BRANCHWISE: A PATH IS NOT A POLICY — UNFOLDING SUCCESSFUL TRAJECTORIES INTO MODEL-TAILORED DECISION BOUNDARIES

Anonymous authors
Paper under double-blind review

ABSTRACT

Tool-use agents make sequential decisions among multiple possible actions, yet successful trajectories expose only one realized path. Direct supervised fine-tuning (SFT) underutilizes their latent supervision: long trajectories entangle local decisions and expose only one reference branch per state, leaving alternative failure directions implicit; static demonstrations do not explicitly reveal model-specific errors; and uniform token-level supervision does not explicitly target the regions that distinguish behaviors. When trained on general tool-use data, SFT can therefore transfer inconsistently and even underperform the base model. We propose BranchWise, a post-training framework that unfolds successful trajectories into model-tailored local decision boundaries. It decomposes trajectories into state-conditioned decisions and constructs verified counterfactuals from tool constraints and execution evidence. These counterfactuals are fused with verifiable error branches generated by probing the initial policy to form compact local decision boundaries. BranchWise localizes comparisons to decision forks and combines evidence-calibrated reference-branch supervision with margin-aware listwise ranking to emphasize insufficiently separated boundaries. With all methods trained from the same general tool-use corpus rather than benchmark-specific supervision, BranchWise outperforms Vanilla SFT across τ^2-Bench, BFCL V3, ToolHop, and ToolTalk-Hard. It achieves the best overall performance among all compared methods, demonstrating stronger and more consistent downstream transfer.

1 INTRODUCTION

Successful trajectories supervise post-training tool-use agents; standard supervised fine-tuning (SFT) imitates their recorded actions. Yet tool use is a branching sequential decision process (Schick

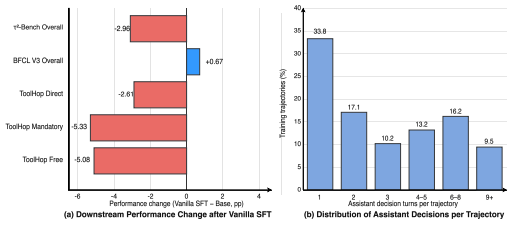


Figure 1: Empirical motivation.

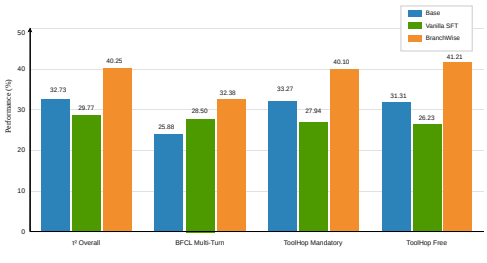


Figure 2: Downstream gains of BranchWise.

et al., 2023; Yao et al., 2023): at each state, the agent must choose among multiple possible actions, whereas a successful trajectory records only the path taken. Imitating general tool-use demonstrations without benchmark-specific supervision may therefore yield unstable downstream transfer (Qin et al., 2024; Patil et al., 2024; Zeng et al., 2024; Chen et al., 2023; Liu et al., 2025).

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.

We analyze Vanilla SFT trajectories and benchmark results (experimental setup in Section 5), which shows changes from the base model of $+0.67$ percentage points on BFCL V3 Overall, -2.96 on τ^2-Bench Overall, and $-2.61/-5.33/-5.08$ on ToolHop Direct/Mandatory/Free (Fig. 1(a)). Training trajectories average 3.83 decision turns—assistant outputs producing tool calls, direct responses, or clarification requests—with 66.2% containing at least two turns and 25.7% at least six (Fig. 1(b)).

This exposes a supervision gap: trajectories contain multiple decision points (Liu et al., 2024; Zhou et al., 2024; Lu et al., 2025), yet each state reveals only the executed action, not incorrect alternatives (Nakano et al., 2022). Prior work adds state- or step-level supervision through state-conditioned preferences (Lightman et al., 2024; Wang et al., 2024; Deng et al., 2025; Zhang et al., 2024; Chen et al., 2025; Khalifa et al., 2025), intermediate-state exploration, and failure-based learning; with linear demonstrations alone, however, alternative actions and local decision boundaries remain unobserved. Thus, *a path is not a policy*: a path records an action, whereas a policy must distinguish candidate behaviors under the same state (Wang et al., 2025)

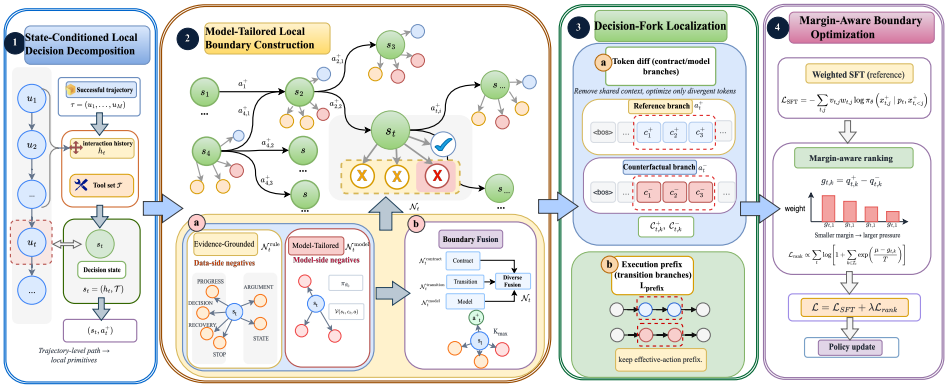


Figure 3: Overview of BranchWise.

Thus, the challenge is to recover verifiable and model-relevant local decision boundaries from a single successful path, beyond splitting trajectories into finer training units. First, demonstrations expose only reference branches; counterfactuals must cover representative failure directions verifiable through task constraints, execution evidence, and environment transitions, while reflecting model-specific error patterns rather than generic task constraints. Second, these boundaries must be optimized where reference and counterfactual branches diverge: most of their content is shared, but behavioral differences concentrate at local decision forks. As boundary separation varies during training, optimization must dynamically focus on distinctions that remain insufficiently learned by the current policy.

We propose **BranchWise**: As shown in Fig. 3, **BranchWise** unfolds successful trajectories into model-tailored local decision boundaries. It decomposes trajectories into state-conditioned decisions and constructs evidence-grounded counterfactuals from tool constraints, execution outcomes, and state changes. Multi-Source Boundary Fusion combines these task-grounded failures with verifiable errors from fixed initial-policy probing, yielding compact boundaries with environmental validity and model relevance. Decision-Fork Localization restricts ranking comparisons to divergent regions; Evidence-Calibrated Reference-Branch Supervision scales reference supervision by verification reliability; and Margin-Aware Local Ranking prioritizes insufficiently separated boundaries. Together, these mechanisms bridge path imitation and local policy structure learning by targeting unresolved behavioral distinctions. Fig. 2 shows our Qwen3-8B results (experimental setup in Section 5). With the same general tool-use training data, BranchWise yields more stable downstream gains: it reverses Vanilla SFT's negative transfer to surpass the base model and further improves performance on datasets where SFT already yields gains.

Our contributions are threefold: **1)** We identify local decision boundaries as the missing structure beyond finer-grained supervision and formulate their verifiable, model-tailored recovery from successful trajectories. **2) BranchWise** uses Multi-Source Boundary Fusion to combine evidence-grounded and model-derived counterfactuals into task- and model-relevant boundaries, learned through localized, evidence-calibrated, and margin-aware optimization. **3)** Across four benchmarks and two

model scales, BranchWise yields stronger, more consistent gains than Vanilla SFT and the evaluated methods; ablations and behavioral analyses support boundary reconstruction and optimization.

2 Related Work

Recent agent post-training moves beyond full-trajectory imitation toward local supervision (Feng et al., 2026; Li et al., 2025c; Jin et al., 2025; Song et al., 2025; Qian et al., 2025; Cheng et al., 2026; Zeng et al., 2025; Li et al., 2023; Patil et al., 2024). TP-LLaMA (Chen et al., 2024) derives step-wise preferences from failed exploratory branches in DFS-generated inference trees, while Agentic-DPO (Chen & Yuille, 2026) samples student actions at expert states for state-conditioned preferences. Yet linear successful trajectories leave surrounding decision boundaries unobserved, so imitation may remain path-specific and the recovery of verifiable failure directions aligned with the model's actual errors remains open. Intermediate-state methods refine learning signals: IPR (Xiong et al., 2024) estimates step-level rewards via Monte Carlo continuations, SPORT (Li et al., 2025b) constructs preferences through step-wise tool exploration and verification, and PivotRL (Yi et al., 2026) applies verifier-based RL from informative states identified by local rollouts. Their locality, however, typically spans a full step, action, or rollout rather than regions inducing behavioral divergence or boundaries still insufficiently separated during training. Failure-driven methods contrast failed explorations with successful trajectories (ETO (Song et al., 2024)) or localize failure-inducing decisions and construct counterfactual interventions or repairs (ToolAnchor (Liu et al., 2026) and CausalFlow Bonagiri et al. (2026)). These methods mainly rely on observed failures, leaving open how to recover useful failure directions from successful demonstrations without first collecting complete failed executions (Nakano et al., 2022; Shinn et al., 2023; Madaan et al., 2023).

