TRACE: TRAJECTORY SELECTION FOR PARALLEL SCALING OF SEARCH AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Parallel scaling improves search agents by generating multiple candidate rollouts for the same query, yet a correct answer may already be present in the pool and still be missed by final-answer voting. We formulate this consolidation stage as *trajectory selection* and introduce TRACE (Trajectory Ranking with Aggregated Cross-Rollout Evidence), a lightweight learned selector that ranks completed trajectories using the search evidence behind their answers. TRACE preserves individual query and evidence occurrences, connects rollouts through shared content or document identity, and propagates information across these relations. Each candidate answer then reads the updated states of its own trajectory, preserving retrieval provenance while incorporating evidence from related rollouts. Trained with answer-level supervision over frozen text embeddings, TRACE returns an existing answer without additional search or autoregressive aggregation. One selector per search setting transfers across rollout policies and agent backbones without agent-specific fine-tuning, improving over voting across six WebQA policies and six long-horizon dataset–backbone combinations at $K = 16$. On Qwen2.5-14B Base/SFT WebQA pools, TRACE achieves 45.2/49.2% EM, compared with 43.9/48.0% for the strongest Qwen3-32B generative aggregators. On long-horizon FRAMES, GAIA, and BrowseComp, it reaches 78.7% average accuracy, exceeding majority voting by 3.2 percentage points. On Base WebQA pools, TRACE with only 8 rollouts comes within 0.4 points of majority voting over 64. Beyond accuracy, TRACE is substantially more efficient than the three LLM-based aggregators, SolAgg, SummAgg, and AggAgent, achieving at least $10\times$ higher processing throughput across all seven WebQA benchmarks. These results show that reusing cross-rollout search evidence provides an effective and efficient alternative to heavyweight generative aggregation for parallel search. Code is available at `https://anonymous.4open.science/r/TRACE-C718`.

1 INTRODUCTION

Search agents access external information through interleaved reasoning, search, and observation (Yao et al., 2023; Jin et al., 2025). A natural way to improve their test-time performance is *parallel scaling*: for the same question, sample multiple completed search trajectories that explore different queries, sources, reasoning paths, and candidate answers (Brown et al., 2024). This strategy can substantially increase the chance of obtaining a successful solution, but it also introduces a distinct consolidation problem: *given K completed search rollouts, which trajectory should the system ultimately return?*

Existing approaches exploit the candidate pool in different ways. Answer-level voting selects by agreement among final outputs (Wang et al., 2023), but discards most of the search process that produced them. Learned verifiers score candidate solutions from their answers or reasoning traces (Cobbe et al., 2021; Montgomery et al., 2025), yet typically evaluate candidates independently. More recent generative aggregators inspect multiple completed trajectories and invoke another large language model to synthesize a final response (Lee et al., 2026). While such methods can combine information across rollouts, they introduce an additional autoregressive reasoning stage after the expensive search trajectories have already been generated. When a correct answer is already present in the candidate pool, this extra generation may be unnecessary and can introduce new synthesis or formatting errors.

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

We instead formulate post-rollout consolidation as *trajectory selection*: directly score the K completed trajectories and return one of the existing candidates. This formulation is motivated by a simple observation: search trajectories contain information that is absent from their final answers alone (Lee et al., 2026). Independent rollouts may retrieve the same passage through different subqueries, or visit the same underlying source while exposing different passages (Appendix A.3). These shared-evidence relations provide useful signals for assessing candidate trajectories. At the same time, naively merging repeated observations would erase the query and retrieval context in which each occurrence was obtained. Effective trajectory selection therefore requires a representation that can exchange information across rollouts while preserving the provenance of each search process.

To this end, we introduce TRACE (Trajectory Ranking with Aggregated Cross-Rollout Evidence), a lightweight selector that jointly evaluates completed search trajectories. TRACE constructs an occurrence-preserving query–evidence graph over the candidate pool. Within each rollout, it retains individual search and evidence occurrences together with their retrieval structure; across rollouts, it connects observations through shared underlying evidence—shared chunk identity for retrieval-based WebQA and shared document identity for long-horizon online search. A relation-specific graph encoder propagates information through these cross-rollout connections. Each candidate answer then attends only to the updated search and evidence states of its own trajectory, yielding a representation that remains locally grounded in the process that produced the answer while being globally informed by related searches. TRACE finally scores these candidate representations and returns the highest-scoring existing answer without additional search or autoregressive answer synthesis.

An important property of this design is that trajectory selection is decoupled from trajectory generation. TRACE is trained from answer-level correctness supervision over frozen text embeddings and does not modify the underlying search agent. One selector per search setting can therefore be applied directly to candidate pools generated by different rollout policies and agent backbones without target-generator fine-tuning. We evaluate this property across six WebQA rollout policies and long-horizon BrowseComp-Plus (Chen et al., 2025), FRAMES (Krishna et al., 2024), and GAIA (Mialon et al., 2023) benchmarks with different browsing agents. At $K = 16$, TRACE consistently improves over answer-level voting and remains competitive with or stronger than Qwen3-32B generative aggregation. On Qwen2.5-14B Base/SFT WebQA pools, TRACE achieves 45.2/49.2% EM, compared with 43.9/48.0% for the strongest generative aggregators. On long-horizon search, it reaches 78.6% average accuracy, exceeding majority voting by 3.1 percentage points. Moreover, on Base WebQA pools, selecting from only 8 trajectories comes within 0.4 points of majority voting over 64, showing that better post-rollout selection can substantially reduce the sampling budget required for competitive final-answer accuracy. Our main **contributions** are summarized as follows:

- **Trajectory selection for parallel scaling.** We formulate post-rollout consolidation as a dedicated trajectory-selection problem: given K completed search rollouts, identify which existing trajectory should be returned. This separates trajectory generation from final consolidation and provides an alternative to both answer-level voting and additional generative aggregation.
- **Occurrence-preserving cross-rollout evidence reasoning.** We introduce TRACE, which preserves each query and evidence occurrence while connecting trajectories through shared underlying evidence. Cross-rollout graph propagation followed by answer-conditioned readout yields candidate representations that retain their own retrieval context while incorporating information from related searches.
- **Transferable and efficient selection across rollout generators.** We show that one selector per search setting generalizes across six WebQA rollout policies and multiple long-horizon search-agent backbones without generator-specific retraining, consistently improving over voting and large-LLM aggregation baselines. We further show that learned selection reduces the rollout budget needed for competitive accuracy and replaces autoregressive aggregation with lightweight text encoding and graph inference.

2 Related Work

Parallel scaling and trajectory aggregation. Test-time scaling improves language models through additional inference compute, including repeated sampling and parallel exploration (Brown

et al., 2024; Snell et al., 2024). For search agents, recent work extends this paradigm to multiple tool-augmented trajectories (Zhu et al., 2025; Zeng et al., 2025; Li et al., 2025). Existing consolidation strategies range from answer-level voting (Wang et al., 2023) to generative aggregation over completed trajectories. In particular, AggAgent uses another language-model agent to inspect parallel rollouts and synthesize a final answer (Lee et al., 2026), while ParallelMuse reuses information across parallel search paths (Li et al., 2025). TRACE instead formulates consolidation as *trajectory selection*: it exploits information across completed rollouts but returns one existing candidate rather than invoking an additional autoregressive generation stage.

