Counterfactual Tool-Use Distillation for Multimodal Question Answering

Pengcheng Xu (second author) et al.
University of California, Irvine

Abstract

External search can improve multimodal question answering. Redundant or misleading evidence, however, can reduce answer quality and increase interaction costs. Effective tool selection requires estimating how much a call can improve an answer given the available evidence and recognizing when further search is unnecessary. We introduce Counterfactual Tool-Use Distillation (CTD), which trains a lightweight predictor on quality differences between paired answers generated by a frozen model with and without each candidate call's evidence. Using only information available before execution, CTD estimates each call's added value and selects a call or stops according to an adjustable cost threshold. On FVQA, InfoSeek, MMSearch, SimpleVQA, and LiveVQA, CTD improves macro-averaged accuracy by 2.93–5.75 percentage points across three backbones while using 50.0–52.6% fewer calls than the strongest evaluated baseline for each backbone under a shared two-call limit. CTD uses the expected value of additional evidence to guide selective search with a frozen answering model.

1. Introduction

Answering questions about unfamiliar visual entities often requires knowledge that is absent from the image and cannot be reliably recalled from model parameters [1]. Text and image search extend multimodal question answering by connecting visual observations to external evidence [9]. For practical visual assistants, the challenge is to acquire information that improves the answer while keeping interaction costs manageable.

Recent agents learn when and how to search. MMSearch-R1 learns to invoke image and text search on demand [20], and MM-DeepResearch uses expert-guided trajectories to improve evidence gathering [22]. Task success and the usefulness of an individual interaction can diverge: a correct answer may follow a redundant call, and a useful retrieval may still be followed by an incorrect answer [4]. This motivates measuring each call's contribution when learning which information to acquire.

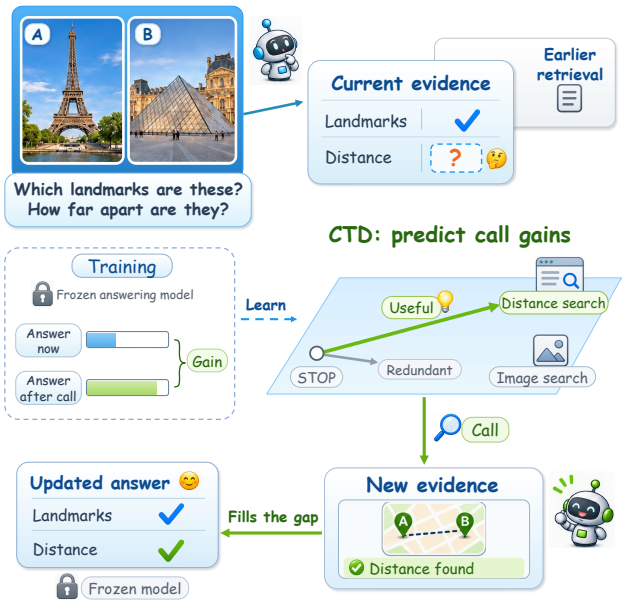

Figure 1. **CTD selects evidence that fills the remaining gap.** In this illustrative example, the landmarks are already identified, but their distance is unknown. CTD predicts call gains before execution, favoring distance search over redundant image search, or stops if no call is worthwhile. Paired answers with and without additional evidence provide supervision while the answering model stays frozen.

Selective tool learning addresses this mismatch. CoBRA compares internal and tool-augmented reasoning to learn when to seek external assistance [24]. TACO rewards tool branches for their contribution to answering correctly [5], and ToolVision aligns tool-use supervision with the learner's capabilities [12]. These methods primarily use tool utility to train an agent's behavior. Choosing the next interaction requires comparing specific calls before their results are known. Their value also changes as evidence accumulates: a call that once supplied a missing fact may later repeat information already available.

We ask whether anticipating the remaining value of evidence can improve both answer quality and interaction efficiency without changing the answering model. We introduce

Counterfactual Tool-Use Distillation (CTD), which learns the expected benefit of a candidate acquisition relative to answering with the evidence already available (Fig. 1). During training, a frozen answering model produces matched answers with and without each candidate's evidence to supervise a lightweight predictor. At deployment, the predictor uses only information available before execution to choose a worthwhile call or stop. An adjustable cost threshold controls this decision without retraining. CTD targets the immediate value of one additional acquisition; we also evaluate repeated decisions in longer searches.

Our contributions are threefold:

- **We reveal how tool utility depends on the available evidence.** Controlled audits show that helpful and harmful candidates can coexist at the same state. Matched evidence interventions further show how the value of a fixed call decreases as the missing information is supplied.
- **We propose CTD for selective evidence acquisition.** CTD separates evidence selection from answer-model training and supports candidate choice and stopping under adjustable interaction costs. Controlled analyses identify complementary gains from better stopping and better candidate selection.
- **CTD improves answer quality with fewer tool calls.** Across three backbones and five search benchmarks, CTD improves macro accuracy by 2.93–5.75 percentage points while using 50.0–52.6% fewer calls than the strongest compared baseline for each backbone under the shared two-call evaluation protocol.

2. Related Work

Multimodal tool learning and search. ReAct [23] established interleaved reasoning and acting. MMSearch [9] organized multimodal search into a prompted workflow. Subsequent agents learn interaction policies through reinforcement learning: Visual-ARFT [11] and SenseNova-MARS [3] combine retrieval with visual operations, and MMSearch-R1 [20] learns on-demand image and text search. Trajectory supervision ranges from verified reasoning chains in Search-Align [13] to synthetic traces in WebWatcher [8] and expert-guided tree search in MM-DeepResearch [22]. CTD focuses on each call's incremental value under a fixed answering model, separating evidence selection from backbone optimization.

Selective retrieval and counterfactual routing. Toolformer [17] filters calls by reductions in future-token loss, and SKR [18] compares answers with and without retrieval to supervise selective augmentation. Counterfactual search routing [10] derives search/no-search labels from paired outcomes, and CoBRA [24] uses internal–external expert reward margins for boundary-aware fine-tuning and reinforcement learning. The latter methods target whether to consult external information. HDPO [21] encourages economical tool use within correct trajectories, and CAM-DF [6] learns where to truncate a ranked tool set before execution. CTD predicts signed task-score gains for multiple parameterized calls from the same multimodal state, ranks feasible candidates, and compares their net gains with stopping.

Fine-grained tool utility and credit. Visual evidence audits [19] find that some returned observations leave answers unchanged, motivating supervision beyond aggregate success. TAPO [4] transfers credit between calls with similar parameters, and TACO [5] measures tool-branch utility through answer probes before and after a call. ToolVision [12] uses capability-aligned evidence gains for trajectory construction and tool/no-tool comparisons for necessity-aware reinforcement learning. These methods use differential utility and related credit signals to update the agent. CTD distills paired score differences with and without one additional acquisition into a separate predictor using frozen backbone features. Its output estimates a candidate's immediate utility before execution and supports an adjustable cost threshold.

3. Preliminary Study

Can a plausible tool call make an answer worse? In multimodal question answering, a valid retrieval call can improve an answer, repeat existing evidence, or reduce answer quality. We examine whether helpful and harmful candidates coexist at the same state, and how their effects vary with the evidence already available.

A frozen-model intervention audit. We audited 200 development task groups from MC-Search [13], covering 469 pre-call states. A frozen behavior policy supplied each initial state and up to two automatically sampled prefix states. A frozen candidate generator proposed up to two text-retrieval and two image-retrieval calls. For each state, we compared answers from the existing evidence with answers augmented by each cached call result, using the same prompt, answer limit, and token-F1 evaluator. Three matched answer draws define the observed difference $d_{s,j} = \bar{V}(s, z_j) - \bar{V}(s, \emptyset)$. We then selected the candidate with the largest measured difference, or STOP when none was positive. A separate set of three answer draws evaluated this action against the first-proposed call. Reference answers were used only for retrospective scoring and figure stratification.

Observed failure patterns. Figure 2(a) maps the best and worst gains at each state, using zero to separate gain signs. Under the stricter thresholds of gains above $+2.5$ and below -2.5 percentage points, 47.5% of all states contain both helpful and harmful candidates. Another 17.5% have no positive-gain candidate. Separately, the first-proposed call reduces F1 by more than 2.5 points in 40.7% of states. Figure 2(b) shows how decision composition varies with depth and direct-answer F1. Among states with two prior calls in the highest F1 quartile, 40.4% select STOP in the initial audit. In the held-out answer draws, retrospective selection

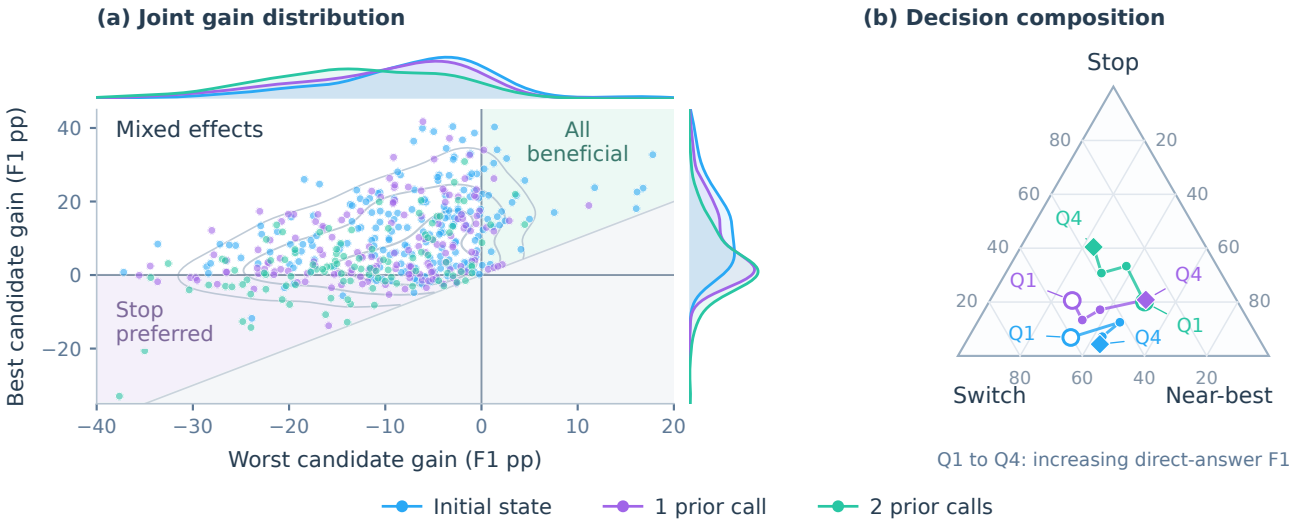

Figure 2. **Joint candidate effects and decision composition. (a)** Each point places a state's worst and best candidate gains relative to answering directly, in token-F1 percentage points. Zero lines define the sign-based regions; the 47.5% reported in the text additionally requires a gain above $+2.5$ and another below -2.5 points. Gray contours show pooled kernel density; marginal curves show density normalized within each depth. **(b)** Each point gives the percentages of STOP (no positive estimated gain), SWITCH (the best positive call exceeds the first by more than 3.5 points), and NEAR-BEST (remaining states) within a depth and global direct-answer F1 quartile. Lines join the Q1–Q4 groups at each depth in order, from hollow circles to diamonds. Ternary ticks are percentages. Both panels share the depth colors.

yields an 11.7-point mean advantage over the first-proposed call for initial states in the lowest global F1 quartile. These results reveal opportunities to improve candidate selection and avoid unhelpful acquisition.

Implication for the method. These within-state differences motivate a predictor that estimates each call's gain relative to answering immediately, using the current state and call parameters before execution.

4. Method

4.1. Overview

Counterfactual Tool-Use Distillation (CTD) predicts how much an additional tool call will improve an answer over using the current evidence alone. Offline, we compare answers from a frozen multimodal model with and without each candidate's result. These score differences train a small predictor that uses only the state and proposed call. Online, it selects the feasible call with the highest positive gain after accounting for cost, or stops and answers from the current evidence. The backbone, candidate generator, and tools remain fixed throughout. Figure 3 summarizes the procedure.

At each decision, state s contains the question, input images, accumulated evidence, and resources available at decision entry. A frozen generator G produces an ordered list of validated calls $\mathcal{T}_s = (t_1, \ldots, t_{K_s})$. Each call $t_j = (k_j, p_j)$ specifies a tool type k_j and parameters p_j; different queries to the same tool are separate candidates. Action 0, denoted STOP, ends acquisition and answers from the existing state. A selected call is followed immediately by a final answer. We apply this one-step rule to both initial states and states reached after earlier tool calls. Supplementary Sec. A.1 describes state collection and candidate construction.

4.2. Counterfactual gain supervision

To construct gain labels, we execute each candidate t_j once using a fixed tool snapshot and cache its result z_j. We compare answering with this result against answering from s alone. The latter is the reference branch: $z_0 = \varnothing$ leaves the additional-evidence slot empty while retaining all evidence in s. For reference answer y and sampling repetition r, branch j receives score

$$u_{j,r}(s) = V(F_\phi(s, z_j; \xi_r), y), \quad j = 0, \ldots, K_s, \tag{1}$$

where F_ϕ is the frozen answer model, ξ_r is the sampling seed, and $V(\hat{y}, y) \in [0, 1]$ scores the answer. Within each repetition, all branches share the seed, prompt template, decoder, and answer-length limit; only the added evidence changes. We use token F1, with scoring and failure rules in Suppl. Sec. A.3.

Subtracting the reference score gives the expected gain and its sample estimate:

$$\begin{aligned} \Delta_j(s) &= \mathbb{E}_\xi[u_{j,r}(s) - u_{0,r}(s)], \\ \widehat{\Delta}_j(s) &= \frac{1}{R}\sum_{r=1}^{R}[u_{j,r}(s) - u_{0,r}(s)]. \end{aligned} \tag{2}$$

Both lie in $[-1, 1]$: positive gains indicate better answers and negative gains indicate worse answers under the fixed

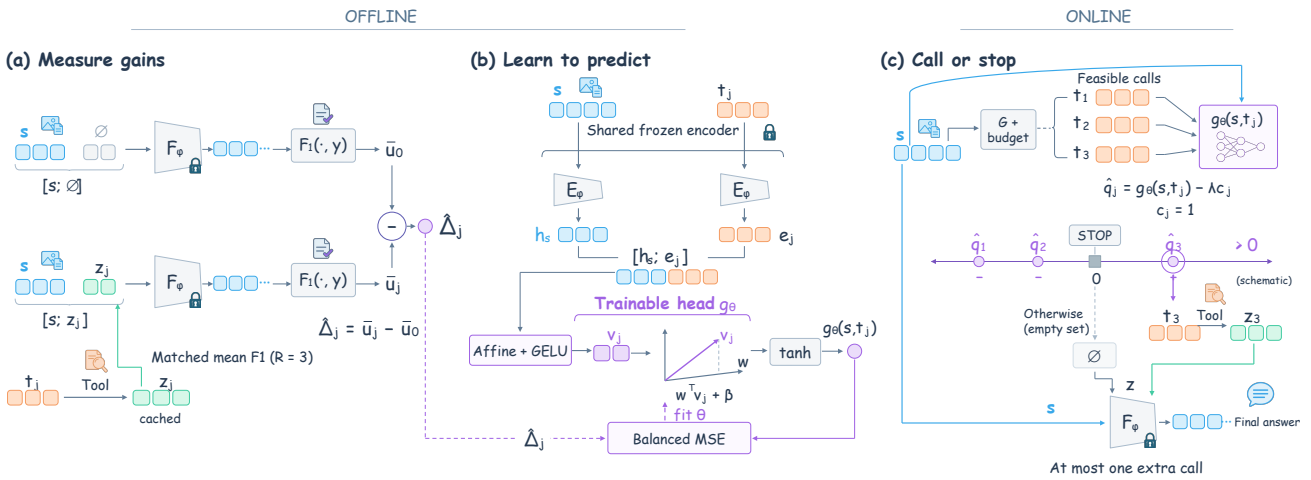

Figure 3. **Overview of Counterfactual Tool-Use Distillation (CTD). (a)** Offline, a frozen model answers from the same state with and without a cached candidate result. Averaging token-F1 differences over $R = 3$ matched repetitions gives the gain target $\widehat{\Delta}_j$. **(b)** A trainable head predicts these gains from frozen pre-call state and call encodings using task-balanced squared loss. **(c)** Online, the router executes the feasible call with the highest strictly positive predicted net gain $g_\theta(s, t_j) - \lambda c_j$; otherwise, it chooses STOP with value 0. Both paths end in a final answer after at most one additional tool call.

protocol. The expectation is over answer sampling with the state and cached result held fixed. We use $R = 3$ repetitions and retain all measurable estimates, including negative and near-zero gains.