3 Preliminaries

3.1 Tool-Use Trajectories and Decision States

A tool-use trajectory is a message sequence $\tau = (u_1, \ldots, u_M)$, where u_i is the i-th message from the user, agent, or tools. The available tool set $\mathcal{T}$ is specified by tool names and argument schemas. Each agent turn may produce a natural-language response, request clarification, or issue one or more tool calls. Let t index agent decision turns, with complete visible history h_t before the action, state $s_t = (h_t, \mathcal{T})$, and demonstrated reference branch a_t^+. BranchWise constructs verified counterfactuals $\mathcal{N}_t = \{a_{t,1}^-, \ldots, a_{t,K_t}^-\}$, where $a_{t,k}^-$ denotes the k-th counterfactual branch and $K_t = |\mathcal{N}_t|$ is the number of counterfactual branches retained for state s_t. All branches share the same interaction history and tool context, differing only in the current agent action, and thus form sibling branches from s_t.

3.2 Training-Side Action Contract

Each decision state s_t is associated with a training-side Action Contract used only for supervision construction: $c_t = (d_t, \kappa_t, \xi_t, \rho_t)$, where d_t denotes the expected decision type, including tool invocation, direct response, and clarification request; κ_t denotes the contract verification status; ξ_t aggregates evidence such as tool constraints, argument provenance, execution outcomes, and state changes; and $\rho_t \in [0, 1]$ denotes the overall confidence. The Action Contract is used only to construct training supervision and is never provided to the model as additional input.

3.3 Model and Sequence Notation

Let θ and θ_0 be the current and initial model parameters. The fixed policy π_{θ_0} probes model-specific errors before training; π_θ computes current reference–counterfactual margins. The input is $p_t = \mathrm{Render}(s_t)$, with deterministic formatting Render. Reference and counterfactual token sequences are $x_t^+ = (x_{t,1}^+, \ldots, x_{t,n_t^+}^+)$ and $x_{t,k}^- = (x_{t,k,1}^-, \ldots, x_{t,k,n_{t,k}^-}^-)$, with lengths n_t^+ and $n_{t,k}^-$, respectively. The prefix before token j is $x_{t,<j}^+$, and analogously for counterfactuals.

4 METHOD

BranchWise decomposes successful trajectories into audited local decisions, fuses evidence-grounded counterfactuals with verified initial-policy errors, and learns model-tailored boundaries through localized, margin-aware ranking and evidence-calibrated reference supervision.

4.1 STATE-CONDITIONED LOCAL DECISION DECOMPOSITION

Trajectory-to-Decision Decomposition. BranchWise decomposes each trajectory into state-conditioned units (s_t, a_t^+), making each s_t a shared branching point for comparing reference and counterfactual actions under the same context. This decomposition separates local decisions from the surrounding trajectory, allowing supervision to focus on action distinctions at each state rather than the successful path as a whole.

Reference-Branch Auditing. We audit reference branches for redundant actions using canonical tool-call signatures $\sigma(a)$ and indicators $m_t \in \{0, 1\}$. If an action repeats a recent call after a relevant call has succeeded, $m_t = 0$: the state remains as context, but a_t^+ is excluded from SFT and ranking. Otherwise, $m_t = 1$. This prevents locally redundant successful actions from being reinforced as positive supervision while preserving their surrounding interaction context.

4.2 EVIDENCE-GROUNDED COUNTERFACTUAL BOUNDARY EXPANSION

Execution Evidence Extraction. From the latest call $\bar{a}_t$ and observation O_t, BranchWise extracts verifiable facts, including success and recoverability by repeating the call. It constructs no branch without sufficient evidence. Recurring tool transitions across trajectories inform typical progression and termination patterns, but judgments remain grounded in s_t so local regularities do not become global rules.

Evidence-Grounded Counterfactual Branch Construction. Drawing on prior analyses of tool-use errors, state-dependent execution, and agent failures, together with a training-side empirical taxonomy study, we identify six recurring and locally verifiable failure directions around successful actions (Appendix A.2.8). They comprise two tool-contract violations (`ARGUMENT` and `STATE`) and four execution-transition errors (`DECISION`, `PROGRESS`, `RECOVERY`, and `STOP`), each checked against tool constraints, execution outcomes, or state changes.

For each state, BranchWise examines six failure directions across argument, state, and execution control that can be reliably certified from structured evidence, and instantiates only the counterfactual branches whose corresponding evidence conditions are satisfied. `ARGUMENT` (argument correctness) and `STATE` (state consistency) form the contract-constrained category; `DECISION`, `PROGRESS`, `RECOVERY`, and `STOP` capture premature execution, redundant post-success actions, ineffective retries, and failure to terminate, respectively, forming the execution-transition category. After deduplication, the resulting rule-derived counterfactual branch set $\mathcal{N}_t^{\text{rule}}$ expands each reference decision (s_t, a_t^+) into

$$(s_t, a_t^+, \mathcal{N}_t^{\text{rule}}), \tag{1}$$

yielding a data-side local decision boundary consisting of one reference branch and multiple verified counterfactual branches.

4.3 MODEL-TAILORED DECISION BOUNDARY COMPLETION

To complement task-grounded boundaries with model-specific errors, BranchWise probes the fixed initial policy π_{θ_0} for verifiable counterfactuals it actually generates.

Initial-Policy Error Probing. For each state, we sample R candidate next actions from π_{θ_0} without executing full rollouts, where $\widehat{a}_t^{(r)} \sim \pi_{\theta_0}(\cdot \mid p_t)$ for $r = 1, \ldots, R$. Since deviation from the reference branch does not necessarily imply an error, we retain only candidates that can be explicitly rejected by tool constraints or the Action Contract. Using the deterministic verifier $\mathcal{V}(s_t, c_t, a)$, we

define the verified model-derived counterfactual candidate set

$$\widetilde{\mathcal{N}}_t^{\text{model}} = \left\{ \widehat{a}_t^{(r)} : \mathcal{V}(s_t, c_t, \widehat{a}_t^{(r)}) \neq \emptyset \right\}. \tag{2}$$

After deduplication and cardinality control, at most M_{model} verified candidates are retained, yielding the model-derived counterfactual branch set $\mathcal{N}_t^{\text{model}}$. This probing is performed only once before training, keeping the candidate boundary space fixed throughout optimization.

Multi-Source Boundary Fusion. Counterfactuals from structural constraints, execution dynamics, and model behavior form complementary sets $\mathcal{N}_t^{\text{contract}}$, $\mathcal{N}_t^{\text{transition}}$, and $\mathcal{N}_t^{\text{model}}$. Under budget $K_{\max}$, selection prioritizes source diversity, then fills remaining slots in model–transition–contract order:

$$\mathcal{N}_t = \text{Select}_{\text{div}} \left(\mathcal{N}_t^{\text{contract}}, \mathcal{N}_t^{\text{transition}}, \mathcal{N}_t^{\text{model}}; K_{\max} \right), \qquad K_t = |\mathcal{N}_t| \leq K_{\max}. \tag{3}$$

Here, $\text{Select}_{\text{div}}$ implements this selection, yielding a compact boundary that combines task constraints and model-specific errors:

$$\left(a_t^+, \mathcal{N}_t \right), \qquad \mathcal{N}_t = \{ a_{t,1}^-, \ldots, a_{t,K_t}^- \}. \tag{4}$$

4.4 Decision-Fork Localization and Margin-Aware Boundary Optimization

Complete branches often share most of their tokens, while behavioral differences are concentrated in local decision regions. Moreover, the importance of different failure directions changes throughout training. BranchWise therefore first localizes the actual divergence between reference and counterfactual branches, and then dynamically focuses on local boundaries that remain insufficiently separated according to the model's current ability to distinguish them.

