SHOW OR STOW: LEARNING EVIDENCE ADMISSION FROM STAGE-SPECIFIC UTILITY IN MULTI-AGENT SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic search has become a central approach to reasoning-intensive information retrieval. When multiple agents carry it out, the evidence they collect guides further search and supports final-answer generation, but the same evidence can have different utility at these two stages. Existing approaches organize shared evidence or learn what to share without separately learning its value for each use. To mitigate this mismatch between stages, we propose *StageAdmit*, an evidence admission framework that keeps every collected item in an evidence pool and regresses stage-specific keep-mask utilities to learn separate admission policies for continued search and final-answer generation. A predictor with stage-specific heads decides which evidence enters each search agent's panel during search and which enters the writer's panel for the final answer. The predictor is trained on paired comparisons that keep or mask a piece of evidence, measuring its effect on final-answer quality and subsequent tool calls for continued search, and its marginal gain for final-answer generation. Because all evidence remains in the pool, evidence withheld from search agents can still reach the writer. Experiments on WideSearch and GISA show that StageAdmit outperforms all evaluated baselines, surpassing the strongest baseline by 1.5 and 1.6 Item-F1 points with GLM-5, while reducing completion time by 13.1% and writer input tokens by 36.4%. The same frozen predictor also improves GPT-5.1 and Qwen3.8-27B agents without retraining.

1 INTRODUCTION

Broad information-seeking tasks require answers that integrate many entities, attributes, and sources (Wong et al., 2026). Multi-agent search addresses them by having a coordinator divide each question into search tasks, search agents investigate these tasks in parallel, and a writer assemble the final answer (Zhang et al., 2026a; Huang et al., 2026). The evidence that search agents collect therefore serves two purposes: it guides further search and supports the final answer. The utility of evidence is known to depend on the context in which it is used (Deng et al., 2025; Song et al., 2026); in multi-agent search, it also depends on which of these two stages uses it. Consider a question requesting universities together with their rankings, application deadlines, and fees. A failed search can keep another agent from repeating it without supplying any answer content. A verified ranking record can fill a requested field without changing the remaining search. Conflicting deadlines can affect both what agents investigate next and what the writer reports. As illustrated in Figure 1, utility for continued search need not imply utility for final-answer generation, or vice versa. Deciding which evidence each agent sees, which we call evidence admission, should therefore account for the stage that will use it.

Prior work has approached this decision from two directions, each centered on one stage. Shared-state and communication methods organize intermediate evidence so that parallel agents can coordinate (Zhang et al., 2026a; Huang et al., 2026; Fioresi et al., 2026; Liu et al., 2025). Context-utility methods estimate how much retrieved context contributes to the final answer (Deng et al., 2025; Zhu et al., 2025; Song et al., 2026). However, neither direction separately measures how the same evidence item affects continued search and final-answer generation, so neither can tell when an item helps one stage but not the other.

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

This gap motivates the central research question of this paper:

> *Can learning evidence admission from stage-specific downstream utility improve the quality and efficiency of multi-agent search?*

Answering this question first requires measuring the utility of each evidence item at each stage directly. To this end, we conduct keep-mask experiments. The keep-mask utility of an item is the change in a downstream outcome when the item is shown to a recipient rather than withheld, with the preceding state fixed. For continued search, it combines the change in final-answer quality with the change in subsequent tool calls; for final-answer generation, it is the item's marginal gain in answer quality given the evidence already admitted. On held-out WideSearch and GISA questions, the two utilities have opposite signs for 47% of evidence items and share only 18% of their top-8 items, so a single admission policy cannot serve both stages.

In this paper, we therefore propose StageAdmit, an evidence admission framework that keeps every collected item in an evidence pool and regresses stage-specific keep-mask utilities to learn separate admission policies for continued search and final-answer generation. StageAdmit rests on two design choices. First, it separates storing evidence from showing it: admission decides only which panel displays an item, namely a search agent's search evidence panel while search is running or the writer's answer evidence panel afterward. An item withheld from search agents can therefore still reach the writer. Second, one predictor with a shared backbone and stage-specific heads learns each admission from its own utility, and each stage admits items with positive predicted utility.

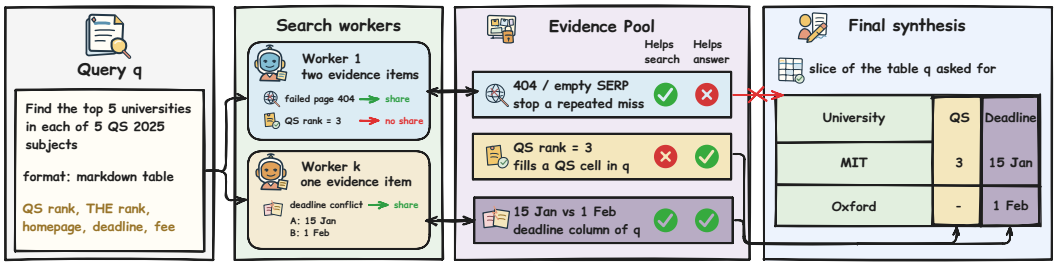


Figure 1: The same evidence item can help continued search, final-answer generation, both, or neither. The evidence pool retains every item: a failed lookup may steer later search, a ranking may fill an answer cell, and a conflict may matter to both.

Our main contributions are listed below:

1. We design stage-specific keep-mask comparisons that separately measure an item's effect on continued search and its contribution to the final answer, and show that the two utilities often diverge.
2. We propose StageAdmit, which retains all evidence items and learns separate admission policies for the search and answer evidence panels from stage-specific keep-mask utilities. A single predictor makes both decisions without keep-mask comparisons at inference.
3. On WideSearch and GISA, StageAdmit outperforms all evaluated baselines with all three backbones, exceeding the strongest baseline by 1.5 and 1.6 Item-F1 with GLM-5 under Avg@4, and improves over admission at either stage alone. It also reduces completion time by 13.1% and writer input tokens by 36.4%.

2 Related Work

Multi-agent search and context sharing. Broad search systems organized intermediate results so parallel agents could coordinate. SearchOS gave agents role-specific views of structured search state, Web2BigTable used a shared workboard, and Argus tracked missing evidence in a graph (Zhang et al., 2026a; Huang et al., 2026; Zhang et al., 2026b). Table-as-Search and A-MapReduce instead structured the work as table completion and map-reduce execution (Lan et al.,

2026; Chen et al., 2026a). At the infrastructure level, Cordis supported dynamic composition of runtime components through reversible effects and reactive dependencies (Shi et al., 2026). Closer to evidence admission, Learning to Share (LTS) learned which intermediate steps to share across parallel teams from task outcomes and usage-aware rewards, while RCR-Router routed context by role and interaction round (Fioresi et al., 2026; Liu et al., 2025). These methods organized execution or controlled access to intermediate information. StageAdmit learns from two separately measured outcomes for the same evidence item: its effect on continued search and on the writer's answer.