Verification and candidate selection. Best-of-N methods and learned verifiers select among sampled candidates using outcome-level or process-level supervision (Cobbe et al., 2021; Lightman et al., 2023; Montgomery et al., 2025). Most existing verifiers score candidates independently from their answer or reasoning trace. TRACE differs by treating trajectory quality as relational: the score of one rollout can depend on evidence encountered by other rollouts, which is especially relevant for search agents that may revisit the same chunk or source through different search paths.

Graph-structured evidence reasoning. Graphs have been used to organize evidence and reasoning dependencies in multi-hop QA, fact verification, and LLM reasoning (Fang et al., 2020; Zhou et al., 2019; Cao, 2024; Hao et al., 2026). TRACE instead constructs an occurrence-preserving graph over a *set of completed search trajectories*. It preserves each query and evidence occurrence while connecting rollouts through shared chunk or document identity, enabling cross-rollout message passing without erasing retrieval provenance before answer-conditioned trajectory scoring.

3 TRACE: Trajectory Selection with Cross-Rollout Evidence

3.1 Problem Setup and Overview

Given a question q and K completed search trajectories $\mathcal{T}_K = (\tau_1, \ldots, \tau_K)$, our goal is to select one existing trajectory rather than generate another answer. Each trajectory τ_i contains its search process, retrieved evidence, and final answer a_i. After validity filtering, let $\mathcal{I}_q$ index the retained candidates and $R_q = |\mathcal{I}_q| \leq K$. A selector S returns

$$\hat{\imath} = S(q, \mathcal{T}_K), \qquad \hat{a} = a_{\hat{\imath}}, \tag{1}$$

where $\hat{\imath} \in \mathcal{I}_q$. TRACE is designed around the observation that trajectory quality can depend on evidence encountered elsewhere in the candidate pool. Different rollouts may retrieve the same chunk through different subqueries or inspect different passages from the same underlying document. At the same time, collapsing these repeated observations would erase the query and retrieval context in which each occurrence was obtained. TRACE therefore preserves individual search and evidence occurrences while explicitly connecting occurrences that share the same underlying evidence.

Figure 1 summarizes the workflow. TRACE first converts the completed trajectory pool into an occurrence-preserving query–evidence graph. A relational GNN then propagates information across within-trajectory search relations and cross-rollout shared-evidence relations. Finally, each candidate answer queries only the updated process states of its own trajectory through an answer-conditioned readout and receives a selection score. Thus, cross-rollout reasoning occurs at the evidence level, while the final decision remains trajectory-specific.

3.2 Occurrence-Preserving Query–Evidence Graph

We parse each completed interaction into search queries, returned evidence, source references, and a final answer. Nodes correspond to concrete *occurrences*: two searches that retrieve the same text still produce distinct Evidence nodes. This choice preserves the retrieval provenance of each observation, including which query produced it and where it appeared in the trajectory. Shared evidence is represented through graph relations rather than node merging.

Across both search protocols, let $s_{i,t}$ denote the t-th Subquery, or search-context, node in trajectory i, and let $e_{i,t,j}$ denote its j-th Evidence occurrence. For WebQA, an Evidence node represents a returned text chunk. For long-horizon browsing, it represents an observation produced by Open or Find. We additionally introduce Doc nodes d_m in the browsing setting to represent underlying document identity.

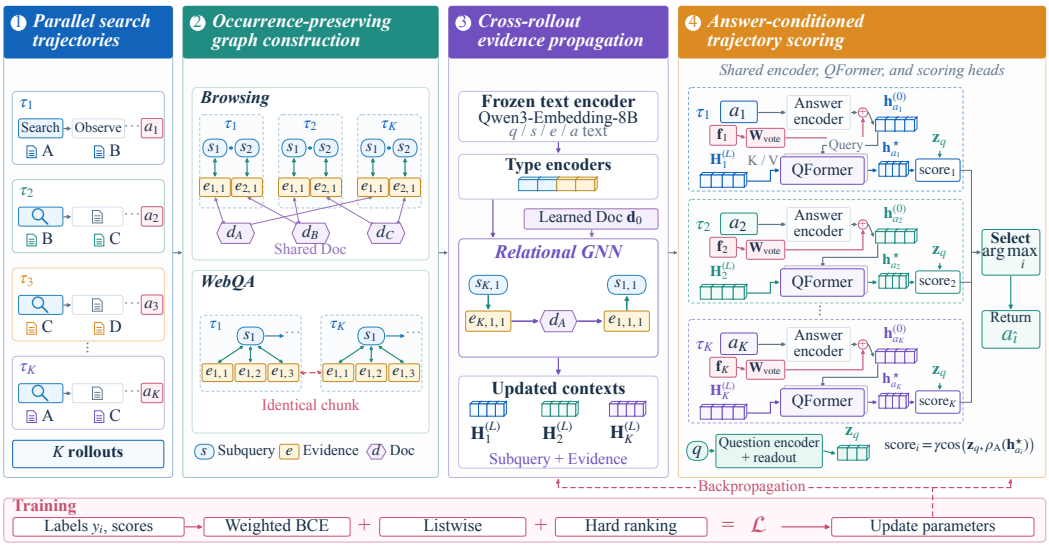

Figure 1: TRACE selects one answer from K completed search trajectories. It first constructs an occurrence-preserving query–evidence graph, connecting trajectories through identical chunks in WebQA or shared documents in long-horizon browsing. A relation-specific GNN propagates information across these links. Each candidate answer then queries the updated Subquery/Evidence states of its own trajectory through a shared answer-conditioned QFormer and receives a selection score. The highest-scoring existing answer is returned without additional search or autoregressive generation. Graphs are schematic; WebQA's Query node and some within-rollout edges are omitted for clarity.

WebQA: ranked chunk retrieval. The agent alternates reasoning with Search calls, each returning three ranked text chunks before a final answer is produced. We create one Subquery occurrence for every Search with a valid return and one Evidence occurrence for each returned chunk at ranks 1–3. Rank-specific relations connect each Subquery to its retrieved chunks in both directions, while temporal edges preserve Search order within a rollout. A Query node connects to the first Subquery of every trajectory.

When two Evidence occurrences contain the same normalized chunk, they are connected bidirectionally. We distinguish within-rollout and cross-rollout matches with separate relation types. Hence, repeated chunks can exchange search-context information while retaining their own rank, query association, and trajectory membership.