Decision-Fork Localization. For contract-constrained and model-derived branches, BranchWise removes the longest common token prefix and suffix between x_t^+ and $x_{t,k}^-$. Let $\ell_{t,k}$ and $r_{t,k}$ denote their lengths. The resulting contrast regions are

$$\mathcal{C}_{t,k}^+ = x_t^+[\ell_{t,k} : n_t^+ - r_{t,k}], \qquad \mathcal{C}_{t,k}^- = x_{t,k}^-[\ell_{t,k} : n_{t,k}^- - r_{t,k}]. \tag{5}$$

Their token masks $M_{t,k}^+$ and $M_{t,k}^-$ indicate membership in the corresponding contrast regions. For execution-transition branches, BranchWise instead retains at most L_{prefix} tokens from the effective action start to focus on the next execution direction. Let $b_{t,k}^+$ and $b_{t,k}^-$ denote the effective action starts:

$$\mathcal{C}_{t,k}^+ = x_t^+ \left[b_{t,k}^+ : \min(b_{t,k}^+ + L_{\text{prefix}}, n_t^+) \right], \qquad \mathcal{C}_{t,k}^- = x_{t,k}^- \left[b_{t,k}^- : \min(b_{t,k}^- + L_{\text{prefix}}, n_{t,k}^-) \right]. \tag{6}$$

Tool-call regions begin at the call position, while response regions begin at the first effective response token. Pairs whose verification evidence becomes invisible after truncation are excluded from ranking; valid counterfactual indices are denoted by $\mathcal{J}_t \subseteq \{1, \ldots, K_t\}$.

Evidence-Calibrated Reference-Branch Supervision. BranchWise assigns token-level weights $w_{t,j}$ to the reference branch using the Action Contract. When evidence confidence is low, the weights fall back toward the default SFT weight of 1, while group-wise normalization preserves the overall supervision scale. For minibatch $\mathcal{B}$ and valid-token mask $v_{t,j} \in \{0, 1\}$,

$$\mathcal{L}_{\text{SFT}} = -\frac{1}{|\mathcal{B}|} \sum_{t \in \mathcal{B}} \frac{\sum_j v_{t,j} w_{t,j} \log \pi_\theta(x_{t,j}^+ \mid p_t, x_{t,<j}^+)}{\max(1, \sum_j v_{t,j})}. \tag{7}$$

Audit gating sets $w_{t,j} = 0$ when $m_t = 0$; details are provided in the Appendix A.2.1.

Margin-Aware Local Ranking. BranchWise computes the average log-probabilities of reference and counterfactual branches only over the localized decision-fork regions:

$$q_{t,k}^+ = \frac{\sum_j M_{t,k,j}^+ \log \pi_\theta(x_{t,j}^+ \mid p_t, x_{t,<j}^+)}{\max(1, \sum_j M_{t,k,j}^+)}, \qquad q_{t,k}^- = \frac{\sum_j M_{t,k,j}^- \log \pi_\theta(x_{t,k,j}^- \mid p_t, x_{t,k,<j}^-)}{\max(1, \sum_j M_{t,k,j}^-)}. \tag{8}$$

The local margin is defined as

$$g_{t,k} = q^{+}_{t,k} - q^{-}_{t,k}, \tag{9}$$

where a smaller margin indicates that the counterfactual branch remains insufficiently separated from the reference branch. After evidence-visibility and exposure control, let $\mathcal{I}_t \subseteq \mathcal{J}_t$ denote the active counterfactual indices. Given target margin μ and temperature $T > 0$, BranchWise applies listwise ranking over the active set with loss $\ell_{\text{rank}}(t) = \log\left[1 + \sum_{k\in\mathcal{I}_t} \exp\left((\mu - g_{t,k})/T\right)\right] / \log(1 + |\mathcal{I}_t|)$. Smaller-margin branches therefore receive greater optimization pressure. For $\mathcal{B}_{\text{rank}} = \{t \in \mathcal{B} : |\mathcal{I}_t| > 0\}$, $\mathcal{L}_{\text{rank}} = \frac{1}{|\mathcal{B}_{\text{rank}}|} \sum_{t\in\mathcal{B}_{\text{rank}}} \ell_{\text{rank}}(t)$. The final objective combines weighted SFT and ranking:

$$\mathcal{L} = \mathcal{L}_{\text{SFT}} + \lambda \mathcal{L}_{\text{rank}}. \tag{10}$$

Because initial-policy probing is performed only once before training, BranchWise dynamically reallocates optimization pressure within a fixed model-tailored boundary space according to the current margin, without additional training-time rollouts.

5 Experiments

Training Data. We sample 9,000 ToolMind (Yang et al., 2025b) trajectories proportionally across sources and add 1,200 execution-verifiable state trajectories from τ-bench (Yao et al., 2025) for state-changing interactions. After instance-level decontamination and replacement, the final set contains 10,200 trajectories with no overlap with the τ^2-Bench evaluation data. Details are provided in Appendix A.3.1.

Evaluation. We evaluate BranchWise on four tool-use benchmarks: τ^2-Bench (Barres et al., 2025), BFCL V3 (Patil et al., 2025), ToolHop (Ye et al., 2025), and ToolTalk-Hard (Farn & Shin, 2023), with deterministic decoding (temperature 0) and matched reasoning and tool-use settings across methods within each benchmark. Results average repeated checkpoint evaluations across independent training runs with different seeds. Detailed evaluation protocols and aggregation procedures are provided in Appendix A.6.

Training Setup. We primarily use Qwen3-8B (Yang et al., 2025a) and additionally train and evaluate Qwen3-4B. All methods start from the same pool of 10,200 candidate trajectories. Full-corpus methods are trained for one epoch with LoRA (Hu et al., 2022), while data-selection baselines use a matched final fine-tuning budget. The global batch size is 32, maximum sequence length 8,192, and learning rate 2×10^{-4}. Additional training and implementation details are provided in Appendix A.4.

Baselines. We compare against the original Qwen3, Vanilla SFT, and six strong baselines: T-SHIRT (Fu et al., 2025) and DaaR (Ling et al., 2025) as representative methods for sample-level data selection, Token Cleaning (Pang et al., 2025) for token-level supervision filtering, LESS (Xia et al., 2024) and GIST (Min et al., 2026) for target-aware data selection, and Beyond Log Likelihood (Li et al., 2025a) for SFT objective redesign. Together, they cover the main alternative post-training strategies and test whether BranchWise's gains can be explained by data selection, supervision filtering, or objective redesign.

5.1 Main Results

Table 1 compares BranchWise with the base model, Vanilla SFT, and six post-training baselines on Qwen3-8B, and with the base model and Vanilla SFT on Qwen3-4B to assess consistency across model scales.

BranchWise yields stronger overall tool-use gains. On Qwen3-8B, BranchWise improves over Vanilla SFT by 10.48 percentage points on τ^2-Bench, 1.23 on BFCL V3, 12.16/14.98 on ToolHop Mandatory/Free, and 4.00/3.40 on ToolTalk-Hard Success/Soft. It outperforms Vanilla SFT across all four benchmarks and achieves the strongest overall results among the evaluated methods, suggesting more effective transfer of decision supervision from the same successful trajectories. The persistence of these gains across different forms of tool interaction suggests that BranchWise more effectively converts the decision supervision latent in successful trajectories into transferable tool-use capabilities.

Table 1: Results across four tool-use benchmarks and two model scales (%). Bold marks the best result per metric and backbone.

(a) τ^2-Bench and BFCL V3

		τ^2-Bench(%)				BFCL V3(%)			
Backbone	Method	Overall	Airline	Retail	Telecom	Overall	Non-Live	Live	Multi-Turn
Qwen3-8B	Base	32.73	12.00	42.23	33.33	63.59	**86.48**	78.41	25.88
	Vanilla SFT	29.77	13.50	35.31	31.36	64.26	85.92	78.37	28.50
	T-SHIRT	29.41	12.00	34.65	31.80	63.10	86.17	78.63	24.50
	Token Cleaning	33.45	37.00	33.55	32.02	62.78	84.52	77.70	26.12
	GIST	29.68	**39.50**	33.33	21.71	64.82	85.15	78.32	31.00
	DaaR	34.26	17.50	45.18	30.70	64.32	85.88	77.83	29.25
	Beyond Log Likelihood	37.05	24.00	45.61	32.21	63.40	86.17	78.72	25.32
	LESS	27.25	6.50	41.50	27.75	63.62	85.10	78.50	27.25
	BranchWise	**40.25**	25.50	**52.74**	**34.21**	**65.49**	85.10	**78.99**	**32.38**
Qwen3-4B	Base	28.54	18.67	31.00	**30.41**	61.34	84.97	76.06	23.00
	Vanilla SFT	22.90	**22.67**	33.04	12.87	61.88	**86.12**	**79.52**	20.00
	BranchWise	**29.38**	15.33	**43.27**	21.64	**62.35**	85.53	76.63	**24.87**

(b) ToolHop and ToolTalk-Hard

		ToolHop(%)			ToolTalk-Hard(%)	
Backbone	Method	Direct	Mandatory	Free	Success	Soft
Qwen3-8B	Base	**10.55**	33.27	31.31	20.00	49.59
	Vanilla SFT	7.94	27.94	26.23	24.00	54.31
	T-SHIRT	7.84	30.45	32.36	20.00	50.27
	Token Cleaning	5.93	37.12	25.83	18.00	47.14
	GIST	6.03	24.32	24.22	24.00	55.17
	DaaR	7.64	35.86	27.57	20.00	49.23
	Beyond Log Likelihood	4.42	28.94	26.13	22.00	53.12
	LESS	10.05	26.33	25.70	16.00	35.70
	BranchWise	8.94	**40.10**	**41.21**	**28.00**	**57.71**
Qwen3-4B	Base	5.63	16.88	15.88	8.00	36.85
	Vanilla SFT	**7.64**	9.55	12.46	12.00	42.14
	BranchWise	5.63	**35.38**	**32.06**	**18.00**	**47.37**

BranchWise makes downstream transfer more reliable. Vanilla SFT and several baselines improve some tasks but fall below the base model on others: SFT loses 2.96 percentage points on τ^2-Bench and 5.33/5.08 on ToolHop Mandatory/Free. This suggests that imitation, selection, or objective modification centered on observed successful paths alone does not necessarily yield stable cross-task gains. BranchWise instead yields positive overall gains across all four benchmarks without benchmark-specific downstream supervision. These results suggest that reconstructing local boundaries transfers general tool-use supervision more reliably than approaches centered on observed successful paths alone.

