remain identical.

E.3 Weighting Exponent α : Design Choices

The exponent α in Δ_t^α controls how aggressively the weight function discriminates between selected steps of different criticality. Table 10 summarizes the design space.

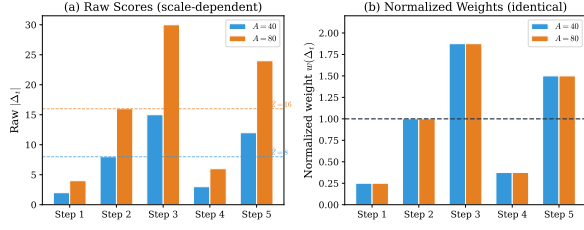


Figure 3: Scale invariance of $\hat{Z}$ -normalized weights. (a) Raw criticality scores under $A = 40$ and $A = 80$: all values double, as do $\hat{Z}$. (b) Normalized weights are identical regardless of rubric scale, confirming that $\hat{Z}$ absorbs A automatically.

Table 10: Design choices for the weighting exponent α . All choices satisfy the mean-1 property; they differ only in how sharply they discriminate between steps.

α	Shape	Effect	When to use
0.5	Concave	Small divergences get significant weight	Noisy estimates; avoid missing signals
1	Linear	Proportional to normalized gap	Default — simple and interpretable
2	Convex	Near-zero gaps strongly suppressed	High-quality rollouts; sharp focus

We recommend starting with $\alpha = 1$. If rollout estimates are noisy (small n, m), $\alpha = 0.5$ avoids over-emphasizing steps with accidentally large $\hat{\Delta}_t$. If rollouts are plentiful and reliable, $\alpha = 2$ concentrates learning most aggressively on the hardest bottlenecks.

Figure 4 plots the normalized weight function for each α , showing the concave, linear, and convex shapes respectively.

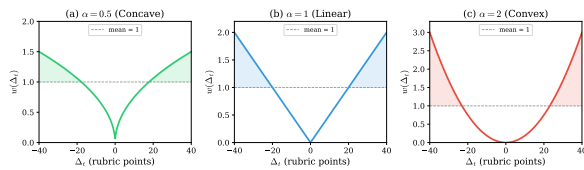


Figure 4: Weight function shapes for different α values. (a) $\alpha = 0.5$: concave—small divergences still receive significant weight. (b) $\alpha = 1$: linear—weight proportional to normalized gap. (c) $\alpha = 2$: convex—near-zero gaps are strongly suppressed. The dashed line marks $w = 1$ (average weight).

E.4 Pair-Level Score Weight Example

Within each critical step, individual rollout pairs also differ in informativeness. We first orient each pair so that the higher-scoring action is preferred, and then assign the nonnegative weight $q_{t,ij} = |r_{t,i}^+ - r_{t,j}^-|/A$. Ties receive zero weight. Table 11 summarizes the three cases.

Table 11: Three cases handled by pair orientation and the nonnegative score-gap weight $q_{t,ij}$.

Condition	$q_{t,ij}$	Effect
$r_{t,i}^+ > r_{t,j}^-$	$(r_{t,i}^+ - r_{t,j}^-)/A$	Prefer a^+
$r_{t,i}^+ = r_{t,j}^-$	0	Zero contribution
$r_{t,i}^+ < r_{t,j}^-$	$(r_{t,j}^- - r_{t,i}^+)/A$	Prefer a^-

Table 12 shows a concrete 4×4 pair weight matrix for a critical step with $A = 40$.

Table 12: Absolute pair-level score-gap weights for $n = m = 4$ rollouts ($A = 40$). Expert scores: 38, 32, 35, 29. Local scores: 18, 25, 12, 22. Larger score gaps receive greater weight.

	Local 1 (18)	Local 2 (25)	Local 3 (12)	Local 4 (22)
Expert 1 (38)	0.50	0.33	0.65	0.40
Expert 2 (32)	0.35	0.17	0.50	0.25
Expert 3 (35)	0.42	0.25	0.57	0.33
Expert 4 (29)	0.28	0.10	0.42	0.17

Pairs with larger score gaps (e.g., Expert 1 vs. Local 3: $q = 0.65$) provide stronger supervision, while near-tied pairs (e.g., Expert 4 vs. Local 2: $q = 0.10$) contribute minimally. Tied pairs remain in the fixed all-pairs average but have zero contribution.

Figure 5 visualizes the full 4×4 pair weight matrix as a heatmap, where darker cells correspond to more informative pairs.