Long-horizon browsing: source-linked observations. Long-horizon agents interact with the web through Search, Open, and Find. Every Search becomes a Subquery node, and each associated Open/Find observation becomes an Evidence occurrence connected to its originating Subquery through tool-specific bidirectional relations. Successive Subqueries are linked by forward and backward temporal edges. Appendix A.1 specifies provenance tracking, including observations that do not have an explicit Search ancestor.

Exact-content matching is insufficient in this setting because different visits to the same source may expose different passages. We therefore introduce a shared Doc node d_m for each document identity and connect all Evidence occurrences from that source to it:

$$e_{i,t,j} \leftrightarrow d_m \leftrightarrow e_{i',t',j'}. \tag{2}$$

Cross-rollout information consequently travels through a two-hop Evidence–Doc–Evidence path without merging the underlying observations. Most same-document cross-rollout pairs in our analyzed browsing pools have different content identities (Appendix A.3), supporting document identity as a useful relation beyond exact-text matching. Appendix B compares document- and content-based connection schemes.

3.3 CROSS-ROLLOUT EVIDENCE PROPAGATION

Frozen Qwen3-Embedding-8B (Zhang et al., 2025) produces 4,096-dimensional embeddings $\mathbf{x}_v$ for questions, Subqueries, Evidence nodes, and answers. Trainable type-specific encoders map them to $\mathbf{h}_v^{(0)} \in \mathbb{R}^d$ with $d = 256$. For candidate answer a_i, the initial state additionally includes an answer-frequency feature:

$$\mathbf{h}_{a_i}^{(0)} = \phi_{\mathrm{A}}(\mathbf{x}_{a_i}) + \mathbf{W}_{\mathrm{vote}}\mathbf{f}_i, \qquad \mathbf{f}_i = \left[\log(1+c_i),\ \frac{c_i}{R_q}\right]^\top, \tag{3}$$

where c_i is the number of retained candidates with the same normalized answer. Doc nodes share a learned initial vector $\mathbf{d}_0$, initialized to zero, and obtain document-specific information from their neighboring Evidence nodes.

We use $L = 4$ relation-specific GraphSAGE layers (Hamilton et al., 2017). For node v at layer ℓ, messages are aggregated separately over relation types:

$$\begin{aligned} \mathbf{m}_v^{(\ell)} &= \sum_r \left(\mathbf{W}_r^{(\ell)}\,\mathrm{Mean}_{u\in\mathcal{N}_r(v)}\,\mathbf{h}_u^{(\ell)} + \mathbf{b}_r^{(\ell)}\right), \\ \mathbf{h}_v^{(\ell+1)} &= \mathrm{LN}\left(\mathbf{h}_v^{(\ell)} + \mathrm{Dropout}\left(\mathrm{GELU}(\mathbf{m}_v^{(\ell)})\right)\right). \end{aligned} \tag{4}$$

Subquery and Evidence states are updated in both settings, together with Doc states in long-horizon browsing. Query and Answer states receive no graph messages. Thus, the graph encoder performs *answer-independent evidence propagation*: matching WebQA chunks exchange information directly in one hop, whereas browsing observations communicate through their shared Doc node in two hops. The complete process-encoding details are provided in Appendix A.4.

3.4 ANSWER-CONDITIONED TRAJECTORY READOUT

After graph propagation, each candidate is evaluated from the updated process states of its own trajectory. Let

$$\mathcal{C}_i = \{v : v \text{ is a Subquery or Evidence occurrence in } \tau_i\}, \tag{5}$$

and let $\mathbf{H}_i = [\mathbf{h}_v^{(L)}]_{v\in\mathcal{C}_i}$. Although $\mathcal{C}_i$ contains only nodes from trajectory i, these states may already contain information propagated from other rollouts through shared-evidence relations. This design therefore preserves local retrieval provenance while making each trajectory globally informed.

We use the candidate answer as the query of a shared cross-attention block, which we term an answer-conditioned QFormer. A learned type embedding is added to each process state:

$$\widetilde{\mathbf{h}}_v = \mathbf{h}_v^{(L)} + \mathbf{u}_{\mathrm{type}(v)}. \tag{6}$$

For one attention head of width d_{h},

$$\alpha_{i,v} = \mathrm{softmax}_{v\in\mathcal{C}_i}\left(\frac{(\mathbf{W}_{\mathrm{Q}}\mathbf{h}_{a_i}^{(0)})^\top(\mathbf{W}_{\mathrm{K}}\widetilde{\mathbf{h}}_v)}{\sqrt{d_{\mathrm{h}}}}\right), \qquad \mathbf{z}_i = \sum_{v\in\mathcal{C}_i}\alpha_{i,v}\mathbf{W}_{\mathrm{V}}\widetilde{\mathbf{h}}_v. \tag{7}$$

Four attention heads followed by residual and feed-forward updates produce the candidate representation $\mathbf{h}_{a_i}^\star$.

Importantly, answer states do not participate in graph message passing. Candidate conditioning is introduced only at this readout stage. The computation therefore separates *cross-rollout evidence propagation* from *candidate-specific verification*:

$$\mathbf{h}_{a_i}^\star = f(a_i, \tau_i, \mathcal{T}_K). \tag{8}$$

Each representation remains locally grounded in trajectory τ_i, while indirectly incorporating evidence from related rollouts.

Finally, trainable projections ρ_{Q} and ρ_{A} map the question and updated answer into a common scoring space. Let $\mathbf{z}_q = \rho_{\mathrm{Q}}(\mathbf{h}_q^{(0)})$. We compute

$$\mathrm{score}_i = \gamma\cos\left(\mathbf{z}_q, \rho_{\mathrm{A}}(\mathbf{h}_{a_i}^\star)\right), \qquad \hat{\imath} = \arg\max_{i\in\mathcal{I}_q}\mathrm{score}_i, \qquad \hat{a} = a_{\hat{\imath}}, \tag{9}$$

where γ is learned. Identical answer strings may receive different scores because they remain associated with distinct search trajectories and process contexts. The highest-scoring existing candidate is returned directly.

3.5 LEARNING TO SELECT

Training uses answer-level correctness labels $\mathbf{y}_q = (y_i)_{i \in \mathcal{I}_q}$, where $y_i \in \{0, 1\}$ is obtained by exact match for WebQA or the available correctness annotations for long-horizon browsing. Let

$$\mathcal{P}_q = \{i \in \mathcal{I}_q : y_i = 1\}, \qquad \mathcal{N}_q = \mathcal{I}_q \setminus \mathcal{P}_q. \tag{10}$$

We optimize three complementary objectives. Weighted binary cross-entropy learns candidate-level correctness; a listwise objective concentrates probability mass on correct trajectories within the candidate pool; and a hard-ranking objective separates the highest-scoring incorrect candidate from the strongest correct candidate. The full objective is

$$\mathcal{L} = \mathcal{L}_{\text{BCE}} + \mathcal{L}_{\text{list}} + \mathcal{L}_{\text{hard}}. \tag{11}$$

BCE is averaged over candidate nodes, while the ranking terms are computed only for questions for which their required positive/negative sets are defined. Appendix A.4 provides the exact loss definitions and eligibility conditions.