BranchWise gains more in sustained tool interactions. Compared with Vanilla SFT, BranchWise improves Qwen3-8B by 3.88 points on BFCL Multi-Turn and by 12.16/14.98 points on ToolHop Mandatory/Free. This concentration of gains suggests that BranchWise does not merely strengthen successful trajectories uniformly, but more effectively improves the local decisions that determine subsequent interaction behavior, consistent with its explicit construction and optimization of local decision boundaries. The larger gains on settings requiring repeated tool use and feedback-dependent action selection further suggest that these local distinctions become increasingly important as interactions unfold over multiple decision steps.

BranchWise exhibits consistent effects across model scales. Similar overall gains on Qwen3-4B and Qwen3-8B suggest that BranchWise's benefits extend across model scales.

5.2 Ablation Studies

We ablate Qwen3-8B under matched data, training budgets, and evaluation settings. *w/o Counterfactual Ranking* removes reference–counterfactual ranking; *w/o Evidence-Grounded CF* and *w/o*

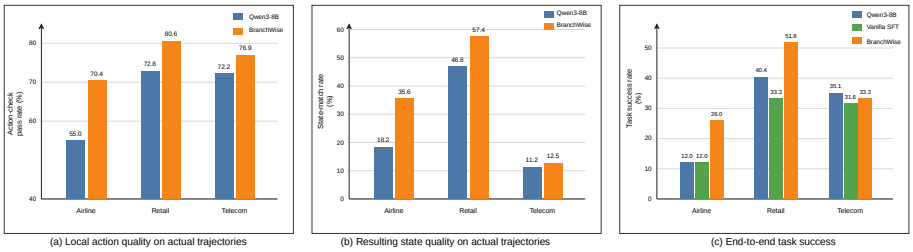
Table 2: Ablation study of BranchWise on Qwen3-8B. All values are percentages. Bold indicates the best result in each column.

	τ^2-Bench	BFCL V3	ToolHop			ToolTalk-Hard	
Method	Overall	Overall	Direct	Mandatory	Free	Success	Soft
BranchWise	**40.25**	**65.49**	**8.94**	**40.10**	**41.21**	**28.00**	**57.71**
w/o Counterfactual Ranking	35.61	64.66	6.13	31.46	31.56	20.00	50.79
w/o Evidence-Grounded CF	32.37	63.34	8.14	34.00	32.16	18.00	48.57
w/o Model-Derived CF	37.41	63.27	6.43	28.24	26.53	24.00	54.71
w/o Fork Localization	34.89	63.99	8.42	30.55	32.67	16.00	46.53
w/o Margin-Aware Ranking	35.25	63.57	8.64	31.05	33.07	20.00	49.99

Model-Derived CF remove branches from execution evidence and initial-policy probing, respectively; *w/o Fork Localization* compares the same pairs over complete branches; and *w/o Margin-Aware Ranking* removes low-margin prioritization. Every ablation lowers all reported metrics (Table 2), supporting complementary counterfactual construction, localized comparisons, and optimization of insufficiently separated boundaries.

6 Analysis

We analyze local decision quality, supervision locality, feedback use, and boundary learning.



(a) Local action quality on actual trajectories (b) Resulting state quality on actual trajectories (c) End-to-end task success

Figure 4: Action quality, state matching, and task success on τ^2-Bench.

BranchWise improves local action selection and state-transition quality. Using τ^2-Bench's intermediate evaluation signals (Section 5), we measure action quality by the action-check pass rate—the fraction of task-specified requirements met by executed calls—and state-transition correctness by the state-match rate—the fraction of tasks ending in the target database state. BranchWise improves both over Qwen3-8B across all three domains (Fig. 4(a)–(b)), alongside higher task success than Qwen3-8B and Vanilla SFT (Fig. 4(c)). These results link local improvements to task gains and support state-conditioned decision decomposition and evidence-grounded counterfactual expansion as informative supervision from successful trajectories.

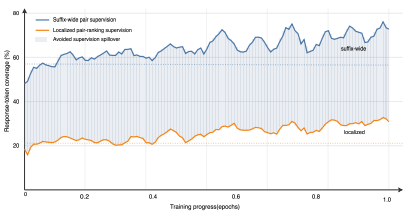
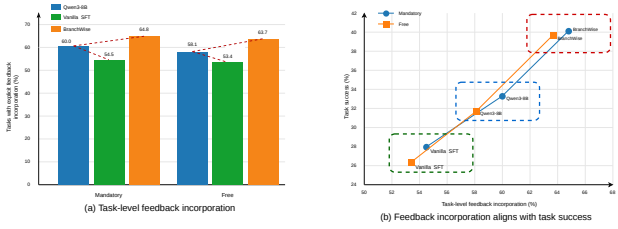


Figure 5: Ranking supervision coverage.

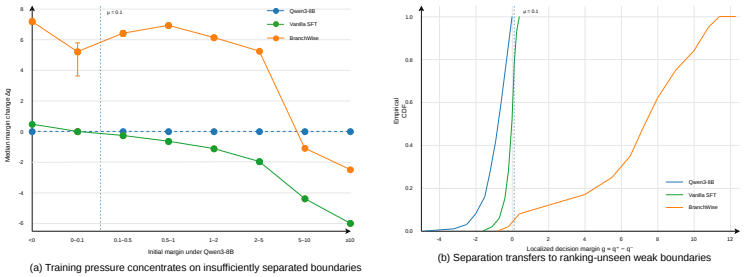


(a) Task-level feedback incorporation (b) Feedback incorporation aligns with task success

Figure 6: Feedback incorporation and task success.

Decision-Fork Localization concentrates ranking supervision near decision forks. Across all reference–counterfactual ranking exposures in the main BranchWise run, we compare the actual local mask with a hypothetical mask extending from the intervention boundary to the response end. With one training epoch, Fig. 5 plots the fraction of response tokens covered across chosen and rejected branches against epoch completion, using a trailing average over approximately 0.04 epoch. Suffix-wide coverage would be 57.01%, versus 21.10% locally, avoiding 35.91 percentage points of potential spillover. The persistent gap indicates that ranking supervision remains concentrated around intervention regions. This suggests that Decision-Fork Localization consistently suppresses supervision on shared continuation tokens throughout training, rather than merely shortening a subset of individual comparisons.

BranchWise strengthens feedback-conditioned decision making. We analyze the main ToolHop evaluation trajectories of Qwen3-8B, Vanilla SFT, and BranchWise, using tool-response-to-next-call transitions to quantify feedback use. Among tasks with usable new feedback, the feedback incorporation rate measures the fraction with at least one subsequent tool call explicitly using it. Relative to Qwen3-8B, BranchWise gains 4.8/5.6 percentage points on Mandatory/Free, while Vanilla SFT loses 5.5/4.7 (Fig. 6(a)). BranchWise also improves task success in both settings, whereas SFT reduces both measures (Fig. 6(b)). These results support evidence-grounded counterfactual expansion and model-specific error completion in adapting subsequent actions to feedback.



(a) Training pressure concentrates on insufficiently separated boundaries

(b) Separation transfers to ranking-unseen weak boundaries

Figure 7: Local boundary separation and generalization.

BranchWise learns clearer, transferable local decision boundaries. We score all verified training-side reference–counterfactual pairs offline with Qwen3-8B, Vanilla SFT, and BranchWise. The localized margin $g = q^+ - q^-$ compares reference and counterfactual mean log-probabilities within decision-fork regions; the small positive target $\mu = 0.1$ favors reference branches without excessive pressure on already separated boundaries. Many pairs have small or negative initial margins under Qwen3-8B. SFT enlarges them inconsistently or even reduces them, whereas BranchWise enlarges low-to-moderate margins (Fig. 7(a)), indicating nontrivial negatives that path imitation alone inadequately separates. Among boundaries with base-model $g < \mu$ and no direct ranking supervision, 93.1% reach $g \geq \mu$ under BranchWise versus 21.9% under SFT (Fig. 7(b)). Together with the ablations, these findings support Decision-Fork Localization and Margin-Aware Local Ranking in learning transferable local branch distinctions.