Each repetition shares one reference score across candidates, requiring $(K_s + 1)R$ answer generations per state. This common subtraction preserves candidate ordering and sets the stopping reference to zero. Final evaluation uses the same answer sampler with fresh, independent draws. Suppl. Sec. D characterizes these score differences and their sampling covariance.

4.3. Predicting gains before execution

The router predicts gains from pre-call information. The frozen backbone separately encodes the state as $\mathbf{h}_s \in \mathbb{R}^d$ and the serialized call as $\mathbf{e}_j \in \mathbb{R}^d$. Each representation is the last-layer hidden vector at its input's final non-padding token. State inputs include the available images and evidence; call inputs specify the tool type and parameters. Suppl. Sec. A.2 details serialization and image indexing.

A shared two-layer head maps these features to a gain:

$$g_\theta(s, t_j) = \tanh\big(\mathbf{w}^\top \sigma(\mathbf{W}[\mathbf{h}_s; \mathbf{e}_j] + \mathbf{b}) + \beta\big), \quad (3)$$

where $\mathbf{W} \in \mathbb{R}^{m \times 2d}$, $\mathbf{b}, \mathbf{w} \in \mathbb{R}^m$, $\beta \in \mathbb{R}$, $m = 256$, and σ is GELU. Concatenation $[\mathbf{h}_s; \mathbf{e}_j]$ joins the state and call features, and $\tanh$ keeps predictions in $[-1, 1]$. Only $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{w}, \beta\}$ is trained.

For each training task x, let $\mathcal{S}_x$ contain sampled states with at least one candidate and complete branch measurements. Let $\mathcal{D}$ contain tasks with nonempty $\mathcal{S}_x$. With $\mathbf{g}_\theta(s)$ and $\widehat{\mathbf{\Delta}}(s)$ denoting the K_s-dimensional prediction and target vectors, we minimize

$$\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \frac{1}{|\mathcal{S}_x|} \sum_{s \in \mathcal{S}_x} \frac{\|\mathbf{g}_\theta(s) - \widehat{\mathbf{\Delta}}(s)\|_2^2}{K_s}. \quad (4)$$

The loss weights tasks equally and averages over states within each task and candidates within each state. Squared error targets the conditional mean gain in answer-score units, with zero meaning no expected improvement. States excluded from regression remain in task-level evaluation. Suppl. Sec. A.3 specifies measurement exclusions; Algorithm S1 gives the training procedure.

4.4. Resource-constrained selection

After accounting for candidate generation and feature extraction, we check which calls fit the remaining budget. The feasible set $\mathcal{F}(s)$ includes calls whose execution leaves enough resources for the final answer. Suppl. Sec. A.4 details this resource ledger. Let $c_j \geq 0$ be the known charge for executing t_j, with $c_0 = 0$. The predicted net gains are

$$\widehat{q}_0(s) = 0, \qquad \widehat{q}_j(s) = g_\theta(s, t_j) - \lambda c_j, \quad (5)$$

where $\lambda \geq 0$ converts call charges to answer-score units. We select

$$\hat{a}(s) \in \operatorname*{arg\,max}_{a \in \mathcal{F}(s) \cup \{0\}} \widehat{q}_a(s). \quad (6)$$

Ties favor STOP, then lower charge, then earlier position in $\mathcal{T}_s$. This rule returns a direct answer if no call is feasible or all predicted net gains are nonpositive. Otherwise, we execute the selected call once and use F_ϕ to answer from $(s, z_{\hat{a}})$ under the same protocol used for labeling. Algorithm S2 includes the resource checks.

Every attempted call, including failures, costs one charge unit. With uniform charges, λ controls stopping while predicted gains rank the tools. We select λ on development tasks and can adjust it without retraining the predictor. Model-token consumption and execution time are accounted for separately.

For a fixed state and feasible set, if every gain prediction is within ε of its true value, the selected action loses at most 2ε in expected net gain relative to the best feasible action. Suppl. Sec. D.3 proves this bound and gives the condition for recovering the optimal action.

5. Experiments

We evaluate CTD through four research questions:

RQ1. Can CTD improve answer quality with fewer tool calls? (Sec. 5.2)

RQ2. Which decision mechanisms account for the gains? (Sec. 5.3)

RQ3. Can the quality–cost balance be controlled without retraining? (Sec. 5.4)

RQ4. Does tool value adapt to the evidence already available? (Sec. 5.5)

5.1. Experimental setup

Benchmarks and metrics. Our evaluation centers on MC-Search [13], with task- and image-disjoint training, development, and test groups. Transfer evaluation covers MMSearch [9], FVQA-test [20], InfoSeek [1], SimpleVQA [2], and LiveVQA [7]; BrowseComp-VL [8], MAT-Search [11], and HR-MMSearch [3] provide harder search and perception tests. We report answer F1 and improvement over direct answering on MC-Search, native end-to-end F1 on MMSearch, and judge accuracy on the search-agent evaluation subsets. Efficiency uses mean tool attempts; resource profiling on Qwen2.5-VL-7B additionally measures model tokens and end-to-end latency on MC-Search and MMSearch-171. Gain MAE, signed calibration, and an independently evaluated local reference gap assess the routing mechanism. Dataset variants and metrics are specified in Suppl. Sec. B.

Comparison protocol. The primary comparison permits one additional acquisition followed by a final answer, sharing the frozen answer model, candidate calls, tool returns, and resource limits across routers. Controls include direct answering, fixed retrieval, confidence gating, success-only imitation, a Toolformer-inspired likelihood proxy [17], and a CoBRA-inspired counterfactual gate [24]. Separate agent comparisons include ReAct [23], Search-Align (ICLR 2026) [13], MMSearch-R1 (ACL 2026) [20], MM-DeepResearch (ICML 2026) [22], Visual-ARFT [11], WebWatcher [8], and SenseNova-MARS [3]. These use matched evaluation budgets within each backbone family; released checkpoints and methods adapted to restricted tools are identified separately.

Backbones and implementation. We instantiate CTD with Qwen2.5-VL-7B-Instruct [15], Qwen3-VL-8B-Instruct [16], and InternVL3.5-8B [14]. The hardware configuration uses eight NVIDIA H100 GPUs. Each backbone supplies frozen features to a two-layer gain head of hidden width 256. The protocol uses at most four candidates, three paired answer samples per label, and AdamW with learning rate 3×10^{-4}. Head selection and the call penalty use development groups only. Controlled-router comparisons use three training seeds and paired group-bootstrap intervals. Suppl. Sec. B details the supplementary comparisons and resource accounting.

5.2. RQ1: Answer Quality and Tool Use

We assess quality and tool use at two levels: controlled routing with shared candidate outcomes, and search-agent comparisons under matched evaluation protocols.

Controlled tool selection. CTD achieves the highest F1 across all three backbones in Table 1: 43.08, 49.56, and 45.72 for Qwen2.5-VL, Qwen3-VL, and InternVL3.5, respectively. These scores improve on direct answering by 6.66, 6.73, and 6.09 points with 0.46–0.49 calls per task. Against the strongest routing control, absolute-value regression, CTD gains a further 1.46, 1.37, and 1.39 points at similar call rates. With candidates, tool returns, and answer sampling held fixed, these gains support learning signed values for individual calls. Table S3 evaluates the greedy two-call extension across reasoning graphs.

Transfer to search benchmarks. Table 2 compares the search agents under the shared protocol with a two-call cap, identical evaluation subsets, and common tool interfaces and judging. At $B = 2$, CTD reaches macro accuracies of 61.33, 70.33, and 60.16 across the three backbone families, exceeding their strongest matched baselines by 2.93, 3.29, and 5.75 points while using 50.0–52.6% fewer calls. Section 5.4 examines how this balance changes with the decision threshold and budget.

Harder acquisition and fine-grained perception. CTD also improves the matched harder-benchmark results in Table S4(c). With Qwen3-VL-8B, it reaches 70.34 F1 on full MMSearch, 42.78% accuracy on BrowseComp-VL, and 47.43% accuracy on HR-MMSearch, improving on the strongest same-family baseline for each metric by 3.53, 1.89, and 4.26 points. MAT-Search reaches 78.09 F1 and 71.33% EM, while mean usage across the four benchmarks is 0.92 calls. Suppl. Sec. C.2 reports additional native metrics.

Takeaway RQ1. CTD improves answer quality and reduces mean tool calls across the evaluated backbones and search benchmarks under the shared cap.

Table 1. **Controlled one-step routing on MC-Search.** Each backbone uses the same grouped test split, frozen answer model, candidates, and one-call cap. F1 is on a 0–100 scale; ΔF1 is the change from direct answering; Calls counts attempted acquisitions. Bold and underlining indicate the best and second-best values within each backbone, including ties.

	Qwen2.5-VL-7B			Qwen3-VL-8B			InternVL3.5-8B		
Router	F1↑	ΔF1↑	Calls↓	F1↑	ΔF1↑	Calls↓	F1↑	ΔF1↑	Calls↓
Direct answering [13]	36.42	0.00	**0.00**	42.83	0.00	**0.00**	39.63	0.00	**0.00**
Fixed-first retrieval [13]	37.68	+1.26	0.96	44.16	+1.33	0.95	40.56	+0.93	0.95
Random routing	37.08	+0.66	0.48	43.39	+0.56	0.48	40.07	+0.44	0.48
Confidence gate	39.16	+2.74	0.47	45.47	+2.64	0.46	42.02	+2.39	0.48
Entropy gate	38.79	+2.37	0.49	45.11	+2.28	0.47	41.67	+2.04	0.47
Success-only imitation [13]	39.84	+3.42	0.46	46.55	+3.72	0.48	42.78	+3.15	0.46
Likelihood proxy [17]	40.31	+3.89	0.48	47.01	+4.18	0.47	43.21	+3.58	0.48
Binary gate + ranker [24]	40.93	+4.51	0.47	47.72	+4.89	0.49	43.94	+4.31	0.49
Absolute-value regression	41.62	+5.20	0.46	48.19	+5.36	0.48	44.33	+4.70	0.48
CTD (ours)	**43.08**	**+6.66**	0.47	**49.56**	**+6.73**	0.46	**45.72**	**+6.09**	0.49

Table 2. **Search-agent comparison under the shared protocol ($B = 2$).** All methods share evaluation subsets, tool interfaces, cached returns, and the judge. Scores are judge accuracies (%); Macro and Calls average per-dataset accuracy and attempted calls, respectively. Bold and underlining indicate the best and second-best values within each backbone.

Backbone	Method	FVQA-test 1,800	InfoSeek 2,000	MMSearch 171	SimpleVQA 1,013	LiveVQA 3,602	Macro↑	Calls↓
Qwen2.5-VL-7B	Fixed image–text RAG [20]	54.28	52.17	51.46	53.27	48.34	51.90	1.91
	ReAct [23]	50.89	47.83	45.61	50.64	44.43	47.88	1.18
	Visual-ARFT (restricted) [11]	56.31	50.78	49.71	54.13	47.51	51.69	1.43
	MMSearch-R1 [20]	59.43	55.68	54.97	58.01	51.59	55.94	1.39
	WebWatcher (restricted) [8]	54.74	52.63	52.24	55.81	50.93	53.27	1.66
	MM-DeepResearch (restricted) [22]	60.04	58.07	59.45	59.62	54.82	58.40	1.72
	CTD (ours)	**63.09**	**60.77**	**61.99**	**62.88**	**57.91**	**61.33**	**0.84**
Qwen3-VL-8B	Fixed image–text RAG [20]	62.41	56.73	57.12	63.21	54.32	58.76	1.93
	ReAct [23]	56.19	53.42	50.49	57.52	48.74	53.27	1.22
	SenseNova-MARS (restricted) [3]	68.33	63.62	66.28	69.27	58.63	65.23	1.55
	MM-DeepResearch (restricted) [22]	69.61	69.17	67.25	67.36	61.82	67.04	1.76
	CTD (ours)	**72.37**	**70.83**	**70.57**	**72.92**	**64.98**	**70.33**	**0.88**
InternVL3.5-8B	Fixed image–text RAG [20]	57.15	53.77	53.80	56.24	51.10	54.41	1.92
	ReAct [23]	52.59	49.83	47.95	52.71	46.11	49.84	1.20
	CTD (ours)	**62.46**	**59.53**	**60.23**	**62.36**	**56.23**	**60.16**	**0.91**

5.3. RQ2: Sources of Better Decisions

We examine gain estimation, stopping, and candidate ranking, with controls for supervision cost and decoding.

Gain prediction and routing quality. Relative to absolute-value regression (AV), CTD reduces gain MAE by 20.2–21.3% and signed-gain calibration error by 61.4–63.4% across the three backbones (Figure 4). Independently evaluated reference gaps decrease by 1.30–1.34 F1 points on initial states and 1.21–1.38 points on collected-prefix states, with corresponding mean call counts differing by at most 0.02 (Table S6).

Policy-swap attribution. We cross the stopping gates and candidate rankers of CTD and AV on 24,000 states with four feasible candidates ($B = 1$, $\lambda = 0.05$), holding candidate outcomes fixed. CTD achieves 42.73 F1 versus 41.37 for AV at comparable call rates (0.463 versus 0.459). A symmetric decomposition attributes 0.52 of the 1.36-point gain to stopping and 0.84 to ranking.

Complementary roles. The advantage varies with tool value (Figure 5). When the best candidate exceeds the call penalty ($G > 0$), ranking contributes 1.75 of the 2.42-point mean gain. When all candidates fall well below the penalty ($G < -10$ points), stopping contributes 0.64 of the 0.66-point gain. Suppl. Sec. C.4 gives the decomposition and regional results.