Gradients jointly update the type-specific encoders, relational GNN, answer-conditioned QFormer, and scoring layers. The text embedding model and rollout-generating search policy remain fixed. Consequently, TRACE can be trained as a separate post-rollout selection module and then applied to candidate trajectories generated by different search policies or agent backbones without modifying those generators.

4 EXPERIMENTS

We organize our experiments around four research questions (RQs). **RQ1:** Can TRACE select better final answers from the same K completed rollouts than answer-level heuristics and LLM-based generative aggregation? **RQ2:** Does a selector trained once per search setting transfer across rollout policies, model scales, and agent backbones without generator-specific fine-tuning? **RQ3:** Do occurrence preservation, cross-rollout evidence propagation, and answer-conditioned readout each contribute to the gains of TRACE? **RQ4:** Can a fixed selector exploit different rollout budgets, and can better selection reduce the number of rollouts needed for competitive accuracy?

4.1 EXPERIMENTAL SETUP

Benchmarks and metrics. We evaluate TRACE in two search regimes. Retrieval-based WebQA uses 3,125 questions from NQ (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), TriviaQA (Joshi et al., 2017), PopQA (Mallen et al., 2023), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), and Bamboogle (Press et al., 2022), and reports question-weighted exact match (EM). Long-horizon online search uses BrowseComp-Plus (Chen et al., 2025), FRAMES (Krishna et al., 2024), and GAIA (Mialon et al., 2023), with accuracy taken from the saved Qwen3-32B correctness judgments of the candidate pools. Unless stated otherwise, the long-horizon aggregate averages the two rollout backbones within each benchmark and then the three benchmarks equally.

Rollout generators and transfer setting. WebQA candidate pools come from Base, SFT, and RL variants of Qwen2.5-7B and Qwen2.5-14B, giving six rollout policies with substantially different single-rollout accuracies and candidate distributions. Long-horizon pools come from OpenResearcher-30B-A3B and gpt-oss-120B; a single long-horizon checkpoint is applied to both without agent-specific fine-tuning.

Selector training. For WebQA, we sample $K = 16$ Qwen2.5-14B trajectories per question on the NQ and HotpotQA training sets (101,323 training and 5,309 validation questions). For long-horizon search, we use existing 16-trajectory pools with correctness labels from the OpenResearcher dataset (Li et al., 2026) (2,655 training and 132 validation questions). Both settings keep only questions whose pools contain both correct and incorrect answers.

Table 1: WebQA EM (%) at $K = 16$ with Qwen2.5-14B Base and SFT rollouts. SolAgg, SummAgg, and AggAgent use Qwen3-32B. Overall is question-weighted within each rollout policy. Bold marks the best result in each column.

	NQ		HotpotQA		TriviaQA		PopQA		2Wiki		MuSiQue		Bamboogle		Overall	
Method	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT
Single rollout	29.4	41.4	27.6	36.8	49.6	64.2	35.2	38.6	24.6	31.4	10.4	18.4	31.2	45.6	29.5	38.8
Majority Voting	44.6	46.6	40.4	46.8	63.8	72.2	46.0	46.2	40.8	46.2	18.4	22.4	48.8	59.2	42.6	47.2
Weighted Voting	44.4	46.6	40.6	47.2	64.8	72.6	46.0	46.2	41.6	45.6	19.0	22.4	48.8	60.0	43.0	47.3
Fewest Tools	36.0	45.0	30.2	42.6	58.2	71.2	39.4	43.4	26.0	41.0	9.6	20.2	32.8	57.6	33.2	44.4
SolAgg	43.2	43.4	42.6	**49.0**	65.0	72.4	45.0	46.4	45.8	**48.6**	19.6	25.0	52.8	**60.8**	43.9	48.0
SummAgg	41.2	41.6	**44.2**	48.0	62.2	71.6	43.6	44.0	**48.0**	47.2	21.2	24.6	51.2	58.4	43.7	46.7
AggAgent	41.0	39.6	37.8	44.0	63.6	68.0	40.4	42.2	44.8	43.2	17.0	21.0	**60.0**	59.2	41.5	43.6
TRACE	**47.4**	**48.4**	43.2	48.2	**65.6**	**73.2**	**47.6**	**49.2**	44.8	47.2	**21.4**	**26.0**	51.2	60.0	**45.2**	**49.2**

Baselines. We compare three classes of post-rollout consolidation. *Answer-level heuristics* are majority voting (Wang et al., 2023), confidence-weighted majority voting, and Fewest Tools. *Generative aggregation* covers SolAgg, SummAgg, and AggAgent (Lee et al., 2026), all using Qwen3-32B as the aggregator regardless of the rollout generator. Single-rollout accuracy serves as a reference; Appendix A.2 gives the full protocols.

Implementation. We train for 3 epochs with AdamW (constant learning rate 3×10^{-4}, weight decay 10^{-4}, dropout 0.1) and batch sizes of 256 (WebQA) and 4 (long-horizon) question graphs. The type-specific projections, four-layer relation-specific GraphSAGE encoder, answer-conditioned QFormer, and scoring layers are optimized jointly, while the rollout policies and Qwen3-Embedding-8B stay frozen. We hold out about 5% of training questions for validation, select one checkpoint by validation performance, and keep it fixed in every rollout-budget sweep. Appendix A details filtering, graph parsing, and evaluation.

4.2 Main Results: Effective and Transferable Trajectory Selection (*RQ1, RQ2*)

WebQA across rollout policies. Table 1 compares selectors on Qwen2.5-14B Base and SFT pools, whose single-rollout accuracies differ substantially. **Observation 1: TRACE improves final-answer selection across distinct WebQA rollout distributions.** On Base pools, TRACE reaches 45.2% overall EM, 2.6 and 2.2 points above majority and weighted voting, and improves over majority voting on all seven datasets. On the stronger SFT pools, it reaches 49.2% versus 47.2% for majority voting. Appendix E shows that the same WebQA selector improves over both voting baselines on all six Qwen2.5-7B/14B Base, SFT, and RL policies.

Long-horizon search across benchmarks and agent backbones. Table 2 applies one selector to three benchmarks and two rollout generators, where trajectories involve long multi-step browsing and cross-rollout sharing is captured through document identity rather than identical chunks. **Observation 2: One long-horizon selector transfers across benchmarks and rollout generators.** TRACE reaches 78.6% average accuracy, 3.1 and 1.9 points above majority and weighted voting, and is best (including ties) in five of six settings. The largest gain over majority voting is on gpt-oss-120B BrowseComp-Plus (66.0% to 74.3%). Since one selector serves both generators, post-rollout selection can be decoupled from the model that generates the trajectories.

Observation 3: Discriminative trajectory selection can outperform large generative aggregation. The generative baselines use Qwen3-32B to read all trajectories and synthesize a new response, whereas TRACE is a compact graph selector over frozen embeddings that returns an existing candidate. Still, TRACE exceeds the strongest aggregator by 1.3/1.2 points on WebQA Base/SFT and by 1.6 points in the long-horizon setting, supporting post-rollout consolidation as selection when the pool already contains strong solutions.