7 Conclusion

Successful trajectories reveal realized paths but leave the local policy structure needed for generalization implicit. **BranchWise** recovers this structure as model-tailored decision boundaries by fusing evidence-grounded and model-derived counterfactuals. Localized comparisons, evidence-calibrated reference supervision, and margin-aware ranking then target behavioral distinctions unresolved by the current policy. Across benchmarks and model scales, BranchWise yields more stable gains than direct imitation; ablations and mechanistic analyses support boundary reconstruction and boundary-aware optimization. These results position successful trajectories as partial observations from which transferable local policy structure can be recovered.

AI USE STATEMENT

In this work, we used generative AI tools solely for language editing and proofreading. Specifically, they were used to identify punctuation, spelling, and grammatical errors and to improve sentence-level fluency, clarity, and coherence. The research ideas, problem formulation, methodology, experimental design, implementation, experiments, results, analyses, and scientific conclusions were developed and completed by the authors. The initial manuscript draft was also written by the authors, and generative AI was not used to originate or determine the technical content, experimental findings, or scientific claims of the paper. All AI-assisted language edits were manually reviewed by the authors to ensure that they preserved the intended technical meaning. We take full responsibility for the final content of the paper, including all text and claims that received AI-assisted language editing.

ETHICS STATEMENT

This work focuses on methodological research for post-training tool-use agents and does not involve human subjects, personally identifiable or sensitive data, or user-facing deployment. The study does not introduce datasets containing private information, nor does it involve interventions, data collection from individuals, or applications that raise specific concerns regarding discrimination, fairness, or physical harm. All datasets, models, and benchmarks are used in accordance with their intended research purposes and applicable licenses. We are not aware of any additional ethical concerns specific to this work.

REFERENCES

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-bench: Evaluating conversational agents in a dual-control environment, 2025. URL https://arxiv.org/abs/2506.07982.

Akash Bonagiri, Devang Borkar, Gerard Janno Anderias, Setareh Rafatirad, and Houman Homayoun. Causalflow: Causal attribution and counterfactual repair for llm agent failures, 2026. URL https://arxiv.org/abs/2605.25338.

Baian Chen, Chang Shu, Ehsan Shareghi, Nigel Collier, Karthik Narasimhan, and Shunyu Yao. Fireact: Toward language agent fine-tuning, 2023. URL https://arxiv.org/abs/2310.05915.

Sijia Chen, Yibo Wang, Yi-Feng Wu, Qing-Guo Chen, Zhao Xu, Weihua Luo, Kaifu Zhang, and Lijun Zhang. Advancing tool-augmented large language models: Integrating insights from errors in inference trees. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 106555–106581. Curran Associates, Inc., 2024. doi: 10.52202/079017-3382. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/c0f7ee1901fef1da4dae2b88dfd43195-Paper-Conference.pdf.

Wenxiang Chen, Wei He, Zhiheng Xi, Honglin Guo, Boyang Hong, Jiazheng Zhang, Nijun Li, Tao Gui, Yun Li, Qi Zhang, and Xuanjing Huang. Better process supervision with bi-directional rewarding signals. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 14471–14485, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.747. URL https://aclanthology.org/2025.findings-acl.747/.

Yixiong Chen and Alan Yuille. Agentic-dpo: From imitation to agentic policy optimization on expert trajectories. *arXiv preprint arXiv:2607.10601*, 2026.

Mingyue Cheng, Shuo Yu, Daoyu Wang, Qingchuan Li, Xiaoyu Tao, Jie Ouyang, Yucong Luo, Yitong Zhou, Qi Liu, and Enhong Chen. Agent-r1: A unified and modular framework for agentic reinforcement learning, 2026. URL https://arxiv.org/abs/2511.14460.

Yuchen Deng, Shichen Fan, Naibo Wang, Xinkui Zhao, and See-Kiong Ng. AgentPro: Enhancing LLM agents with automated process supervision. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9981–10006, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.506. URL https://aclanthology.org/2025.emnlp-main.506/.

Nicholas Farn and Richard Shin. Tooltalk: Evaluating tool-usage in a conversational setting, 2023. URL https://arxiv.org/abs/2311.10775.

Jiazhan Feng, Shijue Huang, Xingwei Qu, Ge Zhang, Yujia Qin, Baoquan Zhong, Chengquan Jiang, Jinxin Chi, and Wanjun Zhong. Retool: Reinforcement learning for strategic tool use in llms. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 37909–37926, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/4038c9208dfc22644c60ad39c24e5c53-Paper-Conference.pdf.

Yanjun Fu, Faisal Hamman, and Sanghamitra Dutta. T-shirt: Token-selective hierarchical data selection for instruction tuning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 113932–113958. Curran Associates, Inc., 2025. doi: 10.52202/085713-3800. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/a59ff5f7384176b5d14a5ded77c4aa4f-Paper-Conference.pdf.

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

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. Process reward models that think, 2025. URL https://arxiv.org/abs/2504.16828.

Gaotang Li, Ruizhong Qiu, Xiusi Chen, Heng Ji, and Hanghang Tong. Beyond log likelihood: Probability-based objectives for supervised fine-tuning across the model capability continuum. *arXiv preprint arXiv:2510.00526*, 2025a.

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 Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3102–3116, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.187. URL https://aclanthology.org/2023.emnlp-main.187/.

Pengxiang Li, Zhi Gao, Bofei Zhang, Yapeng Mi, Xiaojian (Shawn) Ma, Chenrui Shi, Tao Yuan, Yuwei Wu, Yunde Jia, Song-Chun Zhu, and Qing Li. Iterative tool usage exploration for multimodal agents via step-wise preference tuning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 59496–59528. Curran Associates, Inc., 2025b. doi: 10.52202/085713-1991. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/55d16334943f8728073e17139e5baa3d-Paper-Conference.pdf.

Xuefeng Li, Haoyang Zou, and Pengfei Liu. Torl: Scaling tool-integrated rl, 2025c. URL https://arxiv.org/abs/2503.23383.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Zhenqing Ling, Daoyuan Chen, Liuyi Yao, Qianli Shen, Yaliang Li, and Ying Shen. Diversity as a reward: Fine-tuning llms on a mixture of domain-undetermined data. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 109260–109306. Curran Associates, Inc., 2025. doi: 10.52202/085713-3645. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/9d7a6dc57a3216dd1a2e80a8c4a55ed6-Paper-Conference.pdf.

Weiting Liu, Jieyi Bi, Wanqi Zhou, Jianfeng Feng, Yining Ma, Ai Han, and Wenlian Lu. Toolanchor: Anchoring counterfactual context to boost agentic tool-use capability, 2026. URL https://arxiv.org/abs/2607.14145.

Weiwen Liu, Xu Huang, Xingshan Zeng, xinlong hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong WANG, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Wang Xinzhi, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of llm function calling. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 41359–41381, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/663865ea167425c6c562cb0b6bcf76c7-Paper-Conference.pdf.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 52989–53046, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/e9df36b21ff4ee211a8b71ee8b7e9f57-Paper-Conference.pdf.

Jiarui Lu, Thomas Holleis, Yizhe Zhang, Bernhard Aumayer, Feng Nan, Haoping Bai, Shuang Ma, Shen Ma, Mengyu Li, Guoli Yin, Zirui Wang, and Ruoming Pang. ToolSandbox: A stateful, conversational, interactive evaluation benchmark for LLM tool use capabilities. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1160–1183, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.65. URL https://aclanthology.org/2025.findings-naacl.65/.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 46534–46594. Curran Associates, Inc., 2023. doi: 10.52202/075280-2019. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/91edff07232fb1b55a505a9e9f6c0ff3-Paper-Conference.pdf.

Guanghui Min, Tianhao Huang, Ke Wan, and Chen Chen. Gist: Targeted data selection for instruction tuning via coupled optimization geometry. *arXiv preprint arXiv:2602.18584*, 2026.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022. URL https://arxiv.org/abs/2112.09332.

Jinlong Pang, Na Di, Zhaowei Zhu, Jiaheng Wei, Hao Cheng, Chen Qian, and Yang Liu. Token cleaning: Fine-grained data selection for LLM supervised fine-tuning. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47837–47858. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/pang25a.html.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive apis. 2024.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. Toolrl: Reward is all tool learning needs, 2025. URL https://arxiv.org/abs/2504.13958.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 9695–9717, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/28e50ee5b72e90b50e7196fde8ea260e-Paper-Conference.pdf.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 68539–68551. Curran Associates, Inc., 2023. doi: 10.52202/075280-2997. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/d842425e4bf79ba039352da0f658a906-Paper-Conference.pdf.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/1b44b878bb782e6954cd888628510e90-Paper-Conference.pdf.