Controls and repeatability. On Qwen2.5-VL-7B, CTD ex-

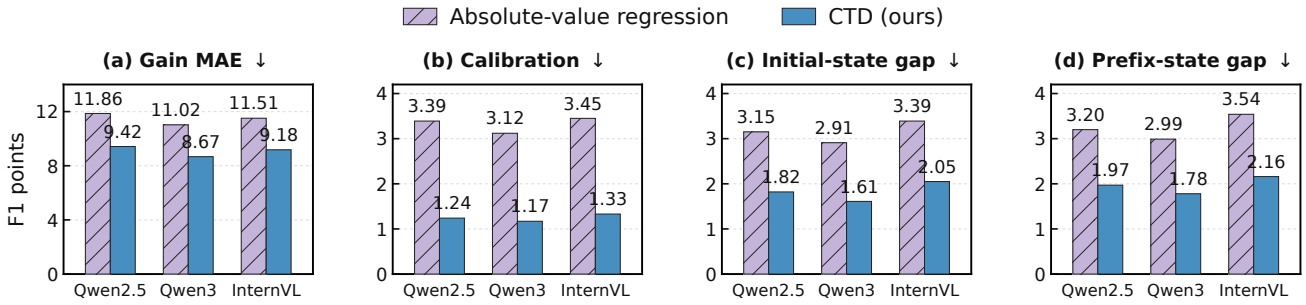

Figure 4. **Gain prediction and local routing on held-out MC-Search states.** Values are three-seed means in F1 percentage points (lower is better); calibration uses SCal_{10}. Qwen2.5, Qwen3, and InternVL denote Qwen2.5-VL-7B, Qwen3-VL-8B, and InternVL3.5-8B, respectively. Table S6(a) reports the complete results, including call counts.

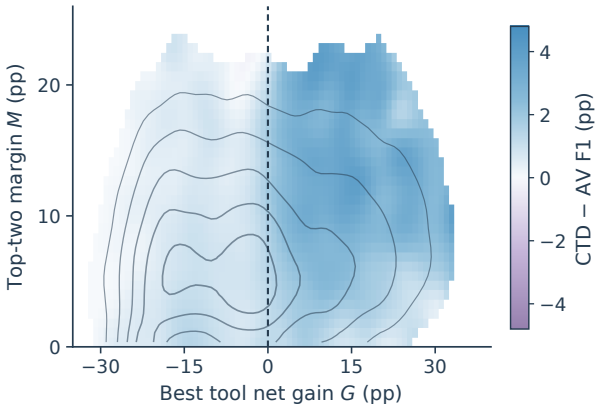

Figure 5. **Where CTD improves over AV.** Color shows the local F1 advantage and solid contours show state density. G is the best tool net gain and M the top-two candidate gap, both in F1 points. The dashed line marks $G = 0$.

ceeds AV by 1.44 F1 points under common supervision-cost limits and by 1.43 points with greedy decoding (Table S11). The paired 95% intervals for the standard-setting F1 gains exclude zero for all backbones (Table S12). The advantage thus persists across the tested supervision budgets and decoders. Table S10 examines sensitivity to candidate count, label repetitions, and head width on the default backbone. Increasing the candidate count from 4 to 8 adds 0.35 F1 points while raising offline tokens from 312.6 to 489.7 million. Five label draws add 0.21 points at 470.1 million tokens, whereas doubling head width adds 0.03 points.

Takeaway RQ2. Gains relative to the current answer provide a common scale for deciding whether to search and which call to select.

5.4. RQ3: Controlling Quality and Cost

We vary the call penalty and budget to examine their effects on answer quality and resource use.

Threshold control without refitting. Figure 6 varies the call penalty while keeping each predictor fixed. Increasing CTD's penalty from $\lambda = 0$ to the development-selected 0.05 reduces mean calls by 33.8–34.7% and improves F1 by 0.76–0.90 points across the three backbones, consistent with avoiding low-value acquisitions. Larger penalties reduce both call usage and answer quality, reaching direct answering at $\lambda = 0.4$. Table S9 gives the full sweep.

Additional acquisition budgets. Table S2 varies the cap in the matched search-agent setting. At $B = 1$, CTD exceeds the strongest baselines' $B = 2$ macro accuracies from Sec. 5.2. Increasing CTD's cap to four calls raises macro accuracy to 63.99, 73.24, and 62.83, with mean usage of 1.31, 1.40, and 1.46 calls. CTD benefits from a larger budget while continuing to select calls according to the current evidence.

Measured resource use. On Qwen2.5-VL-7B and MMSearch-171, CTD reduces uncached median latency by 34.3% relative to MM-DeepResearch (Table S13). The median and P95 latencies are 16.84 and 33.71 seconds for CTD, compared with 25.63 and 50.92 seconds for MM-DeepResearch. Cached replay highlights the contribution of retrieval time to overall latency: fixed image–text RAG has a lower median latency than CTD (9.44 versus 11.09 seconds). For one-step routing, CTD and AV have similar cached medians (5.92 and 5.97 seconds). The table also reports model tokens and auxiliary computation alongside latency and peak GPU memory usage.

Takeaway RQ3. Adjusting the call penalty controls the quality–cost trade-off with a fixed gain predictor. The development-selected penalty improves F1 and reduces calls relative to a zero penalty.

5.5. RQ4: Adapting to Available Evidence

We test how a fixed call's value changes with the available evidence.

Matched evidence interventions. For each of 1,440 parent states, we construct missing, partial, sufficient, and irrelevant evidence conditions while holding the question, images,

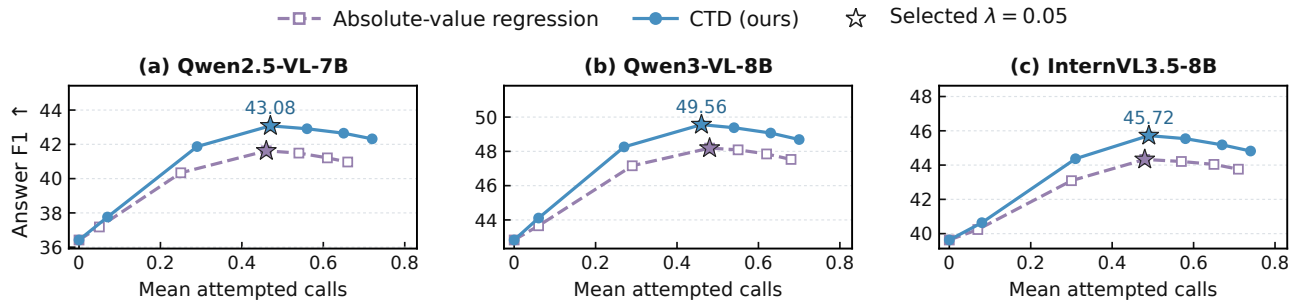

Figure 6. **Quality–cost control without refitting.** MC-Search initial states, $B = 1$. Each curve connects three-seed mean results for seven penalties using a fixed predictor; stars mark $\lambda = 0.05$, selected on development data. The zero-call endpoints recover direct answering. Complete values are in Table S9.

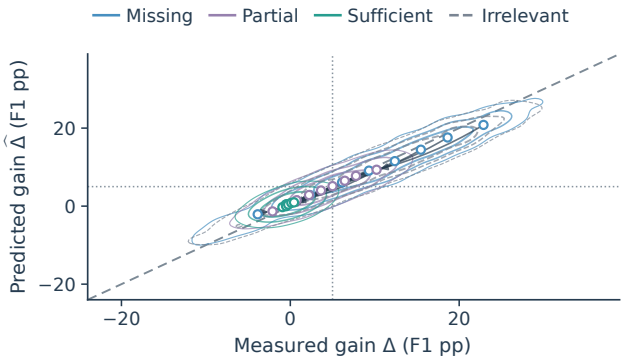

Figure 7. **State-conditioned tool gains.** Contours show gain distributions for four matched evidence states; arrows connect conditional means from missing to partial to sufficient evidence. The dashed diagonal marks equal gains; dotted lines mark the five-point call threshold.

candidate call, cached return, and budget fixed. All inserted contexts contain 128 tokens.

Tracking evidence sufficiency. Mean measured gain decreases from 10.23 to 4.13 to −0.32 F1 points across missing, partial, and sufficient states; CTD predicts 9.87, 4.38, and 0.40 points (Figure 7). Length-matched irrelevant evidence changes measured and predicted gains by only −0.11 and −0.25 points, respectively. These matched changes support CTD's sensitivity to the information supplied by the evidence. Suppl. Sec. C.5 reports the paired contrasts and condition-level results.

Concrete decisions. Figure 8 illustrates the decisions examined in RQ2–RQ4. For the frog, CTD's species-and-coloration query retrieves evidence about *Oophaga pumilio*. AV* uses the same tool for one call but queries *Dendrobates tinctorius*, leading to an incorrect answer. For the eagle balloon, CTD stops with the correct answer; forced image retrieval repeats information already present in the input. For ECCV, CTD issues the date query when the dates are missing and stops with the same answer when the prefix already supplies them.

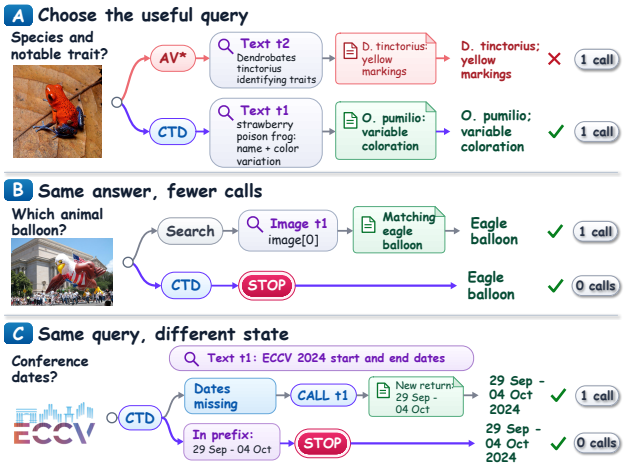

Figure 8. **Qualitative comparison of tool-use decisions.** (A–B) are from MC-Search [13]; (C) is from MMSearch [9]. CTD selects the useful query (A), avoids redundant retrieval (B), and conditions its decision on existing evidence (C). AV* denotes absolute-value regression; Search denotes forced image retrieval. Text is abbreviated; counts indicate additional calls.

Takeaway RQ4. CTD lowers a call's predicted value as relevant evidence accumulates, while length-matched irrelevant evidence has little effect.

6. Conclusion

We presented Counterfactual Tool-Use Distillation (CTD) for deciding which additional evidence is worth acquiring in multimodal question answering. Paired answers from a frozen model supervise the expected improvement over the current answer, guiding both call selection and stopping. Experiments across three backbones show that these two decisions contribute complementary gains, improving answer quality while reducing tool use. The results support selecting tools by the information they add and reassessing their value as evidence accumulates. Extending this approach to coordinated sequences of calls could capture benefits that emerge only from their combined evidence.

LLM Usage Statement

OpenAI Codex assisted with language polishing, related-work searches and citation checks, LaTeX formatting adjustments, and analysis and plotting code. OpenAI's image-generation tool created illustrative assets for Figs. 1 and 3. The vision-language models used for candidate generation, answering, and evaluation are specified in Sec. 5.1 and Suppl. Sec. B. The authors take full responsibility for the final manuscript, code, data, results, and scientific claims.

References

[1] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14948–14968, 2023. 1, 5

[2] Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, Yutao Zeng, Zhoufutu Wen, Ke Jin, Baorui Wang, Weixiao Zhou, Yunhong Lu, Hangyuan Ji, Tongliang Li, Wenhao Huang, and Zhoujun Li. SimpleVQA: Multimodal factuality evaluation for multimodal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4637–4646, 2025. 5

[3] Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Chengjun Xie, Gao Huang, Dahua Lin, and Lewei Lu. SenseNova-MARS: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv preprint arXiv:2512.24330*, 2025. Version 2, January 2026. 2, 5, 6

[4] Chengqi Dong, Chuhuai Yue, Hang He, Yandong Liu, Fenghe Tang, S. Kevin Zhou, Xiaohan Wang, Jiajun Chai, and Guojun Yin. TAPO: Tool-aware policy optimization via credit transfer for multimodal search agents. *arXiv preprint arXiv:2606.05784*, 2026. 1, 2

[5] Mingkuan Feng, Jinyang Wu, Hao Gu, Fangrui Lv, Ruihan Jin, Chuyuan Zhang, Zhengqi Wen, and Jianhua Tao. TACO: Tool-augmented credit optimization for agentic tool use. *arXiv preprint arXiv:2606.30251*, 2026. 1, 2

[6] Yicheng Feng, Yan Zhang, Yan Cheng, and Wei Qi. Scores are not decisions: Cost-aware stopping for tool acquisition in LLM agents. *arXiv preprint arXiv:2607.27083*, 2026. 2

[7] Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S. Yu, and Ranjay Krishna. Seeking and updating with live visual knowledge. *arXiv preprint arXiv:2504.05288*, 2025. 5

[8] Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebWatcher: Breaking new frontier of vision-language deep research agent. *arXiv preprint arXiv:2508.05748*, 2025. 2, 5, 6

[9] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. MMSearch: Unveiling the potential of large models as multi-modal search engines. In *International Conference on Learning Representations*, 2025. 1, 2, 5, 8

[10] Minho Kim. When should LLMs search? counterfactual supervision for search routing. *arXiv preprint arXiv:2607.05752*, 2026. 2

[11] Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning. *arXiv preprint arXiv:2505.14246*, 2025. 2, 5, 6

[12] Delin Mao, Chenghao Sun, Jingwei Song, Chishui Chen, and Linfeng Zhang. ToolVision: Learning when and how to use visual tools with capability-aligned supervision. *arXiv preprint arXiv:2608.08907*, 2026. 1, 2

[13] Xuying Ning, Dongqi Fu, Tianxin Wei, Mengting Ai, Jiaru Zou, Ting-Wei Li, Hanghang Tong, Yada Zhu, Hendrik Hamann, and Jingrui He. MC-Search: Evaluating and enhancing multimodal agentic search with structured long reasoning chains. In *International Conference on Learning Representations*, 2026. 2, 5, 6, 8

[14] OpenGVLab. InternVL3.5-8B model card. Hugging Face, 2025. 5

[15] Qwen Team. Qwen2.5-VL-7B-Instruct model card. Hugging Face, 2025. 5

[16] Qwen Team. Qwen3-VL-8B-Instruct model card. Hugging Face, 2025. 5

[17] Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023. 2, 5, 6

[18] Yile Wang, Peng Li, Maosong Sun, and Yang Liu. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10303–10315. Association for Computational Linguistics, 2023. 2

[19] Zhiheng Wang, Bo Peng, Lai Wei, and Chaochao Lu. The illusion of visual tool-use: A causal audit of thinking with images. *arXiv preprint arXiv:2608.06270*, 2026. 2

[20] Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. MMSearch-R1: Incentivizing LMMs to search. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2456–2487. Association for Computational Linguistics, 2026. 1, 2, 5, 6

[21] Shilin Yan, Jintao Tong, Hongwei Xue, Xiaojun Tang, Yangyang Wang, Kunyu Shi, Guannan Zhang, Ruixuan Li, and Yixiong Zou. Act wisely: Cultivating meta-cognitive tool use in agentic multimodal models. *arXiv preprint arXiv:2604.08545*, 2026. 2

[22] Huanjin Yao, Qixiang Yin, Min Yang, Ziwang Zhao, Yibo Wang, Haotian Luo, Jingyi Zhang, and Jiaxing Huang. MM-DeepResearch: A simple and effective multimodal agentic search baseline. In *Proceedings of the 43rd International Conference on Machine Learning*, pages 147736–147756. PMLR, 2026. 1, 2, 5, 6