Table 2: Long-horizon accuracy (%). Single rollout reports mean Pass@1 over 16 samples; other rows use $K = 16$. OR and OSS denote OpenResearcher-30B-A3B and gpt-oss-120B rollouts. All generative aggregators use Qwen3-32B. **Bold** marks the best result in each column.

	BrowseComp-Plus		FRAMES		GAIA		
Method	OR	OSS	OR	OSS	OR	OSS	Avg.
Single rollout	38.3	45.2	73.1	84.9	55.4	61.7	59.7
Majority Voting	64.5	66.0	88.4	90.4	66.0	77.7	75.5
Weighted Voting	64.5	70.2	88.4	90.6	67.0	**79.6**	76.7
Fewest Tools	64.6	72.7	86.0	89.0	51.5	76.7	73.4
SolAgg	67.0	71.4	88.6	89.0	66.0	77.7	76.6
SummAgg	**67.1**	72.4	87.6	90.2	66.0	78.6	77.0
AggAgent	58.1	62.3	76.6	76.6	66.0	73.8	68.9
TRACE	**67.1**	**74.3**	**90.6**	**91.8**	**68.9**	78.6	**78.6**

Table 3: Mechanism ablations at $K = 16$. WebQA reports EM (%).

Configuration	WebQA	Long-horizon avg.
Full TRACE	**45.2**	**78.6**
w/o cross-rollout communication	43.9	72.7
w/o GNN	43.7	75.5
Fixed-query readout	44.7	75.6

4.3 Mechanism Analysis (*RQ3*)

Table 3 tests whether the gains come from the relational mechanism of Section 3, holding candidate pools, embeddings, supervision, and optimization budget fixed. **Observation 4: Cross-rollout shared evidence is particularly important for long-horizon selection.** Removing only cross-rollout communication lowers WebQA EM from 45.2% to 43.9% and long-horizon accuracy from 78.6% to 72.7%, the largest drop among the main ablations, showing that evidence seen by other trajectories adds substantially beyond encoding each rollout independently. Different trajectories often read different passages of the same source, and document identity links such evidence in a way that answer agreement or exact passage matching cannot.

Observation 5: Graph propagation and answer-conditioned readout provide complementary gains. Removing the GNN gives 43.7/75.5 (WebQA/long-horizon), and replacing the answer-dependent query with a shared learned query gives 44.7/75.6. This supports the two-stage design: first contextualize evidence across rollouts, then let each candidate read its own globally informed context.

Appendix B reports further controls (WebQA/long-horizon). Preserving occurrences matters: merging process occurrences, identical answers, or both gives 44.1/77.1, 44.9/76.3, and 44.4/76.4, indicating that each trajectory–answer occurrence should be kept rather than collapsing candidates that share evidence or answer strings. The evidence-connection controls further validate the graph construction: removing content-identity links lowers WebQA EM to 43.8; in the long-horizon setting, no Doc or content links gives 71.1, content links alone 76.5, and Doc plus content 76.9, versus 78.6 for Doc links alone, supporting source identity as the relation for linking passages of the same document. Replacing the QFormer with graph readout gives 44.7/75.8, and two-stage QFormer training gives 44.8/75.1, so jointly optimizing process encoding and readout performs best.

4.4 Scaling and Efficiency (*RQ4*)

We apply the checkpoint trained at $K = 16$ to pools of other sizes without retraining (Figure 2). **Observation 6: A fixed selector generalizes across rollout budgets.** TRACE stays above major-

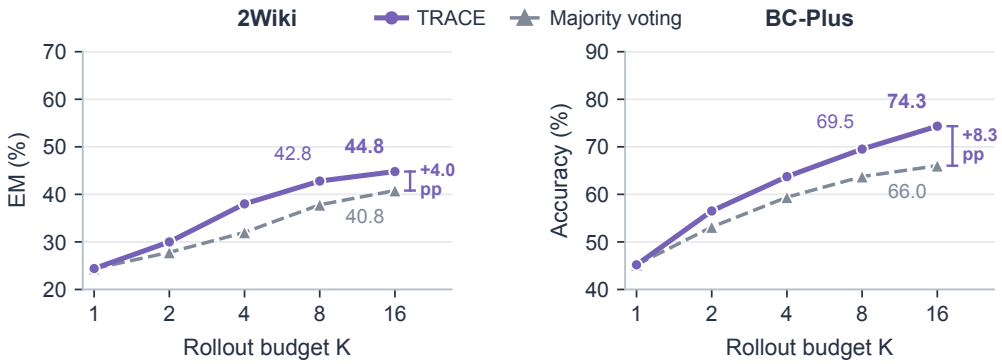

Figure 2: Final-answer performance with fixed TRACE checkpoints as the rollout budget increases. (a) 2Wiki EM with Qwen2.5-14B Base rollouts. (b) BrowseComp-Plus accuracy with gpt-oss-120B rollouts. Dashed curves show majority voting; brackets mark TRACE gains at $K = 16$.

ity voting for every $K \geq 2$, reaching 42.8% and 44.8% EM at $K = 8$ and $K = 16$ on 2Wiki, and rising from 69.5% to 74.3% on gpt-oss-120B BrowseComp-Plus. Across all WebQA datasets, EM rises from 35.2% at $K = 2$ to 45.2% at $K = 16$ and 46.3% at $K = 32$, a budget unseen in training (Appendix C). Because the checkpoint is unchanged throughout the sweep, TRACE naturally operates over candidate sets that differ in size from those seen during training.

Table 4: Post-rollout runtime on the 3,125-question SFT WebQA pool. All LLM aggregators use Qwen3-32B. GPU-hours use H20s; relative cost is normalized to TRACE.

Method	GPU-hours ↓	Relative cost
TRACE	**0.271**	1×
SolAgg	3.28	12×
SummAgg	32.96	122×
AggAgent	69.93	258×

Observation 7: Better selection can substitute for substantially more rollout generation. On WebQA Base pools, TRACE at $K = 8$ reaches 43.3% EM, essentially matching majority voting at $K = 64$ (43.6%) with one eighth of the rollouts (Appendix C).

Post-rollout processing cost. Processing all 3,125 SFT WebQA questions with TRACE takes only 0.271 H20 GPU-hours (16.2 minutes on one H20), including fresh text encoding, graph construction, and trajectory selection. As shown in Table 4, this is substantially cheaper than all three Qwen3-32B generative aggregators: SolAgg, SummAgg, and AggAgent require 3.28, 32.96, and 69.93 GPU-hours, respectively, corresponding to approximately 12×, 122×, and 258× the post-rollout cost of TRACE. Thus, even the fastest generative baseline requires more than an order of magnitude greater GPU compute, while the more elaborate aggregation pipelines incur two orders of magnitude higher cost. Appendix H provides the runtime breakdown and timing scope.