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

Yifan Song, Da Yin, Xiang Yue, Jie Huang, Sujian Li, and Bill Yuchen Lin. Trial and error: Exploration-based trajectory optimization of LLM agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7584–7600, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.409. URL https://aclanthology.org/2024.acl-long.409/.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.510. URL https://aclanthology.org/2024.acl-long.510/.

Zihan Wang, Kangrui Wang, Qineng Wang, Pingyue Zhang, Linjie Li, Zhengyuan Yang, Xing Jin, Kefan Yu, Minh Nhat Nguyen, Licheng Liu, Eli Gottlieb, Yiping Lu, Kyunghyun Cho, Jiajun Wu, Li Fei-Fei, Lijuan Wang, Yejin Choi, and Manling Li. Ragen: Understanding self-evolution in llm agents via multi-turn reinforcement learning, 2025. URL https://arxiv.org/abs/2504.20073.

Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. LESS: Selecting influential data for targeted instruction tuning. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54104–54132. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/xia24c.html.

Weimin Xiong, Yifan Song, Xiutian Zhao, Wenhao Wu, Xun Wang, Ke Wang, Cheng Li, Wei Peng, and Sujian Li. Watch every step! LLM agent learning via iterative step-level process refinement. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 1556–1572, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.93. URL https://aclanthology.org/2024.emnlp-main.93/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

Chen Yang, Ran Le, Yun Xing, Zhenwei An, Zongchao Chen, Wayne Xin Zhao, Yang Song, and Tao Zhang. Toolmind technical report: A large-scale, reasoning-enhanced tool-use dataset, 2025b. URL https://arxiv.org/abs/2511.15718.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023. URL https://arxiv.org/abs/2210.03629.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 9965–10017, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b126cc38b8638e07bef37e7b2bb72bf-Paper-Conference.pdf.

Junjie Ye, Zhengyin Du, Xuesong Yao, Weijian Lin, Yufei Xu, Zehui Chen, Zaiyuan Wang, Sining Zhu, Zhiheng Xi, Siyu Yuan, Tao Gui, Qi Zhang, Xuanjing Huang, and Jiecao Chen. ToolHop: A query-driven benchmark for evaluating large language models in multi-hop tool use. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2995–3021, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.150. URL https://aclanthology.org/2025.acl-long.150/.

Junkeun Yi, Damon Mosk-Aoyama, Baihe Huang, Ritu Gala, Charles Wang, Sugam Dipak Devare, Khushi Bhardwaj, Abhibha Gupta, Oleksii Kuchaiev, Jiantao Jiao, Jian Zhang, and Venkat Srinivasan. Pivotrl: High accuracy agentic post-training at low compute cost, 2026. URL https://arxiv.org/abs/2603.21383.

Aohan Zeng, Mingdao Liu, Rui Lu, Bowen Wang, Xiao Liu, Yuxiao Dong, and Jie Tang. AgentTuning: Enabling generalized agent abilities for LLMs. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL*

2024, pp. 3053–3077, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.181. URL https://aclanthology.org/2024.findings-acl.181/.

Yirong Zeng, Xiao Ding, Yutai Hou, Yuxian Wang, Li Du, Juyi Dai, Qiuyang Ding, Duyu Tang, Dandan Tu, Weiwen Liu, Bing Qin, and Ting Liu. Tool zero: Training tool-augmented LLMs via pure RL from scratch. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 9135–9147, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.485. URL https://aclanthology.org/2025.findings-emnlp.485/.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 64735–64772. Curran Associates, Inc., 2024. doi: 10.52202/079017-2066. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/76ec4dc30e9faaf0e4b6093eaa377218-Paper-Conference.pdf.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. Webarena: A realistic web environment for building autonomous agents. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/4410c0711e9154a7a2d26f9b3816d1ef-Paper-Conference.pdf.

A Appendix

A.1 Limitations

BranchWise relies on tool constraints, execution outcomes, and state transitions that can be reliably verified from training trajectories to recover local counterfactual boundaries. Its supervision construction is therefore limited by environment verifiability. When tool feedback lacks structured execution evidence, task states cannot be reliably observed, or the available evidence is insufficient to determine whether alternative actions are incorrect, BranchWise conservatively avoids constructing the corresponding counterfactual branches. This may reduce the coverage of recoverable local decision boundaries. In addition, model-derived counterfactuals are obtained through one-time probing of the fixed initial policy before training. They therefore primarily reflect the error patterns of the initial model rather than explicitly expanding to new failure directions that may emerge during optimization. Finally, our experiments focus on Qwen3-4B/8B and four tool-use benchmarks; the applicability of BranchWise to larger models, other agent environments, and weakly verifiable tasks remains to be studied.

A.2 Additional Method and Supervision Construction Details

This section provides implementation details for the supervision-construction and optimization procedures introduced in Section 4. We focus on the parts omitted from the main paper for brevity, including Action Contract verification, executable counterfactual construction, initial-policy probing, multi-source boundary fusion, reference-branch weighting, decision-fork localization, and ranking-exposure control.

A.2.1 Action Contracts and Reference-Branch Auditing

Action Contract verification. For each assistant decision turn, BranchWise constructs the training-side Action Contract

$$c_t = (d_t, \kappa_t, \xi_t, \rho_t)$$

defined in Section 3. Each contract is assigned one of three verification states: `CONFIRMED`, `REJECTED`, or `UNKNOWN`. Verification jointly considers the expected decision type, tool capabilities

and schema constraints, argument provenance, execution outcomes, and observed state transitions when available.

We determine the confidence threshold using a held-out Contract verification set rather than downstream benchmark performance. The set contains 2,000 Contracts with independently verified validity labels, obtained primarily through a deterministic verifier and supplemented by manual inspection of a small subset. For each candidate threshold τ, we consider only Contracts marked as `CONFIRMED` and satisfying

$$\rho_t \geq \tau,$$

and compute their empirical validity rate together with its 95% Wilson lower confidence bound.

We predefine the target reliability as 0.9 and select the smallest threshold satisfying $\mathrm{LCB}_{0.95}^{\mathrm{Wilson}}(\tau) \geq 0.9$. This criterion retains as much supervision as possible while maintaining the prescribed validity level. The calibration yields $\tau = 0.8$, which is fixed across all experiments and is not tuned on downstream benchmark performance. Accordingly, a Contract is eligible for high-confidence counterfactual construction only when it is marked as `CONFIRMED` and satisfies $\rho_t \geq 0.8$.

Reference decisions marked as `REJECTED` are excluded from supervision, whereas `UNKNOWN` decisions may remain as ordinary reference supervision but are not used for counterfactual construction that requires confirmed evidence.

Reference-branch auditing. Successful trajectories can still contain redundant intermediate actions. BranchWise therefore applies a conservative execution-based audit to reference tool calls. Let $\sigma(a)$ denote the canonical signature of a tool call. A reference action is masked only when the local interaction history provides explicit evidence that it repeats an already successful call without making progress.

Concretely, the implementation examines recent tool-call signatures and identifies short repeated cycles following successful tool execution. When such a cycle is verified, the corresponding reference completion receives zero SFT weight and its counterfactual pairs are excluded from ranking, while the interaction state itself remains in the trajectory context.

A.2.2 Evidence-Grounded Counterfactual Construction

BranchWise constructs a counterfactual branch only when its incorrectness can be explicitly verified from tool constraints, execution evidence, or observed state transitions. The six failure directions introduced in Section 4.2 are instantiated as summarized in Table 3.

Contract-constrained counterfactuals. `ARGUMENT` and `STATE` form the contract-constrained category. For `ARGUMENT`, BranchWise modifies a locally verifiable argument of the reference tool invocation while preserving the remaining action structure. For `STATE`, BranchWise requires execution-backed state-transition evidence and constructs an alternative that conflicts with the verified transition.

Execution-transition counterfactuals. `DECISION`, `PROGRESS`, `RECOVERY`, and `STOP` capture failures in deciding whether and how execution should continue. These branches are constructed from the relationship between the preceding tool outcome and the current reference action. When the execution outcome is ambiguous and the corresponding failure direction cannot be reliably certified, no such counterfactual is constructed.

A.2.3 Initial-Policy Error Probing

Evidence-grounded counterfactuals represent task-relevant failure directions, but different models need not make the same mistakes. BranchWise therefore supplements them with erroneous branches actually generated by the fixed initial policy π_{θ_0}.