Utility-based evidence selection. Work on answer generation estimated whether selected context improved the final response. CI-value measured passage influence with leave-one-out comparisons, and Context-Picker mined minimal sufficient evidence sets for a selection policy (Deng et al., 2025; Zhu et al., 2025). A distilled utility-based selector learned passage judgments from a teacher model (Zhang et al., 2025). Information Gain Pruning selected passages using generator-aligned utility, while Shapley Context Pruning modeled interactions among context units (Song et al., 2026; Chen et al., 2026b). In parallel-agent settings, AggAgent let a final agent inspect completed trajectories on demand, while Infogent combined selective aggregation with feedback to navigation (Lee et al., 2026; Gangi Reddy et al., 2025). These approaches selected passages for answer generation or combined agents' outputs. StageAdmit also measures whether showing an item changes subsequent search, then admits writer evidence from the full evidence pool, including items withheld from search agents.

3 PRELIMINARIES

3.1 TASK AND EVIDENCE

A broad-search question q specifies a natural-language request and the required answer fields or format, including target entities and attributes when applicable. A coordinator divides q into tasks, search agents investigate them in parallel, and a writer produces the final answer.

An evidence item e is a typed intermediate result with provenance. A retrieved fact records an entity, attribute, value, source, and supporting span. Failed lookups record what was tried and its outcome; conflicting observations record the incompatible claims and their sources. Visiting a page alone does not create an evidence item. Every produced item enters the evidence pool, regardless of whether it is shown to another agent.

A search agent's search evidence panel contains items visible to that agent during execution. The writer's answer evidence panel is an ordered sequence selected after search. At a decision, $M = (m_1, \ldots, m_k)$ denotes the recipient's current sequence of k evidence items: for a search agent, its current panel; for the writer, the items already selected for the answer panel. We write $M \oplus e$ for appending e to this sequence. Admission changes a panel, while the evidence pool retains the item.

3.2 PROBLEM FORMULATION

We distinguish execution and synthesis stages, indexed by $r \in \{\mathrm{exec}, \mathrm{syn}\}$. Let C_r describe the recipient's role, assigned task, and output constraints. The decision is whether to show item e given (q, C_r, M). During execution, showing e to another search agent can change the remaining search. During synthesis, adding e to M can change the writer's answer.

To measure each effect, a keep branch shows e and a mask branch withholds it; both start from the same state under the same continuation protocol. Let $Q_r^+, Q_r^- \in [0, 1]$ be the terminal answer-quality scores of these branches, and define $\Delta Q_r = Q_r^+ - Q_r^-$. For execution, let T^+ and T^- count subsequent search and page-opening calls in the two branches, with $\Delta T = T^+ - T^-$. Given a fixed cost weight $\lambda \geq 0$, the stage-specific utilities are

$$\begin{aligned} U_{\mathrm{exec}}(e \mid q, C_{\mathrm{exec}}, M) &= \Delta Q_{\mathrm{exec}} - \lambda \Delta T, \\ U_{\mathrm{syn}}(e \mid q, C_{\mathrm{syn}}, M) &= \Delta Q_{\mathrm{syn}}. \end{aligned} \tag{1}$$

Execution quality is scored after continued search; synthesis quality is scored on writer calls with $M \oplus e$ or M. Lower subsequent tool use increases execution utility. Section 5.1 specifies the quality metrics and λ.

4 StageAdmit

StageAdmit moves from offline supervision to online admission, and the rest of this section follows the same order. Section 4.1 constructs keep-mask supervision that measures the execution and synthesis utilities U_{exec} and U_{syn} of each evidence item. Section 4.2 introduces a utility predictor with a shared backbone and stage-specific heads that outputs $\widehat{U}_r$, the predicted utility of showing an item at stage r. Section 4.3 trains the predictor on both utilities and uses it to drive the two admission policies. As shown in Figure 2, every evidence item enters the evidence pool at inference. Execution admission shows an item to other search agents when $\widehat{U}_{\text{exec}} > 0$. After search, synthesis admission visits the full pool in a fixed order and appends items with $\widehat{U}_{\text{syn}} > 0$ to the answer evidence panel. The pool keeps items rejected at execution, so they can still reach the writer; the two admissions use separate targets because the two utilities can differ in sign and ranking (Section 5.4).

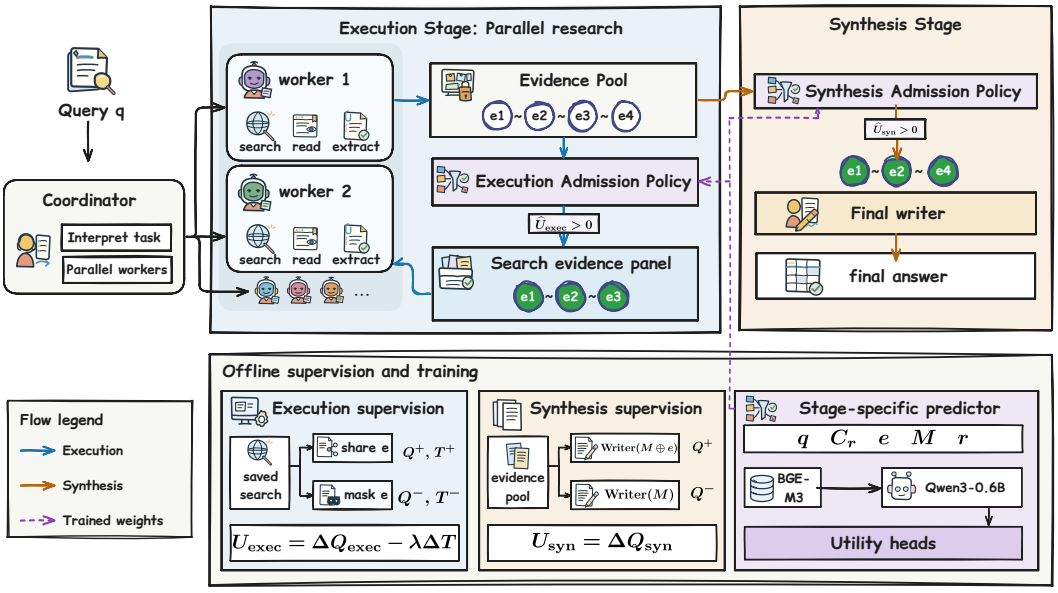


Figure 2: Framework overview. The coordinator assigns parallel search tasks, and workers place every evidence item in the pool. The execution policy shares an item with other workers when $\widehat{U}_{\text{exec}} > 0$, without removing rejected items; here e_4 stays out of the search evidence panel. The synthesis policy then admits items with $\widehat{U}_{\text{syn}} > 0$ for the final writer; here e_3 is withheld. Offline, paired keep-mask rollouts supply $U_{\text{exec}} = \Delta Q_{\text{exec}} - \lambda \Delta T$ and $U_{\text{syn}} = \Delta Q_{\text{syn}}$. One predictor with stage-specific heads learns both targets and drives the two policies.

4.1 Keep-Mask Supervision

To learn admission from downstream outcomes rather than from relevance heuristics, we need labels that isolate the effect of showing one item at one stage. We therefore compare a keep branch and a mask branch that start from the same state, once under continued search and once at the writer.