[23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. 2, 5, 6

[24] Wenhao Zou, Xianglong Liu, Wendong Bi, Hanjie Wang, Simin Zhao, and Gong Zhi. CoBRA: Learning tool-use boundaries via counterfactual margins. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 28092–28110. Association for Computational Linguistics, 2026. 1, 2, 5, 6

Supplementary Material

Counterfactual Tool-Use Distillation
for Multimodal Question Answering

Pengcheng Xu (second author) et al.
University of California, Irvine

Contents

Page

A Technical Details **S2**

A.1 State collection and candidate construction . . . S2
A.2 Input serialization and gain prediction . . . S2
A.3 Answer scoring, sampling, and failures . . . S2
A.4 Resource ledger and feasibility . . . S3

B Experimental Details **S3**

B.1 Benchmarks, splits, and retrieval environments . . . S3
B.2 Controlled router comparisons . . . S4
B.3 Comparisons with complete search agents . . . S5
B.4 Task, resource, and routing metrics . . . S5
B.5 Backbones, optimization, and budgets . . . S6
B.6 Evaluation accounting . . . S6

C Additional Experimental Results **S7**

C.1 Additional benchmark results . . . S7
C.2 Additional benchmark metrics . . . S7
C.3 Independent gain diagnostics . . . S7
C.4 Policy-swap mechanistic analysis . . . S7
C.5 Matched evidence-state interventions . . . S9
C.6 Thresholds and sensitivity . . . S10
C.7 Supervision controls and repeatability . . . S10
C.8 End-to-end resources . . . S10
C.9 Qualitative cases and the one-step horizon . . . S11

D Theoretical Properties and Proofs **S16**

D.1 Shared-reference contrasts . . . S16
D.2 Sampling covariance . . . S16
D.3 Decision error and action recovery . . . S16

References **S17**

A. Technical Details

A.1. State collection and candidate construction

Backbone and tools. We use Qwen2.5-VL-7B-Instruct as the default instance of F_ϕ and use the text and image retrieval interfaces of MC-Search [8]. The same frozen backbone supplies the candidate generator G and the feature encoder. Retrieval uses a versioned knowledge-base snapshot; the retriever, argument schema, return limits, and cached responses are shared across routing methods. A tool call is identified by its registered type and canonical JSON parameters. Its cache key additionally includes the knowledge-base version, tool configuration, and hashes of referenced images.

Candidate generation. The generator receives the current state and the registered tool schemas. Its instruction is: *Propose at most two text-retrieval calls and at most two image-retrieval calls that address missing information in the question. Return a JSON list containing the tool type and arguments for each call. Use only the supplied schemas and available image indices. Do not answer the question.* Generation uses greedy decoding. The validator checks the registered tool type, required fields, argument types, and valid ranges. It removes duplicate calls after canonicalization, preserves the first occurrence, and retains at most two calls per type in generation order, giving $K_s \leq 4$. Malformed entries are discarded. An unparseable response produces an empty list.

Behavior policy and state sampling. For state collection, the frozen backbone selects an index from the validated candidate list using greedy decoding. The selection instruction is: *Select the single candidate most useful for answering the question from the current evidence. Return its list index. Return 0 when the available evidence is sufficient.* An invalid index terminates the trajectory. Each trajectory contains at most four executed calls and terminates on STOP. Tool responses update the evidence history in execution order.

Each task contributes its initial state and at most two subsequent decision states visited before the four-call limit. Subsequent states are ranked by the SHA-256 hash of the task identifier and decision depth, and the two smallest hashes are retained. Selection is independent of answer correctness, measured gains, and retrieval success. The candidate list at each retained state is saved and reused for all routing comparisons.

Data grouping. Task groups are formed before state collection. Records sharing an original task identity are linked; records sharing an input-image identity are also linked. The connected components define the groups assigned to training, development, and test sets. All derived states, candidate branches, and answer repetitions inherit the original group assignment. Reference answers are available only to the scoring procedure. Gold supporting-evidence identifiers and reference reasoning chains are excluded from state collection, candidate generation, and feature extraction.

A.2. Input serialization and gain prediction

The state serialization has four fields in a fixed order: question, indexed input images, chronological evidence history, and the resource budget at decision entry. Images retain their indices throughout the trajectory; newly observed images receive successive indices. Evidence entries retain the producing tool type and canonical arguments together with the returned content. The state encoder receives the images through the backbone's visual input interface.

Each call is serialized as its registered tool type followed by a JSON argument object with lexicographically sorted keys. Image references use the same indices as the state serialization and refer only to images available before the call. Both inputs use the backbone's chat template, and the feature is the hidden vector at the final non-padding input token before answer generation. Both feature vectors therefore have the same hidden dimension d.

The encoder is evaluated without dropout and without gradient updates. Call features are cached by canonical serialization; state features are shared across the candidates at a decision. The head uses GELU, $\sigma(v) = v\Phi(v)$ elementwise, where Φ is the standard Gaussian cumulative distribution function. Its only trainable parameters are those in Eq. (3) of the main paper. Model selection minimizes the development-set counterpart of Eq. (4) of the main paper, using the same task, state, and candidate weighting.

A.3. Answer scoring, sampling, and failures

Answer score. The answer model receives the serialized state followed by one additional-evidence slot. Its instruction is: *Answer the question using the current state and additional evidence. Return only a JSON object with a nonempty string field named answer. Do not call tools.* The reference branch leaves the evidence slot empty. Answer extraction precedes normalization: a missing answer field, an unparsable output, and an empty extracted answer each receive score zero. Valid answers and references are lowercased, stripped of punctuation and English articles, and split on whitespace.

Let N_p and N_y be the normalized prediction and reference token counts, and let O be their multiset overlap. Their F1 is $2O/(N_p + N_y)$ when both counts are positive; it is zero otherwise. The score V is the maximum F1 over the task's accepted references, including the singleton case.

Matched sampling. Each candidate is executed once, and its cached result is held fixed for all $R = 3$ answer repetitions. Repetition r uses the same seed ξ_r for the reference and candidate branches. Repetition seeds are independent. The answer prompt template, stochastic decoding configuration, and maximum answer length are shared by every branch and by final task evaluation. Final evaluation draws fresh seeds independent of all label-construction and local-comparison

Algorithm S1 Counterfactual supervision and gain regression

Require: Grouped training tasks; frozen F_ϕ, G; tool snapshot; scorer V; $R = 3$
Ensure: Fitted gain predictor g_θ

for each training task x **do**
  Initialize $\mathcal{S}_x \leftarrow \varnothing$; collect the initial and hash-selected prefix states
  for each sampled state s **do**
    Load its saved validated candidate list $\mathcal{T}_s$
    if $K_s > 0$ **then**
      Execute every t_j once; cache each result z_j
      Replace failed returns with the failure message
      for $r = 1, \ldots, R$ **do**
        Sample seed ξ_r; score $u_{0,r}$ from s
        for $j = 1, \ldots, K_s$ **do**
          Score $u_{j,r}$ from (s, z_j) using ξ_r
        end for
      end for
      if all branch measurements are available **then**
        Compute $\widehat{\mathbf{\Delta}}(s)$ by Eq. (2) of the main paper
        Extract pre-call features; add s to $\mathcal{S}_x$
      else
        Record the measurement failure
      end if
    end if
  end for
end for
Form $\mathcal{D}$ from tasks with nonempty $\mathcal{S}_x$
if $\mathcal{D} = \varnothing$ **then**
  Terminate with a measurement-coverage failure
end if
Fit θ by Eq. (4) of the main paper; select on development tasks

samples. The expectation in Eq. (2) of the main paper is over this answer sampler, conditional on the state and cached return. Shared seeds couple branch sampling, while the sampled token sequences can differ.

Tool failures. An unsuccessful attempt returns the fixed message `TOOL_FAILURE: no usable evidence.` This replacement applies to timeouts, empty responses, and responses that violate the registered schema. The answer model then completes the branch using the same prompt. Each attempt incurs its call charge and consumes its actual resources; the protocol performs no retry. These branches remain valid measurements, including when their gains are negative.

Measurement coverage. Missing cache records and interrupted inference jobs with no recoverable output are recorded as measurement failures. A state enters $\mathcal{S}_x$ only when all $(K_s + 1)R$ branch scores are available and $K_s > 0$. Tasks with no retained state are excluded from $\mathcal{D}$. Training terminates if $\mathcal{D}$ is empty. All tasks remain in final evaluation, with measurement coverage counted at both state and task level. Completed generations are always scored; empty and malformed answers receive zero, and incorrect answers receive their measured F1.

A.4. Resource ledger and feasibility

The entry budget $\mathbf{B}(s)$ tracks three resources componentwise: remaining tool attempts, model tokens, and tool-execution time. Model-token accounting includes processed input tokens, visual tokens, and generated output tokens in candidate generation, feature extraction, and answering. Tool-execution time is enforced through per-call timeouts; total wall-clock latency is measured separately. A failed attempt consumes one call and its actual token and time use.

Let $\mathbf{r}_G$ and $\mathbf{r}_E$ be the resources charged for candidate generation and feature extraction. Let $\overline{\mathbf{r}}_j$ bound the execution resources for t_j, and let $\overline{\mathbf{r}}_A$ reserve the final answer. The answer reserve covers the current context, the maximum permitted additional evidence, and the common answer-length limit. Tool adapters cap returned text and image content before it enters the answer prompt. After generation and encoding, feasibility is

$$\mathcal{F}(s) = \left\{ j : \overline{\mathbf{r}}_j + \overline{\mathbf{r}}_A \preceq \mathbf{B}(s) - \mathbf{r}_G - \mathbf{r}_E \right\}, \qquad \text{(S1)}$$

where $j \in \{1, \ldots, K_s\}$ and $\preceq$ denotes componentwise comparison. Returned evidence is charged through the answer's input-token reserve, avoiding double counting.

The entry budget must support a direct answer. Before candidate generation, the controller checks its generation cap while retaining the answer reserve. After validation, it checks the encoding cost using the actual input lengths while retaining the same reserve. Failure of either check triggers a direct answer. Empty candidate lists terminate before encoding. The resource ledger is updated after every operation, while the serialized state retains the budget recorded at decision entry. Training and deployment use this same state snapshot for gain prediction and branch answering.

The charge $c_j = 1$ in Eq. (5) of the main paper counts an attempted tool call; it is distinct from the resource vector. Candidate-generation and encoding costs are shared by the actions compared after routing and included in the end-to-end resource log. Offline branch construction is recorded separately as a training expense. Online execution follows the entry budget $\mathbf{B}(s)$.

B. Experimental Details

B.1. Benchmarks, splits, and retrieval environments

In-domain development. MC-Search [8] supplies the training and development tasks. Its public 3,333-record release is distributed under a single split named `train`. The train/development/test split assigns the connected task–image groups defined in Sec. A.1 in a 60/20/20 ratio, using split seed 42. Groups used in the preliminary study are

Algorithm S2 Resource-constrained one-step answering

Require: Entry state s; frozen F_ϕ, G; head g_θ; fixed λ
Ensure: Selected action, final answer, resource log
Initialize the ledger from $\mathbf{B}(s)$; reserve the answer
if candidate generation exceeds the unreserved budget **then**
    return STOP, answer from s, resource log
end if
Generate and validate $\mathcal{T}_s$; charge $\mathbf{r}_G$
if $K_s = 0$ **then**
    return STOP, answer from s, resource log
end if
if feature encoding exceeds the unreserved budget **then**
    return STOP, answer from s, resource log
end if
Extract features from s and $\mathcal{T}_s$; charge $\mathbf{r}_E$
Construct $\mathcal{F}(s)$ by Eq. (S1)
if $\mathcal{F}(s) = \varnothing$ **then**
    return STOP, answer from s, resource log
end if
Predict $g_\theta(s, t_j)$ for $j \in \mathcal{F}(s)$
if any feasible prediction is nonfinite **then**
    return STOP, answer from s, resource log
end if
Select $\hat{a}$ by Eq. (6) of the main paper and the fixed tie rule
if $\hat{a} = 0$ **then**
    return STOP, answer from s, resource log
end if
Execute $t_{\hat{a}}$ once; charge the attempt
Set $z_{\hat{a}}$ to the result; replace a failed return
return $t_{\hat{a}}$, answer from $(s, z_{\hat{a}})$, resource log

reserved for development before allocating the remaining groups. The grouped test split contains 627 tasks: 245 image-chain, 44 multi-image-fork, 32 text-chain, 128 image–text-fork, and 178 text-only tasks. All prefixes and counterfactual branches inherit their task's partition. Results include the five reasoning-graph categories and the task average.

The closed-corpus environment uses MC-Search's text and image retrieval interfaces, with one highest-ranked evidence item per call. Every method uses the same retriever and knowledge-base snapshot, identified by file hashes and document/image counts. For CTD, gold evidence and reference chains are restricted to scoring.

External search benchmarks. The shared search-agent suite follows the subsets used by MMSearch-R1 [13]: FVQA-test (1,800), InfoSeek (2,000 sampled test questions), MMSearch (171 image-containing questions), SimpleVQA (1,013 English questions), and LiveVQA (3,602). All methods in Table 2 of the main paper, including MM-DeepResearch and WebWatcher, use these same evaluation subsets. FVQA-test refers to the collection introduced with MMSearch-R1. InfoSeek comparisons use identical released example identifiers. All external datasets are evaluation-only for the MC-Search-trained CTD head. Overlap checks compare task identifiers, normalized questions, and image hashes against CTD training groups.

MMSearch additionally receives a full 300-question evaluation using its native end-to-end F1 scorer [6]. This score is distinct from its weighted aggregate of end-to-end answering, requery, reranking, and summarization. The 171-question subset receives both F1 and judge accuracy. InfoSeek additionally receives its released answer-type-aware scorer [1]; SimpleVQA retains its correct/incorrect/not-attempted breakdown [2]. These native scores remain separate from the common judge metric. LiveVQA uses a pinned release and retrieval date [4].

Harder evaluation. BrowseComp-VL [5] and MAT-Search (150 questions) [7] assess deeper acquisition. MAT-Search receives native F1 and exact match; the common-environment answers also receive judge accuracy in Table S5. BrowseComp-VL uses all 300 questions in each evaluation. HR-MMSearch (305 questions) [3] is a separate high-resolution extension: it adds at most two crop candidates to the two text and two image-search candidates, giving $K_s \leq 6$. Every compared method receives the same crop interface and image-processing limits. Crop-augmented heads use separate training-only branch measurements and are evaluated separately from the search-only $K_s \leq 4$ setting. The benchmark's original easy/hard partitions remain fixed.

Web environment. Web comparisons share one search provider, query schema, cache, and retrieval window. Following MMSearch-R1's return structure, image search exposes up to five title–thumbnail pairs and text search up to five retrieved pages. A common frozen Qwen2.5-VL-7B helper compresses text returns to at most 2,048 tokens; its calls, tokens, and latency enter the ledger. The same canonical request receives the same cached response across methods. Timing distinguishes cached replay from uncached retrieval. The evaluation environment is specified by the provider version, retrieval dates, response hashes, prompts, and helper checkpoint revision.

B.2. Controlled router comparisons

All routers receive identical states and the same saved candidate list, with at most one executed candidate before the common final-answer sampler. Initial-state task evaluation is primary; collected-prefix diagnostics remain separate and include prefix costs. The following controls are experiment-specific implementations unless a released method is explicitly identified.