5 Conclusion

We presented TRACE, a lightweight trajectory selector for parallel scaling of search agents. TRACE preserves retrieval occurrences, connects trajectories through shared evidence, and combines cross-rollout graph propagation with answer-conditioned readout to select an existing candidate without additional search or autoregressive aggregation. Across six WebQA rollout policies and long-horizon BrowseComp-Plus, FRAMES, and GAIA evaluations, one selector per search setting consistently improves over voting and remains competitive with or stronger than Qwen3-32B generative aggregation. These results show that post-rollout consolidation can be effectively decoupled from trajectory generation and handled by a lightweight, transferable selection module.

AI USE STATEMENT

We used ChatGPT and Codex to create and edit code and to edit the manuscript for readability. We reviewed and verified all AI-assisted work and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

Sections 3 and 4.1 specify the model, objectives, and optimization settings. Appendix A provides implementation and evaluation details, Appendix B gives additional ablations, and Appendices C–H give supporting results, coverage analysis, and efficiency measurements.

All code, including the implementation, training configurations, and evaluation scripts, is available at `https://anonymous.4open.science/r/TRACE-C718`.

REFERENCES

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL https://arxiv.org/abs/2407.21787.

Lang Cao. GraphReason: Enhancing reasoning capabilities of large language models through a graph-based verification approach. In *Proceedings of the 2nd Workshop on Natural Language Reasoning and Structured Explanations (@ACL 2024)*, pp. 1–12, Bangkok, Thailand, 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.nlrse-1.1/.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, Sahel Sharifymoghaddam, Yanxi Li, Haoran Hong, Xinyu Shi, Xuye Liu, Nandan Thakur, Crystina Zhang, Luyu Gao, Wenhu Chen, and Jimmy Lin. BrowseComp-Plus: A more fair and transparent evaluation benchmark of deep-research agent, 2025. URL https://arxiv.org/abs/2508.06600.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. Hierarchical graph network for multi-hop question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8823–8838. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.710. URL https://aclanthology.org/2020.emnlp-main.710/.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://arxiv.org/abs/1706.02216.

Yu Hao, Qiuyu Wang, Cheng Yang, Yawen Li, Zhiqiang Zhang, and Chuan Shi. GNNVerifier: Graph-based verifier for LLM task planning, 2026. URL https://arxiv.org/abs/2603.14730.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580. URL https://aclanthology.org/2020.coling-main.580/.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: Training LLMs to reason and leverage search engines with reinforcement learning, 2025. URL https://arxiv.org/abs/2503.09516.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, 2017. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Satyapriya Krishna, Kalpesh Krishna, Anhad Mohananey, Steven Schwarcz, Adam Stambler, Shyam Upadhyay, and Manaal Faruqui. Fact, fetch, and reason: A unified evaluation of retrieval-augmented generation, 2024. URL https://arxiv.org/abs/2409.12941.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Yoonsang Lee, Howard Yen, Xi Ye, and Danqi Chen. Agentic aggregation for parallel scaling of long-horizon agentic tasks, 2026. URL https://arxiv.org/abs/2604.11753.

Baixuan Li, Dingchu Zhang, Jialong Wu, Wenbiao Yin, Zhengwei Tao, Yida Zhao, Liwen Zhang, Haiyang Shen, Runnan Fang, Pengjun Xie, Jingren Zhou, and Yong Jiang. ParallelMuse: Agentic parallel thinking for deep information seeking, 2025. URL https://arxiv.org/abs/2510.24698.

Zhuofeng Li, Dongfu Jiang, Xueguang Ma, Haoxiang Zhang, Ping Nie, Yuyu Zhang, Kai Zou, Jianwen Xie, Yu Zhang, and Wenhu Chen. OpenResearcher: A fully open pipeline for long-horizon deep research trajectory synthesis, 2026. URL https://arxiv.org/abs/2603.20278.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023. URL https://arxiv.org/abs/2305.20050.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. URL https://arxiv.org/abs/2212.10511.

Grégoire Mialon, Clémentine Fourrier, Craig Swift, Thomas Wolf, Yann LeCun, and Thomas Scialom. GAIA: a benchmark for general AI assistants, 2023. URL https://arxiv.org/abs/2311.12983.

Kyle Montgomery, Sijun Tan, Yuqi Chen, Siyuan Zhuang, Tianjun Zhang, Raluca Ada Popa, and Chenguang Wang. Budget-aware test-time scaling via discriminative verification, 2025. URL https://arxiv.org/abs/2510.14913.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models, 2022. URL https://arxiv.org/abs/2210.03350.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters, 2024. URL https://arxiv.org/abs/2408.03314.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 2022. URL https://arxiv.org/abs/2108.00573.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259. URL https://aclanthology.org/D18-1259/.

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. URL https://arxiv.org/abs/2210.03629.

Weihao Zeng, Keqing He, Chuqiao Kuang, Xiaoguang Li, and Junxian He. Pushing test-time scaling limits of deep search with asymmetric verification, 2025. URL https://arxiv.org/abs/2510.06135.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models, 2025. URL https://arxiv.org/abs/2506.05176.

Jie Zhou, Xu Han, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. GEAR: Graph-based evidence aggregating and reasoning for fact verification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 892–901. Association for Computational Linguistics, 2019. doi: 10.18653/v1/P19-1085. URL https://aclanthology.org/P19-1085/.

King Zhu, Hanhao Li, Siwei Wu, Tianshun Xing, Dehua Ma, Xiangru Tang, Minghao Liu, Jian Yang, Jiaheng Liu, Yuchen Eleanor Jiang, Changwang Zhang, Chenghua Lin, Jun Wang, Ge Zhang, and Wangchunshu Zhou. Scaling test-time compute for LLM agents, 2025. URL https://arxiv.org/abs/2506.12928.

A Implementation and Evaluation Details

A.1 Candidate Pools and Metrics

Training and validation use fixed question-level 5% holdouts: deterministic for WebQA and stored with the long-horizon graphs. WebQA Single rollout scores unfiltered sample 0; long-horizon Single rollout and scaling at $K = 1$ average all 16 original judged samples per question, counting failed or empty samples as wrong. At $K \geq 2$, selection takes the first K samples. WebQA retains trajectories with a final answer and retrieved evidence; long-horizon browsing retains nonempty final responses. Invalid samples are not replaced. Graph memberships and vote features are recomputed for each retained pool, and empty pools receive zero accuracy and F1 while remaining in the denominator.

WebQA EM and token F1 lowercase answers, remove punctuation and English articles, collapse whitespace, and take the best match over reference aliases. Long-horizon accuracy uses the saved Qwen3-32B judgments. WebQA aggregates weight each question equally. Long-horizon aggregates instead use $\frac{1}{3}\sum_d \frac{1}{2}\sum_b A_{d,b}$, where $A_{d,b}$ is accuracy for dataset d and backbone b; each dataset and backbone receives equal weight. This same aggregation rule applies to the long-horizon ablations. The WebQA evaluation includes 500 questions from each dataset except Bamboogle, which contributes 125.