Execution. Execution supervision measures how sharing an evidence item affects the remainder of the search. We pause a live session at a sharing decision and save its continuation state. A paired keep-mask intervention resumes this checkpoint in two branches: the keep branch exposes the item to the designated search agent, and the mask branch withholds it from that agent. Both branches retain the item in the evidence pool and use Share-all for subsequently produced items. Each continuation yields a terminal answer score and a count of subsequent search and page-opening calls, giving $U_{\text{exec}} = \Delta Q_{\text{exec}} - \lambda \Delta T$. The predictor receives the evidence visible to the recipient before the intervention. Subsequent queries, scheduling, and evidence may diverge as agents respond to the changed visibility; these downstream effects are what execution utility measures. Appendix A details the collection protocol.

Synthesis. Synthesis supervision measures an item's contribution to the evidence already admitted for the writer. We freeze a completed evidence pool and visit items in a fixed order, starting with an empty answer evidence panel M. For each item e, we score the writer on M and on $M \oplus e$. The official quality difference gives the target $U_{\text{syn}} = \Delta Q_{\text{syn}}$. We append e only when this difference is positive. The target thus accounts for evidence overlap: a fact may contribute less once equivalent evidence has been admitted. Each comparison uses fresh writer calls with the question, output requirements, and admitted evidence, without browsing. Only the scorer receives reference answers.

All training labels come from a subset of WideSeek-R1 `width_20k` (Xu et al., 2026) (Appendix A). Official WideSearch and GISA questions are used only for evaluation. We retain branch outputs and scoring logs to trace each target to its keep-mask comparison.

4.2 Stage-Specific Utility Predictor

In order to make both admission decisions at inference without keep-mask rollouts, we train one predictor for both utilities. Both depend on the same question, item, and visible evidence, so the predictor shares its representation across execution and synthesis while stage-specific heads add scalar corrections. Each decision is represented by the question, the recipient description, the item under evaluation, and the current evidence sequence M. A frozen BGE-M3 encoder (Chen et al., 2024) maps each object, including each evidence item separately, to a normalized vector.

Let o denote one input object: q, C_r, e, or an item m_i in M. The frozen encoder E maps its text to a vector. The learned matrix W projects that vector, and $b_{\text{role}(o)}$ is an embedding for the object's role; together they give $g(o) = WE(o) + b_{\text{role}(o)}$. Let s_r be the learned stage embedding, v the learned readout embedding, and F_θ a Qwen3-0.6B transformer (Yang et al., 2025) adapted with LoRA (Hu et al., 2022) parameters θ. For $M = (m_1, \ldots, m_k)$, its final-position readout z_r is

$$z_r = F_\theta([s_r, g(q), g(C_r), g(e), g(m_1), \ldots, g(m_k), v])_{\text{last}}. \tag{2}$$

The stage markers `exec` and `syn` index the two stages. Let h_{shared} be a shared scalar head and h_r the scalar residual head for stage r. They jointly predict utility:

$$\widehat{U}_r = h_{\text{shared}}(z_r) + h_r(z_r). \tag{3}$$

The stage marker already lets the shared head vary by stage; zero-initialized residual heads learn additional corrections. Shared-head retains the stage marker and removes only the residual heads.

4.3 Training and Inference

Synthesis labels outnumber execution labels, so we balance the two stages during training to prevent one utility from dominating the shared backbone. At inference, the same predictor then drives both admission policies. Let $\mathcal{D}_r$ contain labeled decisions $x = (q, C_r, e, M)$ at stage r, each with target $u = U_r(e \mid q, C_r, M)$. The Huber loss $\text{Huber}_1(a)$ equals $a^2/2$ for $|a| \le 1$ and $|a| - 1/2$ otherwise. Let $\mathcal{L}_{\text{empty}}$ be an auxiliary loss that ranks a positive-utility item above a zero-utility item when the answer evidence panel is empty; α weights this term. The training objective is

$$\begin{aligned} \mathcal{L}_r &= \mathbb{E}_{(x,u)\sim\mathcal{D}_r}[\text{Huber}_1(\widehat{U}_r(x) - u)], \\ \mathcal{L} &= \tfrac{1}{2}\mathcal{L}_{\text{exec}} + \tfrac{1}{2}\mathcal{L}_{\text{syn}} + \alpha\mathcal{L}_{\text{empty}}. \end{aligned} \tag{4}$$

Each optimization step uses a batch from each stage, resampling the smaller dataset. Appendix A.1 gives the auxiliary loss and key training settings.

At inference, let a_r denote the binary admission decision. In synthesis, e_i is the ith item in fixed pool order, and M_i is the answer evidence panel before that item is considered. It starts empty: $M_0 = ()$. Admission and the synthesis sequence update follow

$$\begin{aligned} a_r(e \mid q, C_r, M) &= \mathbf{1}[\widehat{U}_r(e \mid q, C_r, M) > 0], \\ M_{i+1} &= \begin{cases} M_i \oplus e_i, & a_{\text{syn}}(e_i \mid q, C_{\text{syn}}, M_i) = 1, \\ M_i, & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

During execution, the rule updates the recipient's search evidence panel. During synthesis, it builds the answer evidence panel in pool order. Training constructs this sequence using observed U_{syn}; deployment uses predicted $\widehat{U}_{\text{syn}}$ and requires no reference answers or keep-mask rollouts. Rejected evidence items remain in the pool at both stages.

5 EXPERIMENTS

5.1 EXPERIMENTAL SETUP

Tasks and metrics. Official WideSearch (Wong et al., 2026) and GISA (Zhu et al., 2026) questions are used only for evaluation. We report the full WideSearch set of 200 questions (100 English and 100 Chinese) and the full GISA set of 373 questions (stable and live), averaging four independent runs (Avg@4). All quality scores in Table 1 use this protocol. WideSearch uses official Item-F1 and Row F1, where a predicted row counts as correct only when all of its cells match the corresponding reference row. For GISA we report table Item-F1, table Row F1 (defined analogously), set F1, and list content F1. Label construction uses scores in $[0, 1]$; reported task scores are multiplied by 100.

Supervision and implementation. Keep-mask labels come from 418 tasks drawn from WideSeek-R1's `width_20k`, a pool of 20,000 synthetic tasks disjoint from the WideSearch and GISA evaluation questions (Xu et al., 2026). The predictor learns the utilities in Equation 1, with $\lambda = 10^{-4}$ per subsequent search or page-opening call, and remains frozen during evaluation. The agent backbones are GLM-5 (GLM-5 Team et al., 2026), GPT-5.1 (OpenAI, 2025), and Qwen3.8-27B (Qwen Team, 2026). Appendix A details collection and training.

Comparisons. Table 1 compares StageAdmit with six reproduced baselines: ReAct (Yao et al., 2023), Plan-and-Solve (Wang et al., 2023), Table-as-Search (Lan et al., 2026), A-MapReduce (Chen et al., 2026a), Web2BigTable (Huang et al., 2026), and SearchOS (Zhang et al., 2026a). Controlled admission comparisons fix the agents, tools, budgets, and writer within each backbone (Appendix B). Share-all admits every evidence item at both stages. Exec-only and Syn-only apply learned admission only at the corresponding stage and admit all items at the other. StageAdmit applies learned admission at both stages. All three backbones use the same frozen predictor, trained on GLM-5 supervision.