Training-free and imitation controls. Direct answering executes no tool. Fixed-first retrieval takes the first feasible candidate. A random policy samples a feasible candidate with a development-selected invocation probability. These complement the fixed-retrieval comparisons in MC-Search and MMSearch-R1 [8, 13]. A frozen model selecting a candidate index supplies a common selector for uncertainty gates. Confidence gating thresholds the length-normalized log prob-

ability of a direct-answer draft; entropy gating thresholds its mean next-token entropy. Their extra draft computation is charged. Success-only imitation predicts the behavior action from states on correctly answered collected trajectories, paralleling success-trajectory supervision in Search-Align [8]; it shares CTD's frozen features and head capacity, and its retained-label count is reported.

Supervision controls. The Toolformer-inspired proxy [12] regresses the reduction in reference-answer token loss after a candidate call using the same cached branches, adapting its likelihood signal to the frozen-backbone router. The CoBRA-inspired control [16] fits a binary gate with target $\mathbf{1}[\max_j(\widehat{\Delta}_j - \lambda c_j) > 0]$ and a separate absolute-value candidate ranker. The gate is fitted at its development-selected operating point, and the total trainable parameter budget matches CTD within 1%. This control tests binary boundary supervision. An absolute-value regressor predicts all branch scores, including STOP, and uses the same cost-adjusted decision rule, head width, and counterfactual observations. Its output uses a sigmoid for scores in $[0, 1]$ and a fixed STOP call embedding. Learned controls share training tasks, branch-generation allowances, and development-search budgets. The Qwen2.5-VL-7B comparison in Table S11(a) imposes common offline model-token and tool-attempt limits. Actual usage includes state collection, candidate and branch generation, feature extraction, head fitting, and development selection. Retained-label and parameter counts quantify supervision and model capacity.

B.3. Comparisons with complete search agents

The comparisons include ReAct and fixed image-then-text RAG, following MMSearch-R1 [13, 15]. Search-Align [8] is the domain-specific trained reference on MC-Search. The web suite compares MMSearch-R1 [13], MM-DeepResearch [14], Visual-ARFT [7], WebWatcher [5], and SenseNova-MARS [3]. Released 7B Qwen2.5-VL agents are grouped with CTD on Qwen2.5-VL-7B, and 8B Qwen3-VL agents with CTD on Qwen3-VL-8B. Search-Align additionally supports the InternVL3.5-8B comparison. Its original process-supervised recipe uses training-set gold reasoning chains and is marked as an extra-supervision reference. Table 1 of the main paper uses our shared-data success-only imitation control.

The shared protocol enforces common search interfaces, return limits, and context limits. Table 2 of the main paper fixes the call cap at $B = 2$ for every method; Table S2 additionally evaluates $B = 4$ and CTD at $B = 1$. The two-call panel includes fixed image-then-text RAG; image search is restricted to the initial turn in the MMSearch-R1-aligned panel. Methods with larger native tool inventories are labeled as restricted-tool adaptations. CTD regenerates candidates and reapplies its one-step head after each observation, stopping on STOP or budget exhaustion.

Released-agent comparisons evaluate complete systems with checkpoint-specific training data and backbone updates under the matched evaluation protocol. Table S11 compares offline training budgets among frozen-backbone learned routers on Qwen2.5-VL-7B. Published scores from native datasets, scorers, and environments appear in separate reference blocks.

B.4. Task, resource, and routing metrics

F1 follows Sec. A.3; improvement over direct answering uses identical tasks and answer samplers. Search-agent accuracy uses the released GPT-4o judge prompt with a pinned judge version and deterministic judging. Each answer is scored once using the question, image, reference, and extracted answer, without a method identifier. Per-dataset metrics are reported separately. Macro accuracy averages judge accuracies on the identical five-dataset suite.

For N tasks, attempted-call counts n_i, and a shared cap B, Calls is the primary cost metric. The supplementary $B = 2$ comparisons in Table S5 also give CallRate and SR:

$$\text{Calls} = \frac{1}{N}\sum_i n_i,$$
$$\text{CallRate} = \frac{1}{N}\sum_i \mathbf{1}[n_i > 0], \qquad \text{SR} = \frac{1}{NB}\sum_i n_i. \tag{S2}$$

SR follows the normalized search-count definition in MMSearch-R1's Table 1. Both rates are reported as percentages. Failed attempts consume budget. Resource accounting includes candidate generation, feature extraction, helpers, direct-answer drafts, and final answering, with text/visual input and output tokens separated. Tables S10 and S11 report offline supervision costs. Resource profiling in Table S13 uses Qwen2.5-VL-7B on MC-Search and MMSearch-171, reporting token counts, median/P95 latency, and peak GPU memory. Cached replay and uncached retrieval are timed separately.

Routing diagnostics use fresh counterfactual measurements on held-out MC-Search states. MAE and ten equal-width signed-gain calibration bins retain the task–state–candidate weighting of Eq. (4) of the main paper; bin counts are shown. Gains are signed F1 differences in $[-1, 1]$. For the local reference gap, one independent sample set selects the highest estimated net-value action among STOP and feasible candidates, and another estimates its net-value difference from the router's action. Both sets use three answer draws per branch, independent of training labels and task scoring. Sampling variability can make this estimated gap negative. MC-Search's released HPS and RD measures are computed from the $B = 2$ grouped-split trajectories in Table S5.

Setting	Value
Trainable head	$2d \to 256 \to 1$, GELU, tanh output
Backbone / head precision	BF16 / FP32
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.999)$
Learning rate / weight decay	3×10^{-4} / 10^{-2}
Batch / maximum epochs	32 tasks / 20
Selection / early stopping	Development MSE / patience 3
Gradient clipping	Global norm 1.0
Replication seeds	0, 1, 2
Candidates / paired draws	$K \leq 4$ / $R = 3$
Candidate decoding	Greedy, at most 512 output tokens
Final-answer decoding	$T = 0.2$, top-$p = 0.9$, at most 512 tokens
Context / text tool return	16,384 / 2,048 tokens
Per-call timeout / retries	30 seconds / 0
Primary / multi-step call cap	1 / 2 and 4

Table S1. CTD configuration. Task batches preserve the nested task, state, and candidate averaging of Eq. (4) of the main paper.

B.5. Backbones, optimization, and budgets

The model identifiers are `Qwen/Qwen2.5-VL-7B-Instruct`, `Qwen/Qwen3-VL-8B-Instruct`, and `OpenGVLab/InternVL3_5-8B` [9–11]. Each backbone supplies its frozen answer model, generator, and feature encoder; labels and heads are fitted separately. Qwen2.5-VL covers widely used 7B search agents, Qwen3-VL covers newer 8B agents, and InternVL tests another model family. Within each backbone comparison, the processor, chat template, and image limits are identical. Core inputs are resized to a maximum side length of 1,024 pixels before the native processor. The high-resolution extension preserves source images for cropping and applies this limit to each processed view. Actual visual-token counts are logged.

Table S1 summarizes the CTD configuration. The hardware allocation is eight NVIDIA H100 GPUs. GPU memory capacity, software versions, and checkpoint commit hashes are recorded per run.

The configuration allocates eight GPUs to feature extraction and branch generation, and one GPU to cached-feature head fitting. Resource profiling uses one H100 at batch size one with a shared serving configuration. All learned router controls use the optimizer configuration in Table S1, with checkpoints selected by their respective supervised objectives on development data. Each minibatch samples 32 tasks uniformly and averages over their retained states and candidates, preserving the equal-task objective.

The call charge is $c_j = 1$, with $\lambda \in \{0, 0.01, 0.025, 0.05, 0.1, 0.2, 0.4\}$. The primary operating point maximizes development F1 subject to at most 0.5 mean calls per initial state, including always-STOP as a feasible option; ties prefer fewer calls and then larger λ. For CTD and absolute-value regression, development selection gives $\lambda = 0.05$ on each backbone. External evaluation reuses this operating point. Table S9 reports all seven test operating points for CTD and absolute-value regression, reusing each fitted head. The ledger in Sec. A.4 reserves final answering before tool execution. The total processed-model-token caps are 65,536 per one-step task and 262,144 per multi-step task, including auxiliary computations; tool-time budgets are $30B$ seconds. Sensitivity runs in Table S10 use Qwen2.5-VL-7B on MC-Search only: $K \in \{2, 4, 8\}$, label draws $R \in \{1, 3, 5\}$, and head width $m \in \{128, 256, 512\}$ vary one factor at a time. The default plus six nondefault configurations each use three training seeds. Candidate-count variants retain at most $K/2$ calls per retrieval type in the same validated generation order; $K = 8$ extends the generator instruction to four calls per type. They are separate from the core $K \leq 4$ protocol. Test answer sampling remains at three draws for every variant.

Replication and uncertainty. For the controlled-router comparison on MC-Search, learned routers use training seeds 0, 1, and 2, with three fresh answer draws per fitted model. Table S12 reports per-seed scores, sample standard deviations, and CTD-minus-absolute-value differences with paired 95% intervals from 2,000 bootstrap resamples of original task–image groups, keeping associated states and answer repetitions together. The search-agent result tables report three-evaluation point estimates. Table S11(b) evaluates greedy decoding on Qwen2.5-VL-7B using CTD and control supervision recollected under that sampler. All seeds evaluate the same test tasks. Malformed answers score zero and all test tasks remain in the denominator. Training measurement exclusions and coverage follow Sec. A.3.

B.6. Evaluation accounting

Task evaluation uses the final answers produced by the selected actions. Initial-state evaluation and evaluation after a collected prefix are kept separate. The latter records prefix-acquisition resources in addition to the router's incremental resources. Confidence intervals resample original task groups, keeping their states and repetitions together. Local reference actions are estimated from independent branch samples and evaluated with fresh answer draws.

For N fully measured states with mean candidate count $\bar{K}$, label construction requires $N(\bar{K} + 1)R$ answer generations and $N\bar{K}$ tool executions, excluding state collection and candidate generation. At inference, the router requires one state encoding, the uncached call encodings, and $|\mathcal{F}(s)|$ head evaluations. Resource records include these backbone computations, final answering, and failed attempts.

C. Additional Experimental Results

C.1. Additional benchmark results

Matched comparisons. Table S2 uses the full five-dataset search-agent suite and a shared retrieval cache, helper, judge, and resource ledger. Each released checkpoint is evaluated under the stated cap; restricted-tool agents are identified explicitly. Training data are checkpoint-specific, as explained in Sec. B.3. Macro accuracy weights the five datasets equally; Calls averages per-dataset mean attempts with the same weights. The MC-Search comparison in Table S3 reports the original evaluation protocol and our grouped split in separate blocks.

Reasoning-graph coverage. At $B = 2$, CTD improves over Search-Align in all five graph categories for both supported backbone families in Table S3. The gains range from 2.91 to 4.15 F1 points on Qwen2.5-VL-7B and from 3.04 to 4.66 points on InternVL3.5-8B. On Qwen3-VL-8B, CTD exceeds ReAct by 8.80–9.59 points across the five categories. These within-split comparisons cover both chain and fork structures, with category scores and the task-weighted aggregate reported separately.

Evaluation subsets. Table 2 of the main paper summarizes the $B = 2$ results from Table S2 on the same five evaluation subsets. BrowseComp-VL is evaluated with the harder benchmarks in Table S4(c), whose Calls column averages that separate four-dataset suite. Published native references appear in the labeled blocks of Tables S3 and S4(a,b).

Tool use. MMSearch-R1 reports native budget-normalized macro search ratios of 100.0, 51.9, and 67.1% for its Qwen2.5-VL-7B fixed-RAG, ReAct, and MMSearch-R1 rows, respectively [13]. Our common-environment evaluation records actual call attempts, including failures. At $B = 2$, CTD uses 0.84, 0.88, and 0.91 mean calls on the five-dataset suite, versus 1.72, 1.76, and 1.92 for the highest-macro-accuracy baselines in the corresponding families.

Harder benchmarks and native metrics. Table S4 reports MAT-Search's native F1/EM [7] and the judge accuracies from SenseNova-MARS [3] in separate panels. Under the native MAT-Search protocol, CTD obtains 67.95 F1 and 58.67% EM on Qwen2.5-VL-7B. MAT-Search has 75 simple and 75 hard questions. Full MMSearch uses end-to-end F1 on 300 questions, distinct from its weighted multi-stage aggregate and the 171-question judge score. The common-environment HR-MMSearch comparison uses a shared crop interface, with labels and costs measured in this crop-enabled configuration. Calls is the four-dataset mean in panel (c).

C.2. Additional benchmark metrics

Table S5 applies native metrics to the outputs evaluated by the judge in Table 2 of the main paper. MAT uses the answers from Table S4(c). Web CallRate and SR use the same five-dataset weights as Calls. On grouped MC-Search, task-weighted overall F1 is 47.18/52.43/48.60 for CTD on Qwen2.5-VL/Qwen3-VL/InternVL3.5, with 0.88/0.91/0.93 calls. The mean reference-chain length is 3.70 steps. CTD's lower HPS and more negative RD indicate reduced reference-chain coverage alongside improved final-answer quality.

C.3. Independent gain diagnostics

Table S6 separates initial and collected-prefix states. Independent sample sets A and B each contain three paired answer draws per branch. Set A selects the highest measured net-value action among STOP and feasible calls; B evaluates its gap from each router's action. All routers share these banks and the development-selected cost penalty. MAE and calibration use set A, independently of head training. Absolute-value regression implies a gain by subtracting its predicted STOP score; a binary gate predicts whether to call a tool.

Predictions in $[-1, 1]$ fall into ten equal-width bins. With normalized task–state–candidate weights from Eq. (4) of the main paper, bin weights w_b, and weighted predicted/measured gains $\bar{g}_b, \bar{\Delta}_b$, we report $\mathrm{SCal}_{10} = 100 \sum_b w_b |\bar{g}_b - \bar{\Delta}_b|$. MAE is also in F1 points. Detailed bins use training seed 0 and unweighted candidate counts, with empty-bin means left undefined. The aggregate SCal values average three training seeds. CTD reduces MAE from 11.86/11.02/11.51 to 9.42/8.67/9.18 relative to absolute-value regression, with smaller reference gaps at both initial and prefix states.

C.4. Policy-swap mechanistic analysis

Gate and ranker swaps. We analyze 24,000 decision states with four feasible candidates, a one-call budget, and uniform charge $c_j = 1$ at $\lambda = 0.05$. For router $m \in \{C, A\}$, denoting CTD and absolute-value regression (AV), let $\widehat{\Delta}_j^m(s)$ be its predicted F1 gain. AV subtracts its predicted STOP score from each predicted branch score. The stopping gate and conditional ranker are

$$b_m(s) = \mathbf{1}\left[\max_{j \in \mathcal{F}(s)} \widehat{\Delta}_j^m(s) > \lambda\right], \qquad r_m(s) = \arg\max_{j \in \mathcal{F}(s)} \widehat{\Delta}_j^m(s). \tag{S3}$$

Hybrid policy π_{gr} uses router g's gate and router r's ranker. It stops when $b_g(s) = 0$ and otherwise executes $r_r(s)$, including when router r alone would stop. This conditional execution separates ranking from the decision to acquire evidence. Ties follow Eq. (6) of the main paper. All four policies share the candidate outcomes and direct-answer reference. Their mean F1 scores are 42.73 (CC), 41.76 (CA), 42.08 (AC), and 41.37 (AA).