For each selected state, we sample four candidate next actions from the frozen initial policy:

$$\widehat{a}_t^{(r)} \sim \pi_{\theta_0}(\cdot \mid p_t), \qquad r = 1, \ldots, 4.$$

Sampling uses

$$\text{temperature} = 0.8, \qquad \text{top-}p = 0.9,$$

Type	Failure direction	Required evidence	Counterfactual construction
ARGUMENT	Argument-contract violation	A confirmed tool call together with verifiable schema or argument-provenance constraints	Modify a bound argument so that it violates a locally checkable constraint, such as an enum, representation, or identifier constraint.
STATE	State-inconsistent behavior	A verified state mutation associated with the reference action	Construct an alternative inconsistent with the verified transition, such as unnecessarily repeating a non-idempotent mutation or claiming completion without performing the required mutation.
DECISION	Premature execution	Evidence that the conditions required for execution are not yet satisfied	Replace the reference non-execution decision with an unsupported or incomplete tool invocation.
PROGRESS	Failure to advance after success	The preceding tool call succeeds and the reference proceeds to a different next action	Repeat the preceding successful call instead of following the reference progression.
RECOVERY	Ineffective retry after failure	The preceding execution produces a verified non-retryable error and the reference changes its action	Repeat the failed call instead of taking the reference recovery action.
STOP	Failure to terminate	The reference terminates tool execution after a locally verified successful or non-retryable outcome	Continue with an unnecessary tool call instead of producing the reference no-tool response.

Table 3: Executable definitions of the six evidence-grounded counterfactual directions. A negative branch is instantiated only when its incorrectness can be certified from locally available training-side evidence.

with at most 256 generated tokens.

Deviation from the demonstration alone is not treated as an error. Each sampled action is replayed through the deterministic verifier, and only candidates that violate a verifiable Action Contract, tool constraint, or execution condition are retained.

Initial-policy probing is performed only once before training. The resulting model-derived candidate boundary set therefore remains fixed throughout optimization, and BranchWise introduces no additional online rollout during training.

A.2.4 Multi-Source Boundary Fusion

Verified counterfactual branches at state s_t are grouped into three complementary sources: contract-constrained branches, including ARGUMENT and STATE; execution-transition branches, including DECISION, PROGRESS, RECOVERY, and STOP; and model-derived branches, denoted by MODEL_ERROR. Only candidates satisfying the corresponding evidence requirements and verification criteria enter the subsequent fusion procedure.

BranchWise sets the maximum number of counterfactual branches per state to

$$K_{\max} = 3,$$

matching the three complementary counterfactual sources. When contract-constrained, execution-transition, and model-derived branches are all available, diversity-first selection can prioritize representative branches from different sources while bounding the size of each local boundary and the associated ranking cost. This branch budget is fixed across all experiments.

Under this budget, BranchWise first prioritizes source diversity and then fills the remaining slots according to the ordering described in Section 4.3. This yields compact local decision boundaries while preventing multiple highly similar negatives from a single source from occupying the entire branch budget.

A.2.5 EVIDENCE-CALIBRATED REFERENCE-BRANCH SUPERVISION

Section 4.4 defines the weighted reference-branch objective. Here we further specify how confidence enters the token weights.

For an active supervision component, let $\widetilde{w}$ denote its evidence-derived and normalized priority weight, and let ρ_t denote the confidence of the corresponding Action Contract. The effective weight is

$$w = 1 + \rho_t(\widetilde{w} - 1).$$

As evidence confidence decreases, the effective weight therefore smoothly falls back toward the standard SFT weight of 1, avoiding overly strong supervision from uncertain evidence.

Raw priorities are normalized within comparable source and decision-type groups before confidence calibration. The weighting mechanism therefore changes the relative emphasis among local decisions without substantially altering the overall supervision scale. Reference decisions excluded by the audit in Appendix A.2.1 receive zero SFT weight.

A.2.6 DECISION-FORK LOCALIZATION

For contract-constrained and model-derived reference–counterfactual pairs, BranchWise removes their longest identical token prefix and suffix. Ranking is then computed only over the remaining differing regions, as defined in Eq. equation 5.

For execution-transition counterfactuals, the behavioral distinction is concentrated in the immediate next action rather than the full continuation. We therefore determine L_{prefix} from the distribution of decisive action-prefix lengths in the training data.

For each verified execution-transition pair, starting from the effective action start, we define the shortest prefix sufficient to determine the next-action difference between the reference and counterfactual branches as its *decisive action prefix*. For a tool-call branch, this region covers at least the target tool identity and the key argument span responsible for the behavioral divergence. For a textual branch, it covers the first complete action region sufficient to identify a response, clarification, or termination decision. Let d_i denote the length of this prefix for the i-th verified pair.

For N verified execution-transition pairs, we choose the smallest prefix length that fully covers at least 95% of them:

$$L_{\text{prefix}} = \min\left\{ L : \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[d_i \le L] \ge 0.95 \right\}.$$

This criterion yields

$$L_{\text{prefix}} = 24.$$

The value is determined before downstream evaluation, fixed across all experiments, and not tuned on benchmark performance.

Accordingly, for

$$\{\texttt{DECISION}, \texttt{PROGRESS}, \texttt{RECOVERY}, \texttt{STOP}\},$$

BranchWise retains at most 24 tokens from the effective action start.

For tool-call branches, the effective region begins at the tool-call region; for textual responses, it begins at the first effective response token. If sequence truncation removes the local evidence required to interpret a counterfactual, the corresponding pair is excluded from ranking. This avoids comparing branches after the evidence supporting their distinction is no longer visible to the model.

A.2.7 MARGIN-AWARE RANKING AND EXPOSURE CONTROL

The listwise ranking objective in Section 4.4 uses

$$\lambda = 0.05, \qquad \mu = 0.1, \qquad T = 1.0.$$

Each local decision state contributes at most three rejected branches.

To prevent repeatedly sampled states from dominating optimization, each unique reference–counterfactual pair can participate in ranking at most twice during training.

Data source	# trajectories
ToolMind	9,000
Execution-verifiable state trajectories	1,200
Total	10,200

Table 4: Composition of the post-training corpus.

A.2.8 Empirical Derivation and Validation of the Failure Taxonomy

We derive the six evidence-grounded failure directions through a training-side empirical study conducted before constructing the final post-training supervision. The study uses development trajectories drawn from the same training-side source pools as our post-training data but excluded from the final 10,200-trajectory corpus. Taxonomy discovery, taxonomy validation, and final post-training data are mutually disjoint at the trajectory level, and no downstream benchmark test instances are inspected during this process.

For taxonomy discovery, we sample successful decision states from the development trajectories and use the frozen Qwen3-8B initial policy π_{θ_0} to generate alternative next actions. Candidate actions are examined using the Action Contract, tool constraints, execution outcomes, and observed state transitions, and only errors whose incorrectness can be locally verified are retained. We then perform open coding on a subset of these verified errors without providing predefined failure labels. For each error, the annotation considers the interaction state, reference action, sampled alternative, relevant tool constraints, execution evidence, and state transition, and records the local reason why the sampled action is incorrect. Recurring error patterns are subsequently consolidated into six categories: `ARGUMENT`, `STATE`, `DECISION`, `PROGRESS`, `RECOVERY`, and `STOP`.

After fixing the taxonomy, we validate it on a separate held-out set of development trajectories that is disjoint from both the taxonomy-discovery data and the final post-training corpus. We again retain only locally verifiable errors and assign each error to one of the six predefined categories or `OTHER`. Taxonomy coverage is measured as the fraction of held-out errors assigned to the six categories, with the complementary fraction reported as the `OTHER` rate. The resulting six-category taxonomy is then fixed and applied to the final post-training corpus for evidence-grounded counterfactual construction.

A.3 Training Data Construction

A.3.1 Training Corpus

The post-training corpus contains 10,200 tool-use training trajectories: 9,000 successful tool-use trajectories sampled from ToolMind and 1,200 execution-verifiable state trajectories extracted from τ-bench.

We first sample 9,000 trajectories from the constituent sources of ToolMind according to their original proportions. We then construct 1,200 execution-verifiable state trajectories from τ-bench to supplement supervision for interactions with explicit state changes.

These state trajectories are extracted rather than synthetically generated. We first identify state-changing tool calls in the τ-bench Retail training trajectories and recover the corresponding pre-execution and post-execution states from actual tool outputs in the original trajectories. For each candidate transition, we check the consistency among the tool arguments, pre-execution state, and post-execution state according to the semantics of the corresponding mutation operation, and retain only state changes that can be verified from actual execution results. We then extract the compact interaction segment containing the verified transition as a training trajectory. Consequently, both the before-state and after-state are grounded in tool outputs produced by actual environment execution rather than synthesized states.

This procedure ensures that the additional state-changing supervision is grounded in observed execution evidence without introducing synthetic or unverifiable state transitions.

Configuration	Value
Base model	Qwen3-8B
Training epochs	1 (full-corpus methods)
Global batch size	32
Maximum sequence length	8,192
Learning rate	2×10^{-4}
LoRA rank r	64
LoRA α	128
LoRA dropout	0.05
LR scheduler	Cosine
Warmup ratio	0.03

Table 5: Common configuration for the main Qwen3-8B post-training experiments.