5.2 MAIN RESULTS

StageAdmit leads all six metrics for each backbone in Table 1. With GLM-5, it obtains 79.3 WideSearch Item-F1 and 76.0 GISA table Item-F1. These scores exceed SearchOS by 1.5/1.6 points and controlled Share-all by 1.8/1.9 points (Table 2). With GPT-5.1, StageAdmit scores 76.5/73.5, exceeding SearchOS by 1.9/2.3 points. With Qwen3.8-27B, it scores 57.2/52.4, exceeding SearchOS by 2.4/0.7 points.

5.3 ABLATION STUDY

Table 2 reports the controlled admission comparisons on all three backbones and two GLM-5 predictor ablations. Shared-head keeps the stage marker but removes the residual heads; the ranking-loss ablation keeps both residual heads and sets $\alpha = 0$ in Equation 4.

Synthesis admission yields the larger quality gain on every backbone. On GLM-5, adding execution admission to Syn-only raises both scores by a further 0.7 points while cutting tool calls from 222 to 191 and session time from 951s to 841s (Table 4). On GPT-5.1 and Qwen3.8-27B, execution admission alone adds little quality: Exec-only stays within 0.2 Item-F1 of Share-all on both benchmarks. StageAdmit still exceeds Syn-only on every backbone and benchmark. Item-F1 on these two backbones is lower than on GLM-5, the backbone used to train the predictor.

Removing the residual heads lowers the GLM-5 WideSearch and GISA scores by 1.2 points each; removing the ranking loss lowers them by 0.7 and 0.8 points. These comparisons support the residual parameterization and auxiliary synthesis objective.

Reusing one supervision signal across stages. We also train on one stage's labels and use that stage's predictions at both stages. Execution-only supervision yields 76.3 WideSearch Item-F1 and 72.9 GISA table Item-F1. Synthesis-only supervision yields 77.9/74.8, compared with StageAdmit's 79.3/76.0. Both variants apply admission at both stages; Exec-only and Syn-only each apply it at just one. The results favor joint supervision from both stages. However, label coverage also differs, so this comparison does not isolate the effect of stage matching.

Table 1: Main results on **WideSearch** (200 questions) and **GISA** (373 questions), averaged over four independent runs (Avg@4); subscripts give the standard deviation across runs. Baselines are reproduced from their open-source implementations under the same protocol. Controlled admission results appear in Table 2.

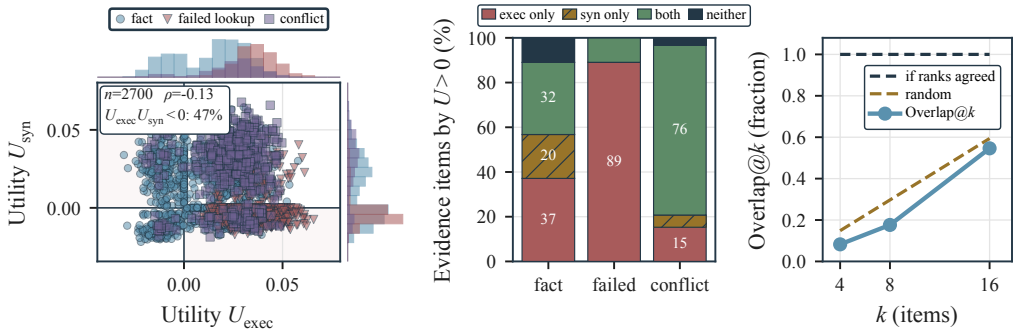
		WideSearch		GISA			
Backbone	**Method**	Item F1↑	Row F1↑	Table Item F1↑	Table Row F1↑	Set F1↑	List F1↑
GLM-5	ReAct	$70.4_{\pm 0.3}$	$48.1_{\pm 0.4}$	$72.3_{\pm 0.2}$	$54.4_{\pm 0.4}$	$57.7_{\pm 0.5}$	$64.6_{\pm 0.4}$
	Plan-and-Solve	$72.7_{\pm 0.2}$	$49.1_{\pm 0.2}$	$68.7_{\pm 0.3}$	$47.5_{\pm 0.3}$	$60.4_{\pm 0.4}$	$50.5_{\pm 0.5}$
	Table-as-Search	$72.9_{\pm 0.3}$	$49.0_{\pm 0.2}$	$70.9_{\pm 0.2}$	$50.8_{\pm 0.2}$	$57.2_{\pm 0.4}$	$52.5_{\pm 0.5}$
	A-MapReduce	$73.4_{\pm 0.2}$	$48.4_{\pm 0.3}$	$70.0_{\pm 0.2}$	$49.6_{\pm 0.3}$	$60.2_{\pm 0.4}$	$54.5_{\pm 0.5}$
	Web2BigTable	$71.3_{\pm 0.1}$	$52.7_{\pm 0.3}$	$65.6_{\pm 0.1}$	$42.4_{\pm 0.2}$	$55.3_{\pm 0.4}$	$63.8_{\pm 0.4}$
	SearchOS	$77.8_{\pm 0.2}$	$53.6_{\pm 0.2}$	$74.4_{\pm 0.1}$	$56.5_{\pm 0.2}$	$72.5_{\pm 0.3}$	$66.2_{\pm 0.4}$
	StageAdmit (ours)	$\mathbf{79.3}_{\pm 0.3}$	$\mathbf{55.8}_{\pm 0.3}$	$\mathbf{76.0}_{\pm 0.2}$	$\mathbf{58.7}_{\pm 0.3}$	$\mathbf{75.8}_{\pm 0.5}$	$\mathbf{67.2}_{\pm 0.5}$
GPT-5.1	ReAct	$65.1_{\pm 0.3}$	$44.8_{\pm 0.3}$	$67.2_{\pm 0.3}$	$47.1_{\pm 0.2}$	$60.3_{\pm 0.3}$	$59.1_{\pm 0.4}$
	Plan-and-Solve	$68.8_{\pm 0.2}$	$48.0_{\pm 0.3}$	$64.8_{\pm 0.2}$	$43.7_{\pm 0.3}$	$61.8_{\pm 0.4}$	$46.3_{\pm 0.5}$
	Table-as-Search	$70.4_{\pm 0.1}$	$46.4_{\pm 0.3}$	$68.2_{\pm 0.2}$	$49.1_{\pm 0.3}$	$60.5_{\pm 0.4}$	$50.9_{\pm 0.5}$
	A-MapReduce	$70.1_{\pm 0.1}$	$48.7_{\pm 0.4}$	$66.1_{\pm 0.2}$	$43.8_{\pm 0.3}$	$62.1_{\pm 0.4}$	$53.4_{\pm 0.5}$
	Web2BigTable	$67.0_{\pm 0.2}$	$51.9_{\pm 0.4}$	$60.8_{\pm 0.2}$	$36.4_{\pm 0.2}$	$58.2_{\pm 0.4}$	$58.8_{\pm 0.4}$
	SearchOS	$74.6_{\pm 0.3}$	$53.6_{\pm 0.3}$	$71.2_{\pm 0.2}$	$53.4_{\pm 0.2}$	$75.6_{\pm 0.4}$	$62.9_{\pm 0.3}$
	StageAdmit (ours)	$\mathbf{76.5}_{\pm 0.1}$	$\mathbf{54.8}_{\pm 0.2}$	$\mathbf{73.5}_{\pm 0.3}$	$\mathbf{54.7}_{\pm 0.2}$	$\mathbf{77.8}_{\pm 0.3}$	$\mathbf{64.1}_{\pm 0.4}$
Qwen3.8-27B	ReAct	$44.7_{\pm 0.2}$	$24.6_{\pm 0.4}$	$46.3_{\pm 0.3}$	$30.4_{\pm 0.4}$	$40.9_{\pm 0.4}$	$39.5_{\pm 0.5}$
	Plan-and-Solve	$49.6_{\pm 0.3}$	$29.1_{\pm 0.3}$	$41.8_{\pm 0.2}$	$22.7_{\pm 0.3}$	$43.7_{\pm 0.4}$	$27.4_{\pm 0.4}$
	Table-as-Search	$47.2_{\pm 0.2}$	$26.8_{\pm 0.4}$	$49.1_{\pm 0.3}$	$27.9_{\pm 0.4}$	$40.3_{\pm 0.4}$	$31.8_{\pm 0.4}$
	A-MapReduce	$51.4_{\pm 0.3}$	$27.4_{\pm 0.4}$	$44.5_{\pm 0.2}$	$25.1_{\pm 0.3}$	$43.5_{\pm 0.5}$	$34.2_{\pm 0.4}$
	Web2BigTable	$46.1_{\pm 0.2}$	$31.5_{\pm 0.2}$	$39.6_{\pm 0.2}$	$18.6_{\pm 0.4}$	$38.3_{\pm 0.5}$	$38.7_{\pm 0.5}$
	SearchOS	$54.8_{\pm 0.1}$	$34.2_{\pm 0.3}$	$51.7_{\pm 0.2}$	$33.8_{\pm 0.3}$	$55.2_{\pm 0.4}$	$41.6_{\pm 0.3}$
	StageAdmit (ours)	$\mathbf{57.2}_{\pm 0.3}$	$\mathbf{35.1}_{\pm 0.4}$	$\mathbf{52.4}_{\pm 0.1}$	$\mathbf{35.6}_{\pm 0.3}$	$\mathbf{56.0}_{\pm 0.4}$	$\mathbf{44.1}_{\pm 0.4}$