Exact attribution. Let $v_{gr}(s)$ denote the answer-F1 improvement of π_{gr} over direct answering, before subtracting

Table S2. **Matched search-agent comparison.** Scores are judge accuracies (%) in the common environment. B is the call cap; Calls is the macro mean of actual attempts. CTD uses its one-step policy at $B = 1$ and its greedy extension at $B = 2, 4$. Bold and underlining mark the best and second-best values within each backbone and budget; single-method groups are unranked.

Method	B	FVQA-test 1,800	InfoSeek 2,000	MMSearch 171	SimpleVQA 1,013	LiveVQA 3,602	Macro↑	Calls↓
Qwen2.5-VL-7B family								
Fixed image–text RAG [13]	2	54.28	52.17	51.46	53.27	48.34	51.90	1.91
ReAct [15]	2	50.89	47.83	45.61	50.64	44.43	47.88	<u>1.18</u>
Visual-ARFT (restricted) [7]	2	56.31	50.78	49.71	54.13	47.51	51.69	1.43
MMSearch-R1 [13]	2	59.43	55.68	54.97	58.01	51.59	55.94	1.39
WebWatcher (restricted) [5]	2	54.74	52.63	52.24	55.81	50.93	53.27	1.66
MM-DeepResearch (restricted) [14]	2	<u>60.04</u>	<u>58.07</u>	<u>59.45</u>	<u>59.62</u>	<u>54.82</u>	<u>58.40</u>	1.72
CTD	1	60.74	58.12	58.09	60.88	55.22	58.61	0.47
CTD	2	**63.09**	**60.77**	**61.99**	**62.88**	**57.91**	**61.33**	**0.84**
MMSearch-R1 [13]	4	61.46	58.72	58.67	59.86	54.04	58.55	<u>2.34</u>
MM-DeepResearch (restricted) [14]	4	<u>62.35</u>	<u>60.67</u>	<u>62.57</u>	<u>61.50</u>	<u>57.13</u>	<u>60.84</u>	2.78
CTD	4	**65.31**	**63.67**	**64.72**	**65.48**	**60.78**	**63.99**	**1.31**
Qwen3-VL-8B family								
Fixed image–text RAG [13]	2	62.41	56.73	57.12	63.21	54.32	58.76	1.93
ReAct [15]	2	56.19	53.42	50.49	57.52	48.74	53.27	<u>1.22</u>
SenseNova-MARS (restricted) [3]	2	68.33	63.62	66.28	<u>69.27</u>	58.63	65.23	1.55
MM-DeepResearch (restricted) [14]	2	<u>69.61</u>	<u>69.17</u>	<u>67.25</u>	67.36	<u>61.82</u>	<u>67.04</u>	1.76
CTD	1	69.94	67.58	68.03	69.86	62.40	67.56	0.46
CTD	2	**72.37**	**70.83**	**70.57**	**72.92**	**64.98**	**70.33**	**0.88**
SenseNova-MARS (restricted) [3]	4	70.93	67.82	69.79	<u>71.27</u>	61.06	68.17	<u>2.54</u>
MM-DeepResearch (restricted) [14]	4	<u>72.65</u>	<u>72.27</u>	<u>71.35</u>	70.09	<u>65.24</u>	<u>70.32</u>	2.87
CTD	4	**75.06**	**74.12**	**73.49**	**75.68**	**67.86**	**73.24**	**1.40**
InternVL3.5-8B family								
Fixed image–text RAG [13]	2	<u>57.15</u>	<u>53.77</u>	<u>53.80</u>	<u>56.24</u>	<u>51.10</u>	<u>54.41</u>	1.92
ReAct [15]	2	52.59	49.83	47.95	52.71	46.11	49.84	<u>1.20</u>
CTD	1	59.61	56.98	56.73	59.23	53.19	57.15	0.49
CTD	2	**62.46**	**59.53**	**60.23**	**62.36**	**56.23**	**60.16**	**0.91**
ReAct [15]	4	<u>56.13</u>	<u>53.67</u>	<u>51.85</u>	<u>55.31</u>	<u>49.09</u>	<u>53.21</u>	<u>2.41</u>
CTD	4	**65.11**	**62.18**	**62.96**	**65.05**	**58.82**	**62.83**	**1.46**

call cost. The symmetric decomposition is

$$\begin{aligned} \phi_{\text{stop}}(s) &= \tfrac{1}{2}\big[(v_{CC} - v_{AC}) + (v_{CA} - v_{AA})\big], \\ \phi_{\text{rank}}(s) &= \tfrac{1}{2}\big[(v_{CC} - v_{CA}) + (v_{AC} - v_{AA})\big], \\ v_{CC}(s) - v_{AA}(s) &= \phi_{\text{stop}}(s) + \phi_{\text{rank}}(s). \end{aligned} \tag{S4}$$

Each component averages its effect under both settings of the other, dividing their interaction equally. The two contributions sum exactly to the CTD–AV difference for every state. All aggregate scores and contributions are unweighted state means.

Landscape coordinates. Measured candidate gains $\widetilde{\Delta}_j(s)$ define the coordinates, and separate branch-score estimates evaluate the hybrid outcomes:

$$\begin{aligned} G(s) &= \max_{j \in \mathcal{F}(s)} \widetilde{\Delta}_j(s) - \lambda, \\ M(s) &= \widetilde{\Delta}_{(1)}(s) - \widetilde{\Delta}_{(2)}(s). \end{aligned} \tag{S5}$$

The subscripts (1) and (2) index the two largest gains. A negative G places every candidate below the call penalty; a larger M indicates greater separation between the best two candidates. Figure 5 of the main paper expresses both coordinates in F1 percentage points.

Smoothing and regional results. We bin states on an 80×54 grid over $G \in [-35, 40]$ and $M \in [0, 39]$ points. A Gaussian kernel with a standard deviation of two bins smooths outcome sums and state counts separately. Their ratio gives the local mean advantage. Cells with smoothed counts at most 1.9 are masked; the displayed range $M \in [0, 26]$ contains all unmasked cells. The color scale spans $[-4.8, 4.8]$ points. Table S7 reports unsmoothed regional means: stopping supplies 96.5% of the improvement for $G < -10$, while ranking supplies 72.3% for $G > 0$.

Table S3. **MC-Search results by reasoning graph.** Scores are answer F1 (0–100). The upper block follows the original MC-Search evaluation [8]; the lower block uses our grouped split and shared corpus at $B = 2$. Search-Align receives gold-chain process supervision. Bold and underlining mark the best and second-best values within each protocol and backbone.

Method / backbone	B	Image chain	Multi-image fork	Text chain	Image–text fork	Text only
Published native protocol; original evaluation partition and knowledge base						
Agentic / Qwen2.5-VL-7B [8]	native	<u>26.30</u>	<u>26.13</u>	<u>31.47</u>	<u>22.94</u>	<u>23.54</u>
Search-Align / Qwen2.5-VL-7B [8]	native	**45.70**	**38.01**	**45.07**	**32.73**	**37.54**
Agentic / InternVL3.5-8B [8]	native	<u>39.11</u>	<u>33.08</u>	<u>38.34</u>	<u>29.74</u>	<u>34.82</u>
Search-Align / InternVL3.5-8B [8]	native	**42.27**	**36.53**	**40.87**	**30.72**	**38.45**
Common corpus and grouped split; matched multi-step evaluation						
ReAct / Qwen2.5-VL-7B [15]	2	38.42	35.19	42.21	33.38	39.80
Search-Align / Qwen2.5-VL-7B [8]	2	<u>45.83</u>	<u>42.92</u>	<u>46.58</u>	<u>39.26</u>	<u>43.97</u>
CTD / Qwen2.5-VL-7B	2	**48.74**	**46.30**	**50.73**	**42.91**	**47.68**
ReAct / Qwen3-VL-8B [15]	2	<u>45.28</u>	<u>42.11</u>	<u>46.86</u>	<u>38.17</u>	<u>43.58</u>
CTD / Qwen3-VL-8B	2	**54.22**	**50.91**	**55.87**	**47.66**	**53.17**
ReAct / InternVL3.5-8B [15]	2	42.16	38.23	43.89	36.39	41.20
Search-Align / InternVL3.5-8B [8]	2	<u>47.64</u>	<u>43.10</u>	<u>46.91</u>	<u>40.22</u>	<u>45.59</u>
CTD / InternVL3.5-8B	2	**50.68**	**47.01**	**51.57**	**43.74**	**49.10**

Table S7. **Regional attribution of the CTD–AV advantage.** Total, Stop, and Rank are mean F1 contributions in percentage points.

Region	States	Total	Stop	Rank
All states	24,000	1.36	0.52	0.84
$G < -10$	8,571	0.66	0.64	0.02
$G > 0$	9,121	2.42	0.67	1.75

C.5. Matched evidence-state interventions

Intervention design. The analysis contains 480 task groups, each represented at depths 0, 1, and 2, giving 1,440 parent states. Each parent is evaluated under four evidence conditions: s^{miss}, s^{part}, s^{full}, and s^{irr}. The first three progressively supply answer-relevant information; the fourth adds irrelevant evidence to the missing-information state. The partial, sufficient, and irrelevant insertions each contain 128 tokens. The question, input images, candidate call, cached return, and resource budget remain fixed within each parent.

Paired gain measurement. For intervention a, the direct and tool branches have expected scores

$$\begin{aligned} u_t^a &= \mathbb{E}_\xi V(F_\phi(s^a, z_t; \xi), y), \\ u_0^a &= \mathbb{E}_\xi V(F_\phi(s^a, \varnothing; \xi), y), \\ \Delta^a(s,t) &= 100(u_t^a - u_0^a). \end{aligned} \tag{S6}$$

We estimate each gain from eight paired answer draws. The pre-execution prediction is $\widehat{\Delta}^a(s,t) = 100 g_\theta(s^a, t)$; reported predictions average three runs. Call rates threshold these averaged predictions at five F1 points. Table S8 averages equally over parent states.

Table S8. **State-conditioned gains and call rates.** Measured and predicted gains are in F1 points; Calls is the percentage of states in which CTD selects the fixed call.

Evidence	Measured	Predicted	Calls (%)
Missing	10.23	9.87	71.67
Partial	4.13	4.38	47.22
Sufficient	−0.32	0.40	5.14
Irrelevant	10.12	9.62	68.89

Paired sufficiency contrasts. For each parent, we compute

$$\begin{aligned} D_{\text{rel}}(s,t) &= \Delta^{\text{full}}(s,t) - \Delta^{\text{miss}}(s,t), \\ D_{\text{irr}}(s,t) &= \Delta^{\text{irr}}(s,t) - \Delta^{\text{miss}}(s,t), \\ \widehat{D}_{\text{rel}}(s,t) &= \widehat{\Delta}^{\text{full}}(s,t) - \widehat{\Delta}^{\text{miss}}(s,t). \end{aligned} \tag{S7}$$

The predicted irrelevant-evidence contrast $\widehat{D}_{\text{irr}}$ is defined analogously. Relevant evidence reduces the mean measured gain by 10.55 points and the predicted gain by 9.47 points. Irrelevant evidence produces changes of -0.11 and -0.25 points, respectively. The relevant-minus-irrelevant contrasts are therefore -10.44 measured and -9.22 predicted points. CTD's call rate falls from 71.67% to 5.14% with sufficient evidence and remains 68.89% under irrelevant insertion.

Response-field construction. Figure 7 of the main paper places measured gain on the horizontal axis and predicted gain on the vertical axis. Each condition uses a 154×154 histogram over $[-24, 39]^2$ F1 points, smoothed with a Gaussian kernel of standard deviation 3.3 bins. Contours lie at 17%, 38%, and 65% of each condition's peak density. Blue, purple, and teal indicate missing, partial, and sufficient evidence; dashed gray contours show the irrelevant control. To summarize paired changes, we divide the 5th–95th percentile

Table S4. **Harder benchmarks and native metrics.** Panels (a,b) use the corresponding native protocols; panel (c) uses the common environment. Scores use a 0–100 scale, with F1/EM and judge accuracy reported separately. Bold and underlining mark the best and second-best values within each panel and backbone, including ties. HR-MMSearch uses the crop-enabled configuration.

(a) Native MAT-Search, 150 questions; Visual-ARFT [7], Table 1

	Simple		Hard		All	
Method / backbone	F1↑	EM↑	F1↑	EM↑	F1↑	EM↑
Direct / Qwen2.5-VL-7B [7]	67.40	61.33	39.59	32.00	53.49	46.67
Visual-ARFT / Qwen2.5-VL-7B [7]	<u>71.78</u>	<u>66.67</u>	<u>55.77</u>	<u>44.00</u>	<u>63.77</u>	<u>55.33</u>
CTD / Qwen2.5-VL-7B	**76.26**	**69.33**	**59.64**	**48.00**	**67.95**	**58.67**

(b) Native judge accuracy; SenseNova-MARS [3], Table 1

Method / backbone	HR-MMSearch (305)	MAT-Search (150)
Visual-ARFT-7B [7]	**24.92**	**74.00**
MMSearch-R1-7B [13]	<u>20.33</u>	**74.00**
SenseNova-MARS-8B [3]	<u>41.64</u>	<u>84.67</u>
CTD / Qwen3-VL-8B	**46.89**	**86.67**

(c) Common environment, $B = 2$; native end-to-end F1 and separately scored judge accuracy

Method / backbone	MMSearch 300 / F1↑	BrowseComp -VL / Acc.↑	MAT-Search (150) F1↑	MAT-Search (150) EM↑	HR-MMSearch 305 / Acc.↑	Calls↓
ReAct / Qwen2.5-VL-7B [15]	44.36	23.78	59.31	52.00	18.69	<u>1.22</u>
Visual-ARFT-7B (restricted) [7]	48.52	27.00	66.20	56.00	25.14	1.36
MMSearch-R1-7B [13]	55.67	28.11	64.57	56.00	23.28	1.42
WebWatcher-7B (restricted) [5]	50.81	27.22	62.39	53.33	22.62	1.64
MM-DeepResearch-7B (restricted) [14]	<u>57.06</u>	<u>29.67</u>	<u>68.13</u>	<u>60.67</u>	<u>30.82</u>	1.72
CTD / Qwen2.5-VL-7B	**60.28**	**32.33**	**70.06**	**63.33**	**34.64**	**0.88**
SenseNova-MARS-8B (restricted) [3]	65.72	39.22	74.31	65.33	<u>43.17</u>	<u>1.62</u>
MM-DeepResearch-8B (restricted) [14]	<u>66.81</u>	<u>40.89</u>	<u>75.62</u>	<u>68.67</u>	42.08	1.73
CTD / Qwen3-VL-8B	**70.34**	**42.78**	**78.09**	**71.33**	**47.43**	**0.92**
ReAct / InternVL3.5-8B [15]	<u>48.79</u>	<u>25.67</u>	<u>61.81</u>	<u>54.00</u>	<u>24.48</u>	<u>1.24</u>
CTD / InternVL3.5-8B	**58.92**	**35.22**	**69.49**	**61.33**	**36.28**	**0.95**

range of missing-state measured gains into eight quantile intervals. Within each interval, arrows connect the mean missing, partial, and sufficient coordinates. The contours use all parent states.