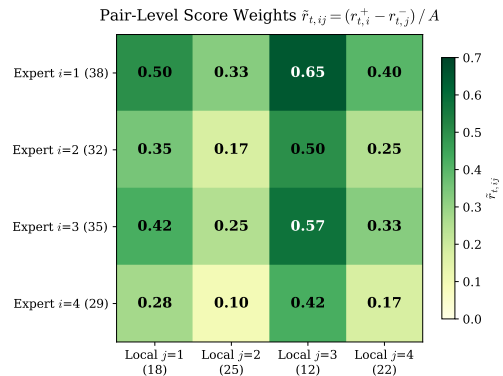


Figure 5: Pair-level score weight heatmap for $n = m = 4$ rollouts ($A = 40$). Each cell shows $q_{t,ij} = |r_{t,i}^+ - r_{t,j}^-|/A$. Darker green indicates larger score gaps and stronger preference supervision.

E.5 Binary QA vs. Rubric CDPO: Component-Level Comparison

Table 13 summarizes the component-level differences between the binary QA version of CDPO and

the rubric-based extension presented in this work.

The rubric loss makes step weights scale-free and mean-1 through $\hat{Z}$; the fixed $1/nm$ average removes variation in the pair normalizer; and explicit orientation with the absolute score gap provides continuous, partial-credit supervision across all nm rollout combinations.

F Iterative Refinement

The CDPO framework naturally supports iterative refinement. After each training round, the updated policy π_{θ_i} can be deployed to collect fresh trajectories, and the full critical step identification and preference construction pipeline is repeated:

$$\pi_{\theta_0} \xrightarrow{\mathcal{D}_{\text{pref}}^{(0)}} \pi_{\theta_1} \xrightarrow{\mathcal{D}_{\text{pref}}^{(1)}} \pi_{\theta_2} \xrightarrow{\dots} \quad (23)$$

At iteration i , the reference model is set to $\pi_{\text{ref}} = \pi_{\theta_{i-1}}$ for stable KL penalization. As the policy improves, previously critical steps become non-critical (the local model has learned to handle them), and new, harder decision points emerge as critical. This induces a natural curriculum learning effect: early iterations address frequent, basic errors (e.g., malformed queries), while later iterations target subtler deficiencies (e.g., suboptimal information synthesis strategies). We leave empirical validation of multi-round iterative CDPO to future work.

G Discussion on Weight Design Choices

Split Estimation for Selection and Weighting.

The linear weighting $w(\Delta_t) = \Delta_t/\hat{Z}$ is theoretically motivated by advantage-weighted regression, but introduces a subtle correlation: the same rollouts are used both to select critical steps ($\Delta_t/A > \epsilon$) and to weight them ($w(\Delta_t)$). This can create a positive bias—steps selected because of a lucky high $\hat{\Delta}_t$ also receive inflated weight. A principled solution is split estimation: partition the rollout budget into two independent batches, using one for selection and the other for weighting. This eliminates the correlation at the cost of doubling the rollout budget per candidate step. In practice, we find that the bias is small relative to other sources of noise, and the simpler single-batch approach suffices.

Optimal Rollout Budget Allocation. Propositions 3 and 1 suggest a tradeoff: larger n, m can reduce false positives and gradient variance, but increases compute per candidate step. An adaptive

allocation strategy—running more rollouts at high-uncertainty candidates (where $\hat{\Delta}_t/A$ is close to ϵ) and fewer at clearly critical or clearly non-critical steps—could optimize this tradeoff automatically. We leave the exploration of such adaptive schemes to future work.

H PubMed Search Benchmark Construction

Existing PubMed-oriented benchmarks, such as PubMedQA (Jin et al., 2019), were developed under a closed-form question-answering paradigm: each instance has a definitive answer, and evaluation reduces to binary or categorical matching. While useful for measuring factual recall, this format cannot assess a model’s ability to formulate effective search strategies, iteratively refine queries, synthesize information from multiple retrieved articles, and produce well-attributed, comprehensive responses—capabilities that are central to real-world biomedical literature research. To fill this gap, we introduce **PubMed Search**, a deep research benchmark built around the PubMed API that evaluates open-ended, multi-step retrieval and synthesis.

Question Generation. We sample seed topics from the Medical Subject Headings (MeSH) thesaurus to ensure broad coverage of biomedical sub-disciplines. For each seed topic, GPT-5.2 is prompted to generate an open-ended, complex research question that requires consulting multiple primary sources and synthesizing evidence across studies—for example, comparing treatment efficacy across populations, summarizing mechanistic controversies, or identifying gaps in the current literature.