Hyperparameter	Value
Initial-policy samples per state	4
Initial-policy temperature	0.8
Initial-policy top-p	0.9
Maximum generated tokens	256
Maximum counterfactuals per state $K_{\max}$	3
Minimum Contract confidence	0.8
Ranking coefficient λ	0.05
Target margin μ	0.1
Transition contrast length L_{prefix}	24
Maximum pair exposure	2

Table 6: BranchWise-specific supervision-construction and optimization settings.

A.4 Training and Implementation Details

A.4.1 Common Training Configuration

Our main experiments use Qwen3-8B as the base model. Full-corpus methods are trained for one epoch with a global batch size of 32, a maximum sequence length of 8,192, and a learning rate of

$$2 \times 10^{-4}.$$

For data-selection baselines, we retain each method's selected subset and repeatedly sample only from that subset so that its final fine-tuning parameter-update budget matches that of the full-corpus methods. Except for changes required by the method itself, the remaining major optimization settings are kept consistent.

A.4.2 BranchWise Configuration

BranchWise uses 4-bit NF4 QLoRA with double quantization and BF16 computation. LoRA adapters are applied to the query, key, value, and output projections in attention and to the gate, up, and down projections in the MLP. Training uses completion-only supervision without sequence packing.

The BranchWise-specific configuration is summarized in Table 6.

Model-derived counterfactual branches are generated only once before optimization. All candidate local boundaries therefore remain fixed during training; as the model evolves, only their current probability margins and the resulting optimization pressure assigned by margin-aware ranking change.

A.4.3 Qwen3-4B Configuration

We additionally evaluate BranchWise on Qwen3-4B to examine whether the main empirical trends persist at a second model scale. The Qwen3-4B experiments use the same training data, supervision-construction procedure, training hyperparameters, BranchWise-specific parameters, and evaluation

Method	Final training data / supervision	Key implementation setting
Vanilla SFT	All 10,200 trajectories	Standard completion-level negative log-likelihood without BranchWise counterfactual ranking or evidence-calibrated weighting.
T-SHIRT	Top 5% trajectories	Original hierarchical informativeness selection procedure and selection ratio, with DeepSeek-v4-Flash as the informativeness scorer; $M = 30$, $\alpha = 5$, $\gamma = 2$, and a 50% token-informativeness threshold.
Token Cleaning	All trajectories; top 60% response tokens supervised	Global token-level supervision filtering; filtered response tokens are excluded from the loss while the complete context is retained.
DaaR	Top 20% trajectories	Diversity-based data selection with an additional 5,000 distribution-matched probe trajectories disjoint from the final candidate training pool.
LESS	Top 5% trajectories	Uses 1,000 held-out ToolMind validation trajectories as target data; performs a four-epoch LoRA warm-up on 5% of the candidate data, followed by gradient influence scoring with an 8,192-dimensional gradient projection.
GIST	Top 5% trajectories	Uses the same 1,000 held-out ToolMind validation trajectories as LESS; all target trajectories are used with a rank-9 target projection.
Beyond Log Likelihood	All 10,200 trajectories	Uses its p variant and replaces the standard token-level NLL objective with $1 - p_\theta(y \mid x)$.

Table 7: Baseline configurations. All data-selection methods begin from the same 10,200-trajectory candidate pool; differences in the final amount of selected training data are part of their respective mechanisms.

protocol as the Qwen3-8B experiments. The only change is the backbone model, which is replaced by Qwen3-4B.

A.5 Baseline Implementation and Fair Comparison

A.5.1 Fair Comparison Protocol

All compared methods start from the same pool of 10,200 candidate training trajectories and use the same Qwen3 backbone at the corresponding model scale. Downstream evaluation instances from τ^2-Bench, BFCL V3, ToolHop, and ToolTalk-Hard are not used as supervised training data.

For methods trained on the complete corpus, we match the major optimization settings whenever compatible with the original method, including the training epoch, global batch size, maximum sequence length, and learning rate. For data-selection methods, the final training subset is determined by the method-specific selection mechanism.

A.5.2 Baseline-Specific Settings

To control the amount of final model optimization, all methods use the same final fine-tuning budget, measured by the number of final parameter updates. Data-selection methods retain their method-specific selected subsets and repeatedly sample only from these subsets until reaching the shared update budget; unselected examples are never reintroduced. This protocol preserves the core selection mechanism of each baseline while controlling the final parameter-update budget.

T-SHIRT. We retain T-SHIRT's original hierarchical informativeness selection procedure and selection ratio, but use DeepSeek-v4-Flash as the informativeness scorer. The remaining major selection hyperparameters are

$$M = 30, \qquad \alpha = 5, \qquad \gamma = 2,$$

with a 50% token-informativeness threshold. The top-scoring 5% of trajectories are selected for final fine-tuning.

All data-selection methods start from the same pool of 10,200 candidate trajectories; differences in the final amount of selected training data are inherent to their respective selection mechanisms.

A.6 Evaluation Protocol

Within each benchmark, all methods use identical prompts, tool definitions, reasoning configurations, interaction budgets, decoding settings, and scoring pipelines. All four benchmarks use deterministic decoding with temperature 0.

A.6.1 τ^2-Bench

We evaluate on the Airline, Retail, and Telecom domains of τ^2-Bench and report the three domain-specific scores together with the Overall score.

The complete evaluation contains 278 tasks:

- 50 Airline tasks;
- 114 Retail tasks;
- 114 Telecom tasks.

The evaluated agent uses temperature 0 with thinking disabled. All methods use the same tool definitions, interaction environment, and user simulator. User simulation uses DeepSeek-v4-Flash with temperature 0 and thinking disabled. Each task permits at most 200 environment interaction steps.

All methods are evaluated on the same task set, and final task success is computed by the τ^2-Bench evaluator.

A.6.2 BFCL V3

We evaluate all methods using the complete BFCL V3 function-calling evaluation protocol. The evaluation contains 17 categories and 4,441 cases in total. Following the official aggregation procedure, we report Non-Live, Live, Multi-Turn, and Overall performance.

All methods use the same Qwen3 function-calling formatting and tool definitions, with temperature set to 0 and the maximum context length set to 40,960. Model outputs are parsed and scored using the official BFCL V3 evaluation pipeline, and no category-specific evaluation configuration is changed across training methods.

A.6.3 ToolHop

ToolHop evaluation uses the complete set of 995 queries under each of the Direct, Mandatory, and Free protocols. Each model is therefore evaluated on the same query set under all three interaction settings.

All settings use temperature 0 and the same Qwen3 non-thinking prompt configuration. Direct permits a single model response without iterative tool interaction. Mandatory requires the agent to obtain information through tools during the multi-step interaction, whereas Free allows the model to decide whether and when tools are required. Mandatory and Free allow at most nine agent turns.

Tool-execution failures are returned to the agent as observations rather than directly terminating the task as infrastructure failures. Final correctness is computed using the same ToolHop answer-extraction and matching procedure for all compared methods.

A.6.4 ToolTalk-Hard

For ToolTalk-Hard, we report the two metrics used in our final evaluation: Success and Soft. All compared methods use the same ToolTalk-Hard prompts, tool environment, interaction protocol, decoding configuration, and evaluator, with temperature fixed to 0. No benchmark-specific evaluation configuration is modified across methods, allowing results to be compared under the same evaluation conditions.

Backbone	Method	# Training Runs	τ^2 / BFCL Repeats	ToolHop / ToolTalk Repeats	Aggregation
Qwen3-8B	Vanilla SFT	4	3	3	Mean
Qwen3-8B	T-SHIRT	3	2	2	Mean
Qwen3-8B	Token Cleaning	3	2	2	Mean
Qwen3-8B	GIST	3	2	2	Mean
Qwen3-8B	DaaR	3	2	2	Mean
Qwen3-8B	Beyond Log Likelihood	3	2	2	Mean
Qwen3-8B	LESS	3	2	2	Mean
Qwen3-8B	BranchWise	3	2	2	Mean
Qwen3-4B	Vanilla SFT	4	3	3	Mean
Qwen3-4B	BranchWise	4	3	3	Mean

Table 8: Numbers of independent training runs and repeated evaluations used for the reported results. Repeated evaluations are first averaged for each independently trained model, and the resulting run-level scores are then averaged across training runs.

A.6.5 Repeated Training and Evaluation

To reduce variance from both model training and interactive evaluation, we use multiple independent training runs and repeated evaluations. Repeated evaluations are first averaged within each independently trained model, after which the resulting run-level scores are averaged across training runs. Table 8 summarizes the number of training runs and evaluation repeats used for each setting.

The untrained Qwen3 base models do not involve a training seed and are therefore omitted from Table 8.