C.6. Thresholds and sensitivity

Table S9 varies the penalty without refitting: $\lambda = 0.05$ reproduces the primary operating points, while $\lambda = 0.4$ yields direct answering with zero calls. Table S10 varies one training factor while keeping evaluation sampling fixed. Increasing K from 4 to 8 adds 0.35 F1 points and raises offline tokens from 312.6 to 489.7 million. Increasing R from 3 to 5 adds 0.21 points at 470.1 million tokens; doubling head width adds 0.03 points.

C.7. Supervision controls and repeatability

Table S11(a) limits offline usage to 180 million tokens and 12,000 tool attempts, including failed attempts and all supervision preparation; GPU-hours include head fitting. Retained-label counts can differ. CTD reaches 42.53 ± 0.22 F1 versus 41.09 ± 0.24 for absolute-value regression. Panel (b) uses sampler-specific labels on shared training tasks and tool returns; greedy decoding gives 44.19 ± 0.20 versus 42.76 ± 0.23. Its stochastic rows reuse Table S12. CTD's three-training-seed F1 standard deviations are 0.21/0.24/0.18 points. Paired F1-gain intervals exclude zero on every backbone, while call-count difference intervals include zero.

C.8. End-to-end resources

Table S13 charges auxiliary computation and failures. CTD's cached one-step median latency is similar to absolute-value regression (5.92 versus 5.97 seconds). On MMSearch-171, uncached median/P95 latency is 16.84/33.71 seconds versus 25.63/50.92 for MM-DeepResearch. Fixed RAG is faster in cached replay (9.44 versus 11.09 seconds at the median).

Table S5. **Additional benchmark metrics.** Common environment, $B = 2$. Panel (a) rescores the outputs from Table 2 of the main paper; MAT uses the outputs from Table S4(c) (dashes denote methods outside that panel). CallRate and SR macro-average the five web datasets. Panel (b) uses the grouped MC-Search test split and task-weighted Overall F1. Bold/underlined entries mark best/second-best directed metrics within each backbone. Scores and rates are percentages; RD retains the released scorer's step units and Calls is a count.

(a) Native metrics and search rates

Backbone	Method	MMSearch	InfoSeek	SimpleVQA			MAT	Web suite	
		171 / F1↑	Native↑	Correct↑	Incorrect↓	No answer	Acc.↑	CallRate↓	SR↓
Qwen2.5-VL-7B	Fixed image–text RAG [13]	53.61	50.84	52.52	42.84	4.64	—	98.10	95.50
	ReAct [15]	48.82	46.02	49.85	44.03	6.12	61.33	69.80	59.00
	Visual-ARFT (restricted) [7]	51.46	51.47	54.00	40.57	5.43	67.33	81.20	71.50
	MMSearch-R1 [13]	56.27	54.62	57.26	39.09	3.65	65.33	79.60	69.50
	WebWatcher (restricted) [5]	54.12	52.31	55.28	39.78	4.94	62.67	91.30	83.00
	MM-DeepResearch (restricted) [14]	60.39	57.54	59.03	37.51	3.46	70.67	93.90	86.00
	CTD	**63.47**	**61.92**	**62.09**	**35.54**	2.37	**74.00**	**62.70**	**42.00**
Qwen3-VL-8B	Fixed image–text RAG [13]	58.64	56.82	62.49	34.45	3.06	—	98.50	96.50
	ReAct [15]	55.08	53.78	56.56	37.71	5.73	—	72.10	61.00
	SenseNova-MARS (restricted) [3]	67.81	64.27	68.11	29.52	2.37	75.33	84.40	77.50
	MM-DeepResearch (restricted) [14]	68.46	68.65	66.24	30.70	3.06	77.33	93.40	88.00
	CTD	**72.37**	**71.42**	**72.06**	**25.47**	2.47	**81.33**	**64.90**	**44.00**
InternVL3.5-8B	Fixed image–text RAG [13]	55.24	53.69	55.08	39.98	4.94	—	98.20	96.00
	ReAct [15]	51.77	50.17	51.63	42.15	6.22	63.33	71.80	60.00
	CTD	**62.06**	**60.88**	**61.80**	**35.04**	3.16	**72.00**	**65.80**	**45.50**

(b) MC-Search: overall quality, chain alignment, and search rates

Backbone	Method	Overall↑	HPS↑	RD	Calls↓	CallRate↓	SR↓
Qwen2.5-VL-7B	ReAct [15]	37.75	22.60	−2.39	1.31	78.3	65.5
	Search-Align [8]	43.79	**32.90**	−1.99	1.71	96.1	85.5
	CTD	**47.18**	19.80	−2.82	**0.88**	**66.8**	**44.0**
Qwen3-VL-8B	ReAct [15]	43.20	**24.10**	−2.36	1.34	79.4	67.0
	CTD	**52.43**	22.20	−2.79	**0.91**	**68.3**	**45.5**
InternVL3.5-8B	ReAct [15]	40.52	23.40	−2.38	1.32	77.5	66.0
	Search-Align [8]	45.19	**32.20**	−2.05	1.65	93.7	82.5
	CTD	**48.60**	21.80	−2.77	**0.93**	**70.3**	**46.5**

C.9. Qualitative cases and the one-step horizon

A bridge with no immediate benefit. Figure S1 shows how the one-step horizon affects a bridge call in MC-Search. The current evidence identifies the photographed hall as Keble College. A sister-college lookup returns Selwyn College, followed by the incorrect answer 1870. Its immediate gain is zero, and the predicted gain of $+1.0$ F1 point falls below the $+5.0$-point penalty, so CTD stops. Reaching the correct year, 1909, requires another lookup beyond the one-step budget. A greedy extension may stop before acquiring this bridge evidence.

Case sources and selection. Images and original annotations come from the authors' public Hugging Face releases, YennNing/MC-Search and CaraJ/MMSearch. The main figure uses MC-Search indices 75 and 97 and MMSearch item `technology_0`; the boundary figure uses MC-Search index 32. Cases were selected for visual clarity and correspondence to the decision mechanisms. MC-Search items come from the release's `train` split, and MMSearch uses its public `end2end` configuration and split.

Boundary-case details and scoring. The boundary case uses the original image, full question, reference answer, and chain annotations. The sequence shows the candidate calls, current evidence, returned snippets, predicted gains, and answers. Token-F1 scoring follows Sec. A.3, with three answers per branch. Answers are abbreviated.

Table S6. **Gain prediction and local decisions on held-out MC-Search states.** Panel (a) averages three training seeds. MAE and signed-gain calibration error ($SCal_{10}$) use initial states; reference gaps separate initial and collected-prefix states. Errors and gaps use F1 percentage points; Calls is a count. Gap estimates can be negative. Prefix Calls includes the observed prefix. Dashes denote inapplicable gain metrics. Panel (b) gives CTD's seed-0 calibration bins with candidate counts and weighted predicted/measured gains.

(a) Prediction errors and independently evaluated local decisions

Backbone	Router	MAE↓	$SCal_{10}$ ↓	Reference gap		Calls	
				Initial↓	Prefix↓	Initial↓	Prefix (total)↓
Qwen2.5-VL-7B	Binary gate + ranker [16]	—	—	3.94	3.73	0.47	1.79
	Absolute-value regression	11.86	3.39	3.15	3.20	**0.46**	**1.78**
	CTD	**9.42**	**1.24**	**1.82**	**1.97**	0.47	1.79
Qwen3-VL-8B	Binary gate + ranker [16]	—	—	3.76	3.48	0.49	1.81
	Absolute-value regression	11.02	3.12	2.91	2.99	0.48	1.80
	CTD	**8.67**	**1.17**	**1.61**	**1.78**	**0.46**	**1.78**
InternVL3.5-8B	Binary gate + ranker [16]	—	—	4.16	4.09	0.49	1.83
	Absolute-value regression	11.51	3.45	3.39	3.54	**0.48**	**1.82**
	CTD	**9.18**	**1.33**	**2.05**	**2.16**	0.49	1.83

(b) CTD calibration on initial states; training seed 0

CTD prediction bin	Qwen2.5-VL-7B			Qwen3-VL-8B			InternVL3.5-8B		
(F1 points)	Count	Pred.	Measured	Count	Pred.	Measured	Count	Pred.	Measured
$[-100, -80)$	0	—	—	0	—	—	0	—	—
$[-80, -60)$	0	—	—	0	—	—	0	—	—
$[-60, -40)$	3	−48.5	−49.4	3	−45.7	−45.1	2	−46.7	−46.0
$[-40, -20)$	13	−28.3	−30.2	15	−27.2	−28.8	13	−27.9	−30.1
$[-20, 0)$	840	−8.4	−9.5	831	−7.8	−8.9	832	−8.6	−9.7
$[0, 20)$	705	6.2	7.4	748	6.8	7.8	680	6.1	7.3
$[20, 40)$	18	25.7	26.8	25	26.2	25.3	21	25.8	24.8
$[40, 60)$	7	46.3	44.2	5	44.1	46.2	4	45.3	47.0
$[60, 80)$	1	65.4	62.9	0	—	—	0	—	—
$[80, 100]$	0	—	—	0	—	—	0	—	—

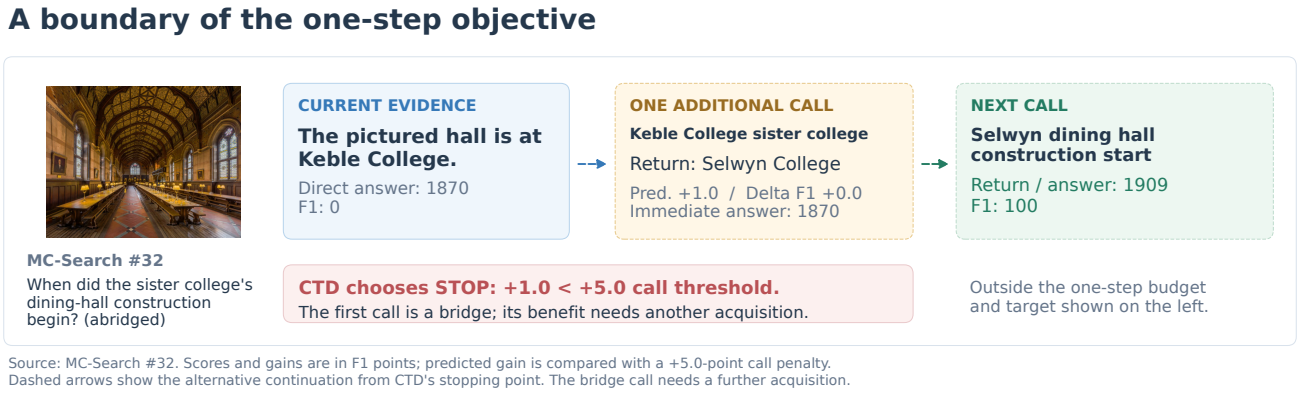

Figure S1. **A bridge call with delayed benefit.** The image, question, reference year, and entity chain come from MC-Search example 32. Starting from an evidence prefix identifying Keble College, a call identifies its sister college, Selwyn. The immediate answer remains incorrect, giving zero incremental F1; a subsequent call supplies the reference year, 1909. Dashed arrows show the alternative continuation from CTD's stopping point.

Table S9. **Quality–cost trade-off without refitting the gain predictor.** MC-Search initial states, $B = 1$. Each method reuses its fitted head across all seven thresholds. Columns report test F1 and mean attempted calls, averaged over three training seeds. Development selection fixes $\lambda = 0.05$ before testing. Bold/underline compare the two methods at the same λ within each backbone.

Method	λ	Qwen2.5-VL-7B		Qwen3-VL-8B		InternVL3.5-8B	
		F1↑	Calls↓	F1↑	Calls↓	F1↑	Calls↓
Absolute-value regression	0	<u>40.97</u>	**0.66**	<u>47.53</u>	**0.68**	<u>43.76</u>	**0.71**
	0.01	<u>41.21</u>	**0.61**	<u>47.85</u>	**0.62**	<u>44.04</u>	**0.65**
	0.025	<u>41.49</u>	**0.54**	<u>48.09</u>	<u>0.55</u>	<u>44.21</u>	**0.57**
	0.05	<u>41.62</u>	**0.46**	<u>48.19</u>	<u>0.48</u>	<u>44.33</u>	**0.48**
	0.1	<u>40.34</u>	**0.25**	<u>47.16</u>	<u>0.29</u>	<u>43.09</u>	**0.30**
	0.2	<u>37.18</u>	**0.05**	<u>43.66</u>	**0.06**	<u>40.24</u>	**0.07**
	0.4	**36.42**	**0.00**	**42.83**	**0.00**	**39.63**	**0.00**
CTD	0	**42.32**	<u>0.72</u>	**48.69**	<u>0.70</u>	**44.82**	<u>0.74</u>
	0.01	**42.65**	<u>0.65</u>	**49.07**	<u>0.63</u>	**45.18**	<u>0.67</u>
	0.025	**42.91**	<u>0.56</u>	**49.38**	**0.54**	**45.54**	<u>0.58</u>
	0.05	**43.08**	<u>0.47</u>	**49.56**	**0.46**	**45.72**	<u>0.49</u>
	0.1	**41.87**	<u>0.29</u>	**48.26**	**0.27**	**44.37**	<u>0.31</u>
	0.2	**37.76**	<u>0.07</u>	**44.11**	**0.06**	**40.64**	<u>0.08</u>
	0.4	**36.42**	**0.00**	**42.83**	**0.00**	**39.63**	**0.00**

Table S10. **Sensitivity of CTD on Qwen2.5-VL-7B.** MC-Search initial states, $B = 1$, three training seeds. Only one factor differs from the default $(K, R, m) = (4, 3, 256)$. F1 is mean ± sample standard deviation; Calls is the seed mean. MAE uses the same default-K diagnostic bank for every variant. Offline columns give actual supervision tokens and tool attempts; R varies label sampling only, while evaluation keeps three answer draws.

CTD variant	K	R	m	F1↑	Calls↓	MAE↓	Tokens (M)	Tool calls
CTD: Default	4	3	256	43.08±0.21	0.47	9.42	312.6	14,280
CTD: Fewer candidates	2	3	256	41.97±0.27	**0.44**	10.68	229.8	7,410
CTD: More candidates	8	3	256	**43.43**±0.24	0.49	<u>9.19</u>	489.7	26,970
CTD: One label draw	4	1	256	42.18±0.33	**0.44**	11.27	155.2	14,280
CTD: Five label draws	4	5	256	<u>43.29</u>±0.21	0.47	**9.01**	470.1	14,280
CTD: Narrower head	4	3	128	42.69±0.29	<u>0.46</u>	9.88	312.1	14,280
CTD: Wider head	4	3	512	43.11±0.28	0.47	9.33	313.4	14,280

Table S11. **Supervision-cost and decoding controls on Qwen2.5-VL-7B.** MC-Search initial states, $B = 1$. Panel (a) applies common offline model-token and tool-attempt limits and reports actual usage. Panel (b) uses the standard training set with sampler-specific labels measured on the same tool returns. Learned-router F1 is mean ± sample SD over three fitted heads. Direct answering has no training-seed SD.