Multi-Model Rollout. To avoid the systematic biases inherent in single-model data construction, we employ three heterogeneous frontier models—GPT-5.2 (OpenAI, 2025b), Gemini 2.5 Pro (Cohan et al., 2025), and Qwen3-235B-A22B (Yang et al., 2025)—each equipped with the same MCP tool suite. Every model independently performs two complete rollouts per question, yielding six trajectories that include the full sequence of tool invocations, intermediate reasoning, and a final synthesized answer.

Cross-Model Rubric Construction. A multi-agent deliberation is then conducted among the same three models. Each model reviews all six

Table 13: Component-level comparison between binary QA CDPO and rubric-based CDPO. The rubric formulation replaces discrete components with continuous alternatives, eliminating edge cases and providing richer supervision.

Component	Binary QA	Rubric (this work)
Criticality score Δ_t	$\hat{V}_{\text{exp}} - \hat{V}_{\text{local}} \in [0, 1]$	$\bar{r}_t^+ - \bar{r}_t^- \in [-A, +A]$
Step weight	$w(\Delta_t)$ raw or binary	$\Delta_t^\alpha / \hat{Z}$ on selected steps
Pair selection	$ \mathcal{P}_t $ pairs where $y^+ = 1, y^- = 0$	All nm pairs
Pair normalizer	$1/ \mathcal{P}_t $ (variable, correlated with Δ_t)	$1/nm$ (constant)
Pair weight	1 (binary)	$ r_{t,i}^+ - r_{t,j}^- /A \in [0, 1]$
Preference direction	Positive-vs. negative outcome	Higher empirical score

rollout answers, and the models engage in a structured discussion to (i) identify the factual claims and evidence threads that should be included in a reference answer, and (ii) construct a rubric consisting of weighted content criteria derived from these claims. This cross-model consensus procedure mitigates the risk that the rubric reflects the idiosyncratic strengths or blind spots of any single model.

Format Rubrics. In addition to the content-based rubric items, the evaluation protocol includes four format rubrics that assess process quality: (1) *tool utilization breadth*, requiring the model to make substantive use of the available search tools rather than relying on parametric knowledge alone; (2) *citation presence*, requiring the final answer to include specific literature references; (3) *citation format compliance*, requiring citations to follow a prescribed format (e.g., PMID-based references); and (4) *citation faithfulness*, penalizing fabricated or hallucinated references that do not correspond to real PubMed articles. These format criteria ensure that high scores reflect not only factual correctness but also methodological rigor in evidence retrieval and attribution.

I Representative Error Cases by Subtype

The following cases are drawn from real model trajectories on CURE-Bench. Each illustrates one of the four error subtypes identified by the algorithm in Appendix B. Gap trajectory values are per-turn tendency gaps: $\text{gap}[j] = \text{score}(t_j, o_{\text{wrong}}) - \text{score}(t_j, o_{\text{correct}})$; negative values indicate the model leaning toward the correct answer.

MISLED-BY-EVIDENCE. *Question:* “Which requirement is part of the THALOMID REMS program to ensure safe prescribing and dispensing?” *Correct:* A (*Prescribers must be certified with the THALOMID REMS program*). Model chose: C (*Dispensing by any licensed community pharma-*

cist). Turns: 10. LCT = T9, MST = T4 ($\delta = +3.0$). Gap trajectory:

$$[+2.0, +2.0, +2.0, -2.0, +1.0, -4.0, -1.0, -3.0, -3.0, -1.0]$$

The model correctly shifted toward option A at T3 (gap = -2.0) after retrieving relevant REMS information. However, the T3 FDA search returned an error (No FDA data found for drug: Thalidomide THALOMID REMS), causing the model to discard its intermediate conclusion and restart reasoning at T4, erasing the correct lean. Despite recovering and favoring A in T5–T9, the model ultimately selected C. *Takeaway:* a tool failure caused the model to abandon a correct intermediate conclusion and re-open all options as equally plausible.

FINAL-REASONING-ERROR. *Question:* “What are the approved indications for Dexmedetomidine Injection and Dexmedetomidine in 5% Dextrose Injection?” *Correct:* C (*Sedation of intubated/mechanically ventilated adult ICU patients ≤ 24 h and sedation of non-intubated adult patients prior/during procedures*). Model chose: A (*Sedation of pediatric patients undergoing diagnostic procedures and treatment of acute pain*). Turns: 4. Gap trajectory: $[-2.0, -2.7, -4.7, -5.7]$ (monotonically decreasing, consistently favoring C).