Table 2: Controlled admission ablations (Avg@4). WS is WideSearch Item-F1; GISA is table Item-F1. Agents, tools, budgets, and writer are fixed within each backbone. GPT-5.1 and Qwen3.8-27B use the predictor trained on GLM-5 supervision. The last two rows are GLM-5 only.

	GLM-5		GPT-5.1		Qwen3.8-27B	
Variant	WS↑	GISA↑	WS↑	GISA↑	WS↑	GISA↑
Share-all	77.5	74.1	74.3	70.9	54.6	51.5
Exec-only	77.8	74.5	74.4	71.1	54.7	51.6
Syn-only	78.6	75.3	75.6	72.4	56.2	52.1
StageAdmit (ours)	**79.3**	**76.0**	**76.5**	**73.5**	**57.2**	**52.4**
Shared-head	78.1	74.8	–	–	–	–
StageAdmit w/o ranking loss	78.6	75.2	–	–	–	–

5.4 Stage-Specific Behavior

We analyze stage disagreement on a held-out paired sample that is disjoint from training and is not used to select thresholds: 50 official WideSearch and 50 official GISA questions drawn uniformly at random (100 questions total). Matching question identity and item content yields 2,700 evidence-item pairs (about 25–30 items per question). Let n_q be the number of paired items for question q; every analyzed question has $n_q \geq 16$. For each item and stage, we average four keep-mask comparisons using Equation 1, with the pre-intervention context fixed. Figure 3 reports signs, rank correlation, and Top-k overlap, the fraction of the k highest-utility items shared by the two stage rankings. The means are used only for analysis: training targets remain the `width_20k` single comparisons (3,680 execution / 5,140 synthesis), and inference uses predicted $\widehat{U}_r$.



(a) Same evidence item, two utilities (b) Utility signs (c) Ranking overlap

Figure 3: Paired-stage analysis uses four-repeat keep-mask means. (a) Mean U_{exec} versus mean U_{syn} on 2,700 evidence items from 100 analysis questions (50 WideSearch + 50 GISA). (b) Combinations of utility signs ($U_r > 0$ or not) for those means, by post-hoc evidence-item type. (c) Top-k ranking overlap of the four-repeat means on the same questions.

Figure 3(a) shows weak rank association between the four-repeat mean execution and synthesis utilities (Spearman $\rho \approx -0.13$), with strictly opposite signs, $U_{\text{exec}}U_{\text{syn}} < 0$, in 47% of pairs ($1269/2700$). Removing the tool-cost term leaves 46% opposite signs ($1242/2700$) and $\rho \approx -0.10$ on the same evidence items (Appendix D). Thus, the small execution-cost adjustment alone does not explain the observed disagreement. WideSearch-only and GISA-only opposite-sign rates are 48% and 46%, within one percentage point of the pooled 47%.

Utility is positive only for execution in 48.5% of items, only for synthesis in 11.5%, for both in 33.5%, and for neither in 6.5%. The same stage dependence appears across evidence-item types in Figure 3(b). Failed lookups have positive utility only for execution in 89% of their paired cases, consistent with their potential to avoid repeated search. Conflicts have positive utility at both stages in 76% of cases. Evidence items with positive synthesis utility but nonpositive execution utility further motivate retaining items independently of execution admission.

The rankings diverge as well: Figure 3(c) shows 18% Overlap@8 between execution and synthesis utilities (Overlap@4/16 are 8%/55%), and the quality-only rankings give a similarly low 18% Overlap@8 on the same questions. Random reference overlaps average k/n_q (about 15%/30%/59% at $k = 4/8/16$). Appendix D reports within-stage split-half reliability and defines the pools, tie handling, and random reference. Together with the sign analysis, this shows that the two stages prioritize different evidence even within the same questions.

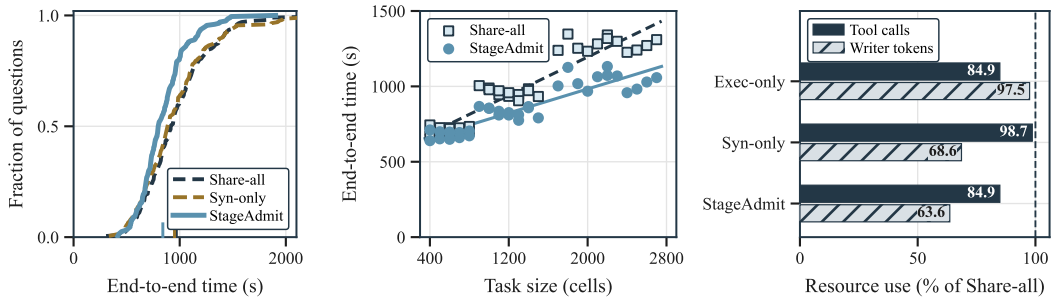
5.5 Inference Efficiency

We compare GLM-5 policies on WideSearch. Resource measurements use 200 sessions from one of the four quality runs (Figure 4; numerical summaries in Appendix D).