Long-horizon evaluation uses all 830 BrowseComp-Plus questions, all 103 questions in the GAIA text-only `gaia_text` split of OpenResearcher/web-bench, and a sampled subset of 500 FRAMES questions. Question sets remain fixed across backbones, consolidation methods, ablations, and rollout budgets.

WebQA graph parsing. A retained rollout must contain a final answer and nonempty retrieved evidence. Each nonempty Search return is split into the ordered rank-1, rank-2, and rank-3 chunks. Rank markers are stored as edge types; the remaining chunk text retains its title and content. Identity normalizes Unicode to NFC and collapses whitespace, then compares text exactly. Each valid Search and each ranked return remains a separate occurrence. Empty returns create no retrieval nodes, and temporal edges follow the retained Search events. Shared text embeddings are lookup reuse, not merged graph nodes.

Browsing graph parsing. We retain nonempty final responses and their Search/Open/Find records. Every recorded Search receives a Subquery node. Open and Find results become Evidence nodes, with their original tool kind and event order. Parent-event and cursor references trace each observation to its Search ancestor; a chain with no such ancestor attaches to its own orphan Subquery instead of the most recent Search. The explicit temporal chain connects Subquery nodes in both directions. Evidence is attached to a Doc using the verified URL/view identity mapping. URL normalization resolves equivalent display and encoding forms and removes fragments, while preserving query information. Find and page-continuation views inherit their parent's source. Retained search-result views are scoped to their rollout, so a search listing cannot accidentally link unrelated webpages across trajectories. Each Evidence occurrence has one document membership. Only Subquery and Evidence states enter that rollout's attention context.

A.2 Baselines

Weighted Majority Voting sums confidence within each normalized-answer group. Confidence is estimated from the completed trajectory and final answer by its generating model, without reference answers. Fewest Tools counts Search calls for WebQA and Search/Open/Find calls for browsing. SolAgg integrates final solution texts; SummAgg first summarizes each interaction; AggAgent accesses candidates through solution, trajectory-search, and segment tools. Generative aggregation uses Qwen3-32B in the main and supplementary WebQA comparisons and for long-horizon browsing. SolAgg and SummAgg use temperature 1 and a 10,000-token output budget.

WebQA answer extraction. SolAgg retains the first complete answer tag. SummAgg retains tagged answers; when tags are absent, it accepts an explicit first-line declaration beginning with "The correct answer is:", "The answer is:", "Final answer:", or "Answer:", removing Markdown bold markers and a trailing period. For AggAgent, we take the first nonempty line of an accepted an-

swer. For completed runs without an accepted answer, we use the first saved `finish.solution` verbatim; runs without a finish call and terminated runs retain empty answers. These post-hoc rules use no reference answers to choose the extracted text. Long-horizon accuracy uses the saved correctness judgments for final responses.

A.3 Graph Statistics and Counting Units

We count the retained $K = 16$ evaluation pools for Qwen2.5-14B Base on WebQA and gpt-oss-120B on BrowseComp-Plus: 3,125 and 830 question graphs, respectively. Counts use the retained graph metadata and document identities described above. WebQA has 39,433 retained rollouts and 188,889 Evidence (chunk) occurrences; browsing has 9,509 retained rollouts and 97,781 Evidence occurrences.

For a content or document group g, let $n_{g,r}$ be its number of occurrences in rollout r. Its number of cross-rollout pairs is $\sum_{r<s} n_{g,r} n_{g,s}$, so each unordered pair is counted once and same-rollout repetitions are excluded. A shared group is a distinct content or source identity present in at least two rollouts. Table A.1 reports unweighted means over question graphs, including graphs with zero shared groups. Medians and 90th percentiles show the distribution of pair counts.

Table A.1: Cross-rollout sharing at $K = 16$. Groups and occurrence pairs are counted within each question graph. "Graphs" is the percentage containing at least one such pair.

Task	Matching rule	Groups	Occurrence pairs			Graphs (%)
		Mean	Mean	Median	P90	
WebQA	Identical chunk	7.6	243.6	228	430	99.9
Browsing	Identical evidence	15.4	462.0	314	1,080	99.0
Browsing	Same document	6.5	1,797.6	1,142.5	4,224	99.0

The graph representation determines how these pairs become messages. WebQA instantiates each identical-content pair in both directions: an average of 487.2 cross-rollout and 14.4 within-rollout directed edges per graph. Browsing uses document incidence edges, not an Evidence clique: its 117.8 Evidence occurrences per graph give 235.6 directed Evidence–Doc edges, providing two-hop paths between same-source observations. Of 1,491,965 cross-rollout same-document pairs, 1,108,528 (74.3%) have different content identities. A document observed m times requires $2m$ directed incidence edges, compared with $m(m-1)$ for a pairwise same-source clique.

A.4 Encoder and Readout Details

Frozen Qwen3-Embedding-8B (Zhang et al., 2025) supplies text embeddings $\mathbf{x}_v$. The embedding pipeline pools the last non-padding token and applies ℓ_2 normalization. A trainable encoder $\phi_\nu(\mathbf{x}) = \mathrm{GELU}(\mathrm{LN}(\mathbf{W}_\nu \mathbf{x} + \mathbf{b}_\nu))$ maps each text-bearing node type ν to dimension d, giving $\mathbf{h}_v^{(0)} = \phi_{\mathrm{type}(v)}(\mathbf{x}_v)$. Answer representations additionally incorporate answer frequency:

$$\mathbf{h}_{a_i}^{(0)} = \phi_{\mathrm{A}}(\mathbf{x}_{a_i}) + \mathbf{W}_{\mathrm{vote}} \mathbf{f}_i, \qquad \mathbf{f}_i = \left[\log(1 + c_i),\ c_i / R_q\right]^\top, \tag{A.1}$$

where c_i counts retained candidates with the same normalized answer. Doc nodes share a learned initial vector $\mathbf{d}_0$, initialized to zero, and obtain document-specific information through their neighbors. Orphan Subqueries have a zero text-embedding input and receive an additional learned indicator vector, initialized to zero; no synthetic query text is embedded.

We use relation-specific GraphSAGE (Hamilton et al., 2017) to propagate process information. For incoming relation types r and their neighbor sets $\mathcal{N}_r(v)$, layer $\ell \in \{0, \ldots, L-1\}$ computes

$$\begin{aligned} \mathbf{m}_v^{(\ell)} &= \sum_r \left(\mathbf{W}_r^{(\ell)} \, \mathrm{Mean}_{u \in \mathcal{N}_r(v)} \, \mathbf{h}_u^{(\ell)} + \mathbf{b}_r^{(\ell)} \right), \\ \mathbf{h}_v^{(\ell+1)} &= \mathrm{LN}\Big(\mathbf{h}_v^{(\ell)} + \mathrm{Dropout}\Big(\mathrm{GELU}(\mathbf{m}_v^{(\ell)}) \Big) \Big). \end{aligned} \tag{A.2}$$

Updates are synchronous, with node self-information retained by the residual. An empty relation neighborhood has a zero mean; the relation transformation still includes its bias. We use $L = 4$ layers with $d = 256$. Subquery and evidence states are updated in both variants, along with Doc states

in long-horizon browsing. Query and Answer states receive no graph messages. Cross-trajectory information travels directly between matching WebQA chunks or along the two-hop Evidence–Doc–Evidence path before answer readout.