The model’s answer text reads: “*The FDA-labeled indications . . . are sedation of initially intubated, mechanically ventilated adult ICU patients (continuous infusion, not to exceed 24 hours) and sedation of non-intubated adult patients prior to/during procedures*”—a verbatim description of option C. Yet the extracted answer letter is A. *Takeaway:* the reasoning trajectory is entirely correct; the error is a failure to map the described content to the correct option letter during answer extraction.

EVIDENCE-IGNORED. *Question:* “Which drug was shown to block the tumorigenic effect of albuterol sulfate in Sprague-Dawley rats?” *Cor-*

rect: A (*Propranolol*). Model chose: C (*Atenolol*). Turns: 2. Gap trajectory: [0.0, 0.0]. The keyword *propranolol* appeared in both tool outputs: T0 returned a PubMed result contrasting propranolol (non-selective) with atenolol and metoprolol (selective β_1 -blockers); T1 returned a Sprague-Dawley rat study involving propranolol. The model dismissed both as “general beta-blocker pharmacology” and continued searching for a more specific study, never connecting the retrieved evidence to the question. *Takeaway*: an overly strict relevance threshold caused the model to discard evidence that was in fact sufficient to answer the question.

INSUFFICIENT-SEARCH. *Question*: “For which type of transplant is Everolimus indicated, with the specification that it should be administered no earlier than 30 days posttransplant?” Correct: A (*Liver transplant in adult patients*). Model chose: D (*Kidney transplant in patients at high immunologic risk*). Turns: 2. Gap trajectory: [+3.0, +3.0]. The model confused everolimus with sirolimus from the first turn and issued 584 total queries (only 9 unique), all variants of “sirolimus extended-release capsules indication renal transplant”. The correct answer required searching for “everolimus liver transplant”, but no such query was ever issued. All retrieved evidence pointed to renal transplant indications for sirolimus, reinforcing the wrong answer throughout. *Takeaway*: an early drug-name confusion led to a misdirected query loop; the correct answer’s key concept (*liver*) was never queried.

J Tool Suite Description

Our agent interacts with the environment through a suite of MCP (Model Context Protocol) tools. These tools are shared across all experimental conditions (base model, Skill, SFT, DPO variants, and CDPO) to ensure fair comparison. Table 14 summarizes the tool availability per benchmark, and the following paragraphs describe each tool.

General-Purpose Tools. Two tools are shared across all three benchmarks. **browse_webpage** fetches the content of a given URL and returns it in Markdown format, serving as a general-purpose reader for FDA drug label pages, PubMed articles, and clinical trial registries. **google_search** performs web search via the Serper API and returns a ranked list of results (title, snippet, URL), primarily used as a fallback when specialized tools cannot satisfy the information need.

CURE-Bench Tools. In addition to the general-purpose tools, CURE-Bench uses **fda_drug_search**, which queries the FDA drug label database given a drug name and an aspect of interest (e.g., adverse reactions, indications, dosage, contraindications, drug interactions, pregnancy safety). It leverages LLM-assisted extraction to return the relevant section from the drug’s prescribing information.

MedBrowseComp Tools. MedBrowseComp employs five specialized tools. **medbrowsecomp_search** is a unified medical information search tool with automatic routing: it dispatches NCT identifier queries to the clinical trial API and drug ingredient queries to the appropriate Orange Book endpoint based on the reason parameter (patent, approval, or exclusivity). **get_trial_info** queries ClinicalTrials.gov by NCT number and returns sponsor, recruitment status, intervention ingredients, and source links. **get_drug_patents** queries the FDA Orange Book for patent numbers, jurisdictions, and expiration dates of a given drug. **get_drug_approvals** returns product name, approval date, marketing authorization holder, and approval status (restricted to post-2000 records, sorted by date). **get_drug_exclusivities** returns exclusivity type (e.g., NCE, orphan drug, pediatric), start date, and end date (sorted by expiration).

PubMed Search Tools. The PubMed Search benchmark uses **pubmed_search**, which queries the PubMed database with medical or scientific keywords (recommended 3–6 terms) and returns matching articles with title, authors, abstract, PMID, publication year, and journal name. Pagination is supported via `limit` and `offset` parameters. This tool provides peer-reviewed biomedical literature as the primary evidence source for open-ended question answering.

Table 14: Tool availability across benchmarks. ✓ indicates the tool is available for the corresponding benchmark.

Tool	Cure	MedBrowse	PubMed
fda_drug_search	✓		
pubmed_search			✓
medbrowsecomp_search		✓	
get_trial_info		✓	
get_drug_patents		✓	
get_drug_approvals		✓	
get_drug_exclusivities		✓	
browse_webpage	✓	✓	✓
google_search	✓	✓	✓