StageAdmit raises Item-F1 from 77.5 to 79.3. In the resource run, it cuts mean time from 968s to 841s (13.1%, including predictor inference), tool calls from 225 to 191, and writer tokens from 12.1k to 7.7k (36.4%). The two stages reduce different resources (Figure 4(c)). Exec-only cuts tool calls by 15.1% and writer tokens by 2.5%; Syn-only cuts them by 1.3% and 31.4%, respectively. Exec-only and StageAdmit both average 191 tool calls, but Exec-only scores 77.8 Item-F1 and uses more writer input.

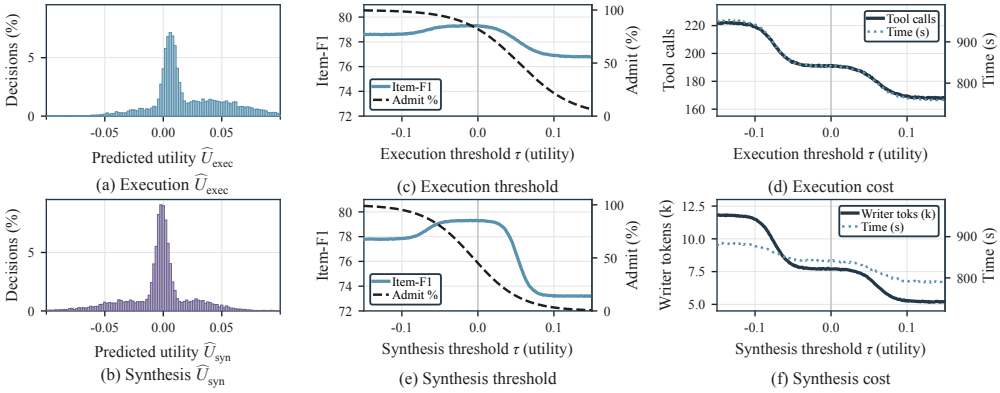
Figure 4(b) uses answer-table size (rows×columns) as a task-size proxy. In this 36-question subset, StageAdmit completes questions faster than Share-all at comparable sizes (Appendix D). Figure 5 examines the deployed $\widehat{U}_r > 0$ rule (Equation 5) using predicted-score distributions and sweeps that vary one stage's threshold at a time.

Small threshold changes around zero have little effect on Item-F1 in this run. Higher thresholds reduce admission and resource use but eventually hurt quality, especially at synthesis. These sweeps describe sensitivity within one run and are not used to select thresholds.



(a) Completion-time CDF (b) Time by task size (c) Stage resources

Figure 4: WideSearch efficiency with GLM-5 in one run. (a) Completion-time CDF over 200 questions per method. (b) Time versus task size for 36 questions, with least-squares fits (Appendix D). (c) Tool calls and writer input tokens, normalized separately to Share-all (=100%, dashed line); lower is better. Means appear in Table 4.



(a) Execution $\widehat{U}_{\text{exec}}$ (c) Execution threshold (d) Execution cost

(b) Synthesis $\widehat{U}_{\text{syn}}$ (e) Synthesis threshold (f) Synthesis cost

Figure 5: GLM-5 WideSearch threshold analysis (one run). (a,b) Predicted execution and synthesis utility distributions. (c,e) Item-F1 and admission rate; (d,f) resource use. Each sweep varies one threshold with the other at zero. Vertical lines mark zero.

5.6 Limitations

StageAdmit is evaluated in one search framework on two broad-search benchmarks. Within this scope, its quality gains over Share-all hold with all three backbones and on both benchmarks. Its keep-mask labels are measured under fixed continuation and evidence-order protocols, which may differ from contexts reached during repeated deployment decisions. Resource results come from one GLM-5 WideSearch run; small Avg@4 quality gaps should be interpreted as mean differences.

6 Conclusion

Our measurements on WideSearch and GISA showed that the same evidence item often had different utility for continued search and final-answer generation: nearly half of paired evidence items had opposite execution and synthesis utility signs. We therefore proposed StageAdmit, which kept every item in the evidence pool and learned two admission decisions from stage-specific keep-mask utilities: one for sharing with search agents and one for building the writer's evidence.

Across three backbones on both benchmarks, StageAdmit outperformed all evaluated baselines while reducing completion time and writer input. Extending these labels to other search frameworks and to repeated online decisions is a natural next step.

AI USE STATEMENT

Generative AI tools were used to edit the writing of this paper for clarity and readability. The authors reviewed the edited text.

REPRODUCIBILITY STATEMENT

Appendix A details supervision and training. Section 5.1 defines the quality metrics and Avg@4 protocol. Table 2 reports the controlled admission ablations. Appendix D gives paired-stage robustness, resource summaries, and the task-size subset.

REFERENCES

Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. In *Findings of the association for computational linguistics: ACL 2024*, pp. 2318–2335, 2024.

Mingju Chen, Guibin Zhang, Heng Chang, Yuchen Guo, and Shiji Zhou. A-MapReduce: Executing wide search via agentic mapreduce. *arXiv preprint arXiv:2602.01331*, 2026a.

Yanqiao Chen, Dongsheng Hou, Yuhan Rui, Zhen Cao, and Yepang Liu. Shapley context pruning: A cooperative game perspective for context reranking and pruning. *arXiv preprint arXiv:2607.16209*, 2026b.

Jiale Deng, Yanyan Shen, Ziyuan Pei, Youmin Chen, and Linpeng Huang. Influence guided context selection for effective retrieval-augmented generation. *Advances in Neural Information Processing Systems*, 38:29225–29252, 2025.

Joseph Fioresi, Parth Parag Kulkarni, Ashmal Vayani, Song Wang, and Mubarak Shah. Learning to share: Selective memory for efficient parallel agentic systems. *arXiv preprint arXiv:2602.05965*, 2026.

Revanth Gangi Reddy, Sagnik Mukherjee, Jeonghwan Kim, Zhenhailong Wang, Dilek Hakkani-Tür, and Heng Ji. Infogent: An agent-based framework for web information aggregation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5760–5773, 2025.

GLM-5 Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chengxing Xie, et al. GLM-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Yuxuan Huang, Yihang Chen, Zhiyuan He, Yuxiang Chen, Ka Yiu Lee, Huichi Zhou, Weilin Luo, Meng Fang, and Jun Wang. Web2BigTable: A bi-level multi-agent LLM system for internet-scale information search and extraction. *arXiv preprint arXiv:2604.27221*, 2026.

Tian Lan, Felix Henry, Bin Zhu, Qianghuai Jia, Junyang Ren, Qihang Pu, Haijun Li, Longyue Wang, Zhao Xu, and Weihua Luo. Table-as-Search: Agentic information seeking is table completion. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 15088–15107, 2026.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks. *arXiv preprint arXiv:2604.11753*, 2026.

Jun Liu, Zhenglun Kong, Changdi Yang, Fan Yang, Tianqi Li, Peiyan Dong, Joannah Nanjekye, Hao Tang, Geng Yuan, Wei Niu, et al. RCR-Router: Efficient role-aware context routing for multi-agent LLM systems with structured memory. *arXiv preprint arXiv:2508.04903*, 2025.