Method	Params (K)	Labels	Tokens (M)	Tool calls	GPU-h	F1↑	Calls↓
(a) Matched offline supervision budgets							
Common limits: model tokens (M) = 180; tool attempts = 12,000							
Success-only imitation [8]	1,836	2,680	165.9	10,920	207.4	39.31±0.27	**0.45**
Likelihood proxy [12]	1,836	6,410	173.6	11,340	219.2	39.78±0.24	0.47
Binary gate + ranker [16]	1,839	6,290	175.1	11,610	224.8	40.36±0.25	0.46
Absolute-value regression	1,839	6,360	176.2	11,700	225.9	41.09±0.24	0.46
CTD	1,836	6,450	175.4	11,680	223.6	**42.53**±0.22	0.47

(b) Answer-sampler control with sampler-specific supervision				
Method	Stochastic: $T = 0.2$, top-$p = 0.9$		Greedy	
	F1↑	Calls↓	F1↑	Calls↓
Direct answering [8]	36.42	**0.00**	37.66	**0.00**
Success-only imitation [8]	39.84±0.17	0.46	40.92±0.25	0.44
Likelihood proxy [12]	40.31±0.15	0.48	41.33±0.22	0.46
Binary gate + ranker [16]	40.93±0.16	0.47	41.94±0.25	0.45
Absolute-value regression	41.62±0.20	0.46	42.76±0.23	0.45
CTD	**43.08**±0.21	0.47	**44.19**±0.20	0.46

Table S12. **Training-seed stability and paired uncertainty.** MC-Search initial states, $B = 1$. Each seed-specific F1 averages three fresh answer draws. Mean ± SD summarizes the three independently fitted heads; Calls is their mean. Panel (b) reports CTD minus absolute-value regression with paired 95% intervals from 2,000 task–image-group bootstrap resamples, preserving all states and answer repetitions within a group.

(a) Independent training seeds; three answer draws per fitted head						
Backbone	Router	Seed 0	Seed 1	Seed 2	F1 mean ± SD	Calls
Qwen2.5-VL-7B	Success-only imitation [8]	39.65	39.91	39.96	39.84±0.17	**0.46**
	Likelihood proxy [12]	40.18	40.28	40.47	40.31±0.15	0.48
	Binary gate + ranker [16]	40.75	41.03	41.01	40.93±0.16	0.47
	Absolute-value regression	41.40	41.68	41.78	41.62±0.20	**0.46**
	CTD	**42.84**	**43.19**	**43.21**	**43.08**±0.21	0.47
Qwen3-VL-8B	Success-only imitation [8]	46.35	46.57	46.73	46.55±0.19	0.48
	Likelihood proxy [12]	46.90	47.18	46.95	47.01±0.15	0.47
	Binary gate + ranker [16]	47.49	47.86	47.81	47.72±0.20	0.49
	Absolute-value regression	48.04	48.23	48.30	48.19±0.13	0.48
	CTD	**49.28**	**49.73**	**49.67**	**49.56**±0.24	**0.46**
InternVL3.5-8B	Success-only imitation [8]	42.62	42.90	42.82	42.78±0.14	**0.46**
	Likelihood proxy [12]	43.01	43.28	43.34	43.21±0.18	0.48
	Binary gate + ranker [16]	43.75	44.03	44.04	43.94±0.16	0.49
	Absolute-value regression	44.06	44.45	44.48	44.33±0.23	0.48
	CTD	**45.51**	**45.80**	**45.85**	**45.72**±0.18	0.49

(b) Paired CTD-minus-absolute-value differences and 95% intervals						
CTD vs. absolute-value	ΔF1 (percentage points)			ΔCalls		
Backbone	Estimate	95% lower	95% upper	Estimate	95% lower	95% upper
Qwen2.5-VL-7B	+1.46	+0.43	+2.49	+0.01	−0.03	+0.05
Qwen3-VL-8B	+1.37	+0.32	+2.39	−0.02	−0.06	+0.02
InternVL3.5-8B	+1.39	+0.29	+2.53	+0.01	−0.03	+0.05

Table S13. **End-to-end resource profiling on Qwen2.5-VL-7B.** One H100, batch size one, identical serving configuration. Panel (a) reports cached-replay execution, including candidate generation, feature extraction, helper/draft computation, and final answering; token columns are means in thousands per task. Latency is per-task median/P95 in seconds; memory is peak device memory in GB. Panel (b) separately times uncached web retrieval in a shared provider window on the same 171 questions.

Method	Calls	Text in (k)	Visual in (k)	Output (k)	P50 (s)	P95 (s)	GPU (GB)
(a1) MC-Search initial states, $B = 1$; cached replay							
Direct answering [8]	**0.00**	**2.80**	**3.20**	**0.18**	**1.86**	**3.07**	**18.70**
Fixed-first retrieval [8]	0.96	4.92	4.64	0.43	4.71	8.21	20.30
Random routing	0.48	4.47	4.09	0.36	3.89	7.42	20.10
Confidence gate	0.47	5.91	5.40	0.58	4.83	8.58	21.00
Entropy gate	0.49	6.03	5.35	0.60	4.96	8.81	21.10
Success-only imitation [8]	0.46	6.84	8.91	0.38	5.74	9.60	21.70
Likelihood proxy [12]	0.48	6.93	9.06	0.41	5.82	9.94	21.80
Binary gate + ranker [16]	0.47	7.02	9.14	0.40	5.86	9.87	21.80
Absolute-value regression	0.46	7.18	9.26	0.41	5.97	10.13	21.90
CTD	0.47	7.13	9.20	0.40	5.92	10.02	21.80
(a2) MMSearch-171, $B = 2$; cached replay							
Fixed image–text RAG [13]	1.92	**13.62**	**12.41**	**1.05**	**9.44**	**16.28**	**23.80**
ReAct [15]	1.20	15.28	13.13	1.37	11.36	20.14	24.20
Visual-ARFT (restricted) [7]	1.45	18.46	15.29	1.62	12.02	21.43	24.60
MMSearch-R1 [13]	1.36	16.11	14.71	1.49	11.72	21.04	24.50
WebWatcher (restricted) [5]	1.63	19.14	16.88	1.84	13.25	23.46	25.30
MM-DeepResearch (restricted) [14]	1.71	21.37	18.19	2.09	14.61	26.48	25.70
CTD	**0.87**	16.49	16.03	1.12	11.09	19.56	24.40

Method	P50 (s)	P95 (s)
(b) MMSearch-171, $B = 2$; uncached retrieval		
Fixed image–text RAG [13]	18.28	35.67
ReAct [15]	18.73	36.44
Visual-ARFT (restricted) [7]	21.85	42.16
MMSearch-R1 [13]	19.62	38.33
WebWatcher (restricted) [5]	23.49	46.17
MM-DeepResearch (restricted) [14]	25.63	50.92
CTD	**16.84**	**33.71**

D. Theoretical Properties and Proofs

D.1. Shared-reference contrasts

Fix a state with $K = K_s \geq 1$ candidates. Write $\mathbf{u}_r = [u_{0,r}, \ldots, u_{K,r}]^\top$ and $\bar{\mathbf{u}} = R^{-1} \sum_r \mathbf{u}_r$. The labels in Eq. (2) of the main paper have the equivalent matrix form

$$\begin{aligned} \mathbf{C}_K &= [-\mathbf{1}_K \ \mathbf{I}_K] \in \mathbb{R}^{K \times (K+1)}, \\ \widehat{\boldsymbol{\Delta}} &= \mathbf{C}_K \bar{\mathbf{u}}, \qquad \boldsymbol{\Delta} = \mathbf{C}_K \mathbb{E}[\mathbf{u}_r]. \end{aligned} \tag{S8}$$

Each row subtracts the direct-answer score from one candidate score. If $\mathbf{C}_K \mathbf{u} = \mathbf{C}_K \mathbf{v}$, every component of $\mathbf{u} - \mathbf{v}$ equals its zeroth component. Hence $\ker \mathbf{C}_K = \operatorname{span}(\mathbf{1}_{K+1})$: the contrast removes a common offset and retains every pairwise action difference. In particular, $\Delta_i - \Delta_j = \mathbb{E}[u_{i,r} - u_{j,r}]$.

For $\boldsymbol{\mu} = \mathbb{E}[\mathbf{u}_r]$, the augmented gain vector $[0, \boldsymbol{\Delta}^\top]^\top$ equals $\boldsymbol{\mu} - \mu_0 \mathbf{1}_{K+1}$. A softmax applied at a common temperature is therefore identical for these two vectors. Direct gain regression fits the scalar differences themselves, with zero fixed by the stopping action. Token-loss contrasts, as used in Toolformer [12], measure changes in likelihood; Eq. (2) of the main paper measures changes in final-answer F1.

D.2. Sampling covariance

Suppose repetition vectors $\mathbf{u}_r$ are independent and identically distributed, with covariance $\boldsymbol{\Sigma}_u$. Linear transformation of their sample mean gives

$$\operatorname{Cov}(\widehat{\boldsymbol{\Delta}}) = \frac{1}{R} \mathbf{C}_K \boldsymbol{\Sigma}_u \mathbf{C}_K^\top. \tag{S9}$$

This expression includes correlations among branches within each repetition. Reusing the reference score introduces a common noise component; its effect on the final covariance also depends on correlations with the candidate scores. The squared loss in Eq. (4) of the main paper applies to correlated candidate labels. Grouped evaluation preserves the dependence among states from the same task.

D.3. Decision error and action recovery

Consider the action set $\mathcal{A} = \mathcal{F}(s) \cup \{0\}$. Set $\Delta_0 = 0$ and $q_a = \Delta_a - \lambda c_a$. Let $a^\star$ maximize q_a over $\mathcal{A}$, and let $\hat{a}$ be selected by Eq. (6) of the main paper. Define $\varepsilon = \max_{j \in \mathcal{F}(s)} |g_\theta(s, t_j) - \Delta_j(s)|$, taking $\varepsilon = 0$ when $\mathcal{F}(s)$ is empty. Since true and predicted values use the same charges, $|q_a - \widehat{q}_a| \leq \varepsilon$ for all feasible tools, and the error for STOP is zero. Therefore,

$$\begin{aligned} 0 \leq q_{a^\star} - q_{\hat{a}} &= (q_{a^\star} - \widehat{q}_{a^\star}) + (\widehat{q}_{a^\star} - \widehat{q}_{\hat{a}}) \\ &\quad + (\widehat{q}_{\hat{a}} - q_{\hat{a}}) \\ &\leq \varepsilon + 0 + \varepsilon = 2\varepsilon. \end{aligned} \tag{S10}$$

The middle term is nonpositive by the definition of $\hat{a}$. When $a^\star$ is unique and there is another feasible action, define its margin as $\gamma = q_{a^\star} - \max_{a \in \mathcal{A} \setminus \{a^\star\}} q_a$. If $\gamma > 2\varepsilon$, every competing action retains a lower predicted value, and $\hat{a} = a^\star$. When STOP is the sole action, selection is exact.

References

[1] Yang Chen, Hexiang Hu, Yi Luan, Haitian Sun, Soravit Changpinyo, Alan Ritter, and Ming-Wei Chang. Can pre-trained vision and language models answer visual information-seeking questions? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14948–14968, 2023. S4

[2] Xianfu Cheng, Wei Zhang, Shiwei Zhang, Jian Yang, Xiangyuan Guan, Xianjie Wu, Xiang Li, Ge Zhang, Jiaheng Liu, Yuying Mai, Yutao Zeng, Zhoufutu Wen, Ke Jin, Baorui Wang, Weixiao Zhou, Yunhong Lu, Hangyuan Ji, Tongliang Li, Wenhao Huang, and Zhoujun Li. SimpleVQA: Multimodal factuality evaluation for multimodal large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4637–4646, 2025. S4

[3] Yong Xien Chng, Tao Hu, Wenwen Tong, Xueheng Li, Jiandong Chen, Haojia Yu, Jiefan Lu, Hewei Guo, Hanming Deng, Chengjun Xie, Gao Huang, Dahua Lin, and Lewei Lu. SenseNova-MARS: Empowering multimodal agentic reasoning and search via reinforcement learning. *arXiv preprint arXiv:2512.24330*, 2025. Version 2, January 2026. S4, S5, S7, S8, S10, S11

[4] Mingyang Fu, Yuyang Peng, Dongping Chen, Zetong Zhou, Benlin Liu, Yao Wan, Zhou Zhao, Philip S. Yu, and Ranjay Krishna. Seeking and updating with live visual knowledge. *arXiv preprint arXiv:2504.05288*, 2025. S4

[5] Xinyu Geng, Peng Xia, Zhen Zhang, Xinyu Wang, Qiuchen Wang, Ruixue Ding, Chenxi Wang, Jialong Wu, Yida Zhao, Kuan Li, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. WebWatcher: Breaking new frontier of vision-language deep research agent. *arXiv preprint arXiv:2508.05748*, 2025. S4, S5, S8, S10, S11, S15

[6] Dongzhi Jiang, Renrui Zhang, Ziyu Guo, Yanmin Wu, Jiayi Lei, Pengshuo Qiu, Pan Lu, Zehui Chen, Guanglu Song, Peng Gao, Yu Liu, Chunyuan Li, and Hongsheng Li. MMSearch: Unveiling the potential of large models as multi-modal search engines. In *International Conference on Learning Representations*, 2025. S4

[7] Ziyu Liu, Yuhang Zang, Yushan Zou, Zijian Liang, Xiaoyi Dong, Yuhang Cao, Haodong Duan, Dahua Lin, and Jiaqi Wang. Visual agentic reinforcement fine-tuning. *arXiv preprint arXiv:2505.14246*, 2025. S4, S5, S7, S8, S10, S11, S15

[8] Xuying Ning, Dongqi Fu, Tianxin Wei, Mengting Ai, Jiaru Zou, Ting-Wei Li, Hanghang Tong, Yada Zhu, Hendrik Hamann, and Jingrui He. MC-Search: Evaluating and enhancing multimodal agentic search with structured long reasoning chains. In *International Conference on Learning Representations*, 2026. S2, S3, S4, S5, S9, S11, S14, S15

[9] OpenGVLab. InternVL3.5-8B model card. Hugging Face, 2025. S6

[10] Qwen Team. Qwen2.5-VL-7B-Instruct model card. Hugging Face, 2025.

[11] Qwen Team. Qwen3-VL-8B-Instruct model card. Hugging Face, 2025. S6

[12] Timo Schick, Jane Dwivedi-Yu, Roberto Dessí, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, 2023. S5, S14, S15, S16

[13] Jinming Wu, Zihao Deng, Wei Li, Yiding Liu, Bo You, Bo Li, Zejun Ma, and Ziwei Liu. MMSearch-R1: Incentivizing LMMs to search. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2456–2487. Association for Computational Linguistics, 2026. S4, S5, S7, S8, S10, S11, S15

[14] Huanjin Yao, Qixiang Yin, Min Yang, Ziwang Zhao, Yibo Wang, Haotian Luo, Jingyi Zhang, and Jiaxing Huang. MMDeepResearch: A simple and effective multimodal agentic search baseline. In *Proceedings of the 43rd International Conference on Machine Learning*, pages 147736–147756. PMLR, 2026. S5, S8, S10, S11, S15

[15] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. S5, S8, S9, S10, S11, S15

[16] Wenhao Zou, Xianglong Liu, Wendong Bi, Hanjie Wang, Simin Zhao, and Gong Zhi. CoBRA: Learning tool-use boundaries via counterfactual margins. In *Findings of the Association for Computational Linguistics: EMNLP 2026*, pages 28092–28110. Association for Computational Linguistics, 2026. S5, S12, S14, S15