OpenAI. Introducing GPT-5.1 for developers, 2025. URL `https://openai.com/index/gpt-5-1-for-developers/`.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL `https://qwen.ai/blog?id=qwen3.8`.

Yifan Shi, Wei Zhang, and Tianyi Cui. A programming paradigm for spatiotemporal composability. *arXiv preprint arXiv:2608.25512*, 2026.

Zhipeng Song, Yizhi Zhou, Xiangyu Kong, Jiulong Jiao, Xinrui Bao, Xu You, Xueqing Shi, Yuhang Zhou, and Heng Qi. Less is more for RAG: Information gain pruning for generator-aligned reranking and evidence selection. *arXiv preprint arXiv:2601.17532*, 2026.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2609–2634, 2023.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. WideSearch: Benchmarking agentic broad info-seeking. In *International Conference on Learning Representations*, 2026.

Zelai Xu, Zhexuan Xu, Ruize Zhang, Chunyang Zhu, Shi Yu, Weilin Liu, Quanlu Zhang, Wenbo Ding, Chao Yu, and Yu Wang. WideSeek-R1: Exploring width scaling for broad information seeking via multi-agent reinforcement learning. *arXiv preprint arXiv:2602.04634*, 2026.

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

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

Hengran Zhang, Keping Bi, Jiafeng Guo, Jiaming Zhang, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. Distilling a small utility-based passage selector to enhance retrieval-augmented generation. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pp. 22–30, 2025.

Yuyao Zhang, Junjie Gao, Zhengxian Wu, Jiaming Fan, Jin Zhang, Shihan Ma, Yao Yao, Weiran Qi, Chuyan Jin, Guiyu Ma, et al. SearchOS-V1: Towards robust open-domain information-seeking agent collaboration. *arXiv preprint arXiv:2607.15257*, 2026a.

Zhen Zhang, Liangcai Su, Zhuo Chen, Xiang Lin, Haotian Xu, Simon Shaolei Du, Kaiyu Yang, Bo An, Lidong Bing, and Xinyu Wang. Argus: Evidence assembly for scalable deep research agents. *arXiv preprint arXiv:2605.16217*, 2026b.

Siyuan Zhu, Chengdong Xu, Kaiqiang Ke, and Chao Yu. Context-Picker: Dynamic context selection using multi-stage reinforcement learning. *arXiv preprint arXiv:2512.14465*, 2025.

Yutao Zhu, Xingshuo Zhang, Maosen Zhang, Jiajie Jin, Liancheng Zhang, Xiaoshuai Song, Kangzhi Zhao, Wencong Zeng, Ruiming Tang, Han Li, et al. GISA: A benchmark for general information-seeking assistant. *arXiv preprint arXiv:2602.08543*, 2026.

A Supervision and Training

Collection. We collect keep-mask labels from 418 synthetic WideSeek-R1 `width_20k` tasks (Xu et al., 2026): 3,680 execution pairs and 5,140 synthesis pairs. One pair is one evidence item in one stage-specific context. Official WideSearch and GISA questions are used only for evaluation and do not overlap with these tasks. Execution pairs restore the same live search checkpoint with the item shared or isolated from its recipient. Both branches share subsequently produced evidence items under Share-all; their terminal answer scores and subsequent search and page-opening counts give $U_{\text{exec}} = \Delta Q_{\text{exec}} - \lambda \Delta T$, with $\lambda = 10^{-4}$ and $\Delta T = T^{+} - T^{-}$. Synthesis pairs visit a frozen evidence pool in creation-time and item-identifier order. Each item is compared against the current answer evidence panel in fresh writer calls. It is appended only if its observed $\Delta Q_{\text{syn}} > 0$. Invalid branch outputs or scoring failures are excluded rather than assigned zero utility. The writer never receives reference answers.

Repeated analysis. Training targets use one observed keep-mask comparison per example on the `width_20k` pool above. Figure 3 instead analyzes 2,700 matched evidence items from 100 held-out questions (50 WideSearch + 50 GISA), averaging four comparisons per item with each stage's pre-intervention context fixed. These means are used only for the paired-stage analysis: they do not replace training targets and are not used to choose thresholds. Future model and web outputs are not deterministic after checkpoint restoration.

A.1 Predictor Objective and Configuration

The Huber regression objective, with equal weights on the two stages, is given in Equation 4. With an empty admitted evidence sequence, its auxiliary term ranks a positive-utility item x^{+} above a zero-utility item x^{0} from the same question:

$$\mathcal{L}_{\text{empty}} = \mathbb{E}_{(x^{+},x^{0})}\left[\max\left(0, m - \widehat{U}_{\text{syn}}(x^{+}) + \widehat{U}_{\text{syn}}(x^{0})\right)\right]. \tag{6}$$

We use Huber transition parameter 1, $\alpha = 1$, $m = 0.02$, and 16 ranking pairs per update. Each step has eight examples per stage, resampling the smaller stage dataset. Training uses AdamW at 5×10^{-5} for six epochs, with gradient norm clipping at 1. The BGE-M3 encoder is frozen; a projection, role embeddings, rank-16 LoRA adapters on Qwen3-0.6B, and scalar output heads are trained. The predictor remains frozen for all benchmark evaluations.

B Search Framework

Our search pipeline uses a coordinator-worker-writer flow. A coordinator breaks the question into coverage gaps and schedules workers. Workers search and place every new evidence item in the evidence pool. Execution admission decides, while search is still running, whether that item enters other workers' search evidence panels. After search ends, synthesis admission walks the full pool and builds the writer's answer evidence panel. The writer then drafts the final answer from that panel. Rejected items stay in the pool at both stages; they are simply not shown to other agents or admitted for writing. Illustrative prompts for these three roles appear in Appendix C.

C Prompt Examples

We provide illustrative prompts to explain the coordinator, worker, and writer roles. These examples are not the complete prompts used in the experiments. The writer example illustrates table-format tasks.

C.1 COORDINATOR

Coordinator prompt example

```
You are the Coordinator for a multi-agent web search session. You
    break the user question into coverage gaps, schedule Worker agents
    , and decide when search is complete. You do not browse the web
    yourself.

Runtime as-of date: <RUNTIME_AS_OF_DATE>. Treat it as an upper bound
    on completed real-world events unless the user asks for forecasts.
     Resolve relative time phrases against this date; prefer explicit
    ISO dates in task briefs. Never present announcements, targets, or
     plans as completed facts.

Agent and skill catalogs, and your toolset, are injected at runtime:
<AGENT_CATALOG>
<SKILL_CATALOG>
<COORDINATOR_TOOLSET>

Core loop.
1) Gap analysis. For non-trivial queries, first enqueue an explore-
    style worker to clarify entity scope, ambiguous terms, and a
    candidate list. Skip only for a single-entity, single-attribute
    lookup.
2) Coverage schema. Represent the answer as one or more entity tables
    with stable primary keys and attribute columns. Separate entity
    types and granularities; do not design tables around the user's
    display layout. Seed known entities from explore results; never
    invent seeds from parametric knowledge.
3) Worker dispatch. Enqueue self-contained task briefs (concrete
    targets, format constraints, source hints from prior rounds).
    Parallelize disjoint gaps up to free slots. After each check,
    backfill empty or conflicting cells, add newly discovered entities
    , and stop redundant tasks.
4) Stopping. Before ending search, audit expected row scope against
    collected rows. If a whole dimension slice is empty and budget
    remains, dispatch for that gap. Do not fill gaps from parametric
    knowledge.

Evidence and admission (system-managed; not your tools).
- Every evidence item a Worker produces is written into the shared
    archive.
- During search, execution admission decides whether a new item is
    shared into other Workers' search evidence panels. Rejected items
    remain in the archive.
- After search ends, synthesis admission walks the full archive and
    builds the writer's selected evidence sequence. You do not compose
     the final answer; the Writer does, using only that sequence.

When dispatching, attach up to three skills from the catalog when
    helpful. Prefer concrete briefs over vague "fill whatever is
    missing" instructions.
```

C.2 WORKER

Worker prompt example

```
You are a Worker: a web-search specialist. The Coordinator dispatches
    you with one natural-language brief targeting one or a few
    coverage cells (entity x attribute). Find grounded answers
    efficiently; work autonomously and never ask clarifying questions.
```

```
     If the brief is ambiguous, adopt the most reasonable reading,
    state it in your summary, and proceed.

Tools (injected at runtime):
{toolset}

Optional skills attached to this dispatch appear under "Loaded Skills"
     below this prompt; if a skill conflicts with the generic rules,
    the skill wins.

Evidence handling.
- Report only facts you observed on pages you opened or from tool
    results. Never invent URLs, numbers, or quotes; label unverified
    inferences.
- Every new evidence item you produce is written into the shared
    archive by the system.
- You do not control whether other Workers see your findings:
    execution admission decides sharing into other Workers' search
    evidence panels while search is running. Rejected items stay in
    the archive for later synthesis.

Search playbook.
- Never paste the full user question as a query. Use precise keywords:
     exact-phrase + site lock when the domain is known; entity +
    attribute (2-4 words); then synonym/angle/language shifts; then
    broader search plus on-page find.
- Include the year for date-sensitive facts. Prefer source hints or
    URLs from the Coordinator brief.
- On failure, diagnose before retrying (too many keywords, wrong term/
    language, paywall, niche source). Stop after three genuinely
    different angles (different source types or languages, not
    rewordings).
- Read surgically: find the relevant header or span, then jump; do not
     scroll entire pages hoping to stumble on a number.
- Stop early on a clear primary/official or authoritative secondary
    reading. For news/blogs, note source quality; if sources conflict,
     report both with provenance rather than opening endless tie-
    breakers unless the brief asks for verification.

Stay in lane. Mention useful side gaps in your final summary so the
    Coordinator can dispatch another Worker; do not chase them
    yourself.

Final summary (natural language, no tool call), answer-first:
- Answer: value(s) or explicit "not found", with source named.
- Trail: queries tried, pages read, failed routes (paywall / 403 /
    absent).
- Gaps: missing facets or conflicts worth a follow-up dispatch.
The Coordinator reads this message and cannot see raw tool outputs---
    make it self-contained.
```

C.3 WRITER

Writer prompt example

```
You are the Writer for a completed multi-agent search session. Search
    is finished. Synthesis admission has already walked the full
    archive and produced a selected evidence sequence; you receive
    only that sequence (shown below as candidate baskets). Rejected
    archive items are omitted from your input on purpose---do not try
    to recover them from prior knowledge.

Write ONE markdown pipe table that answers the user question.
```

```
Rules:
- Use ONLY values from the selected evidence sequence / candidate
    lists below.
- For each cell, pick among that cell's candidates. You may copy a
    value verbatim or keep a list cell as distinct candidates joined
    by '; '.
- If no candidate is adequate, leave the cell empty. Do not invent
    facts, URLs, dates, or numbers.
- Do not cite sources, hedge, or write prose. Output only the table (
    optionally wrapped in a markdown code fence).
- Column names should match the user's requested table if specified;
    otherwise use the schema column names.

User question:
<USER_QUESTION>

Selected evidence sequence (candidate baskets):
<CANDIDATE_BASKETS>
```

D Additional Diagnostics

Paired-stage analysis. Figure 3 uses four-repeat mean utilities for the same 2,700 evidence items from 100 analysis questions (50 WideSearch + 50 GISA). A strict sign disagreement requires $U_{\text{exec}}U_{\text{syn}} < 0$; zeros do not count. Each stage evaluates the item in its own context, so these statistics measure disagreement under the two stage-specific protocols, not an effect of the recipient alone. For a question with n_q matched items, Overlap@k is the intersection of the two top-k utility sets divided by k, with ties resolved by item order. Figure 3(c) averages overlap across all 100 questions ($n_q \geq 16$ for every question; mean $n_q = 27$); its random reference averages k/n_q. The post-hoc type groups in panel (b) contain 785 failed lookups and 458 conflicts; 699 and 348, respectively, have the positive-utility pattern discussed in the main text ($699/785 = 89\%$ and $348/458 = 76\%$).

Table 3: Paired-stage statistics on the same four-repeat means. The quality-only row removes the execution tool-cost term for analysis, without changing training targets.

Comparison	Opposite signs	Spearman ρ	Overlap@8
$U_{\text{exec}}, U_{\text{syn}}$	47.0% (1269/2700)	-0.133	17.6%
$\Delta Q_{\text{exec}}, \Delta Q_{\text{syn}}$	46.0% (1242/2700)	-0.097	18.1%

Same-stage split-half reliability. To check whether the 47% cross-stage opposite-sign rate is measurement noise, we split the four keep-mask repeats into two halves of two and compare within-stage half-means. Same-stage Spearman correlations average $\rho \approx 0.54$ (Spearman-Brown corrected 0.70), and the same-stage strict opposite-sign rate is 23%—well below the cross-stage 47%. Cohen's κ on positive labels is negative across stages ($\kappa \approx -0.13$) but positive within stage ($\kappa \approx 0.44$). A question-cluster bootstrap (500 resamples) gives a 95% CI of $[45.2\%, 48.8\%]$ for the cross-stage opposite-sign rate and $[0.62, 0.71]$ for (same-stage ρ − cross-stage ρ), which excludes 0.

WideSearch resources. Table 4 gives the single-run resource means underlying Figure 4. Each policy uses the same 200 WideSearch questions; quality comparisons use Avg@4 and are reported in Table 2.

Table 4: Resource means over 200 GLM-5 WideSearch sessions in one run. Tool calls combine search and page openings; writer tokens are input tokens.

Policy	Tool calls↓	Session s↓	Writer toks↓
Share-all	225	968	12.1k
Exec-only	191	883	11.8k
Syn-only	222	951	8.3k
StageAdmit	191	841	7.7k

Task-size plot. Figure 4(b) plots 36 questions randomly selected from the official 200-question WideSearch set. The same 36 questions are used for Share-all and StageAdmit, with one end-to-end time per question and policy. Task size is rows$\times$columns of the reference table. These per-question points are distinct from the 200-question resource summaries in Table 4.