The QFormer has one block with four heads of width 64. Each head uses the answer state as its query and the rollout's Subquery/Evidence states as keys and values. Head outputs are concatenated and projected, followed by a $256 \to 1024 \to 256$ GELU feed-forward network. Both sublayers use dropout 0.1, residual connections, and LayerNorm. If $\mathcal{C}_i$ is empty, the attention sum is zero and the residual, feed-forward, and normalization operations still apply. The question and answer readouts ρ_{Q} and ρ_{A} are separate affine $256 \to 256$ maps. Scoring uses $\gamma = \min(\exp(\eta), 100)$ with η initialized to $\log 10$.

Loss definitions. With the positive and negative candidate sets defined in Section 3.5, the three terms are

$$\begin{aligned} \ell_{\mathrm{BCE}}(i) &= -w_+ y_i \log \sigma(\mathrm{score}_i) - (1 - y_i) \log(1 - \sigma(\mathrm{score}_i)) \,, \\ \ell_{\mathrm{list}}(q) &= \log \sum_{i \in \mathcal{I}_q} e^{\mathrm{score}_i} - \log \sum_{i \in \mathcal{P}_q} e^{\mathrm{score}_i}, \\ \ell_{\mathrm{hard}}(q) &= \mathrm{softplus}\left(\max_{i \in \mathcal{N}_q} \mathrm{score}_i - \max_{i \in \mathcal{P}_q} \mathrm{score}_i \right), \end{aligned} \tag{A.3}$$

Here σ is the sigmoid and w_+ is the ratio of negative to positive training candidates. BCE is averaged over candidates and each ranking term over its eligible questions.

Loss eligibility and empty pools. WebQA averages listwise terms over questions with at least one correct candidate; long-horizon browsing uses pools containing both correct and incorrect candidates. Hard ranking applies only to mixed pools in both settings. Empty pools have no selected candidate and count as incorrect, as specified in Appendix A.1.

B Additional Ablations

Protocol. All comparisons use $K = 16$ with the same question splits, frozen text embeddings, correctness supervision, evaluation pools, and optimization budget. The controls that remove cross-rollout graph messages, bypass the GNN, or use a fixed attention query in Table 3 retain the full model's answer-frequency features. The fixed query adds a 256-dimensional learned vector; bypassing the GNN leaves the type encoders, attention readout, and scoring layers trainable. Readout membership remains local to each trajectory in these three controls.

Table B.1: Additional design comparisons at $K = 16$. Columns report WebQA EM and long-horizon average accuracy (%), using the same aggregation as Table 3. Dashes denote unevaluated comparisons.

Configuration	WebQA	Long-horizon avg.
Full TRACE	**45.2**	**78.6**
Occurrence representation		
Merge identical evidence occurrences	44.1	77.1
Merge identical answers	44.9	76.3
Merge both	44.4	76.4
Answer readout and training		
Graph readout (no attention)	44.7	75.8
Two-stage readout training	44.8	75.1
Evidence connections		
No content-identity links	43.8	—
No Doc or content-identity links	—	71.1
Content-identity links only	—	76.5
Doc and content-identity links	—	76.9

Answer-merging controls. Merged answers use canonical-answer groups, the first member's answer embedding, and the union of their trajectories' Subquery/Evidence memberships. Group labels average member correctness. In the browsing control, BCE uses the resulting soft labels; listwise and hard-ranking losses treat groups with mean correctness greater than 0.5 as positive and the remaining groups as negative. Merging preserves the occurrence graph's question split, process nodes, and process edges. The browsing readout and answer-merging controls use the full model's frozen embeddings and source identities.

Occurrence representation and evidence connections. The process-merging control combines identical-content evidence occurrences and their incident retrieval connections. Answer merging changes the evaluation unit: identical answers share a readout over the union of their trajectories' contexts, as defined above. On long-horizon browsing, merging process occurrences, answers, or both gives 77.1%, 76.3%, and 76.4%, respectively, compared with 78.6% for Full TRACE. The corresponding WebQA results are 44.1%, 44.9%, and 44.4%, compared with 45.2%. These comparisons support preserving each occurrence's process context when ranking trajectories.

The connection controls change which identities link evidence. On WebQA, removing all content-identity links gives 43.8% EM. In long-horizon browsing, removing both Doc and content links gives 71.1% average accuracy, 7.4 points below Full TRACE. Direct content links alone achieve 76.5%, while combining Doc and content links gives 76.9%. Doc-based TRACE reaches 78.6%, supporting source identity as a useful relation between different passages while preserving their query and tool contexts. These broader interventions also remove within-rollout matches or document aggregation, unlike the ablation of cross-rollout graph messages in the main text.

Readout architecture and training. Graph readout replaces cross-attention with process-to-answer GNN messages. Two-stage training first fits this graph selector, then freezes its encoders, GNN, and scoring layers while training a residual attention readout. Their long-horizon averages are 75.8% and 75.1%, respectively, compared with 78.6% when TRACE jointly trains its process encoder and attention readout. The resulting gains of 2.7 and 3.4 points support jointly optimizing process encoding and answer-conditioned readout. The fixed-query control in Table 3 more directly tests answer conditioning by retaining attention and changing its query.

C Selection under Different Rollout Budgets

Table C.1 reports aggregate fixed-checkpoint budget results, complementing the individual dataset–backbone curves in Figure 2. WebQA uses Qwen2.5-14B Base rollouts. Long-horizon results weight the six dataset–backbone combinations equally, with the same 500-question FRAMES subset at every budget. For long-horizon $K = 1$, we report mean Pass@1; $K \geq 2$ uses the first K samples.

Table C.1: Budget-scaling results (%). WebQA: TRACE EM/F1 on Qwen2.5-14B Base rollouts through $K = 32$. Long-horizon: accuracy through $K = 16$.

	WebQA		Long-horizon avg.		
K	EM	F1	TRACE	Majority	Pass@K
1	28.5	36.4	59.7	59.7	59.7
2	35.2	43.9	67.8	65.5	72.2
4	40.2	48.8	71.5	69.7	80.1
8	43.3	51.8	75.3	73.8	86.1
16	45.2	53.4	78.6	75.5	90.9
32	46.3	54.6	—	—	—

D Policy-Sampling Results

Table D.1 reports the Pass@K curves discussed in Appendix G. For each question, let c_q denote the number of correct trajectories among 64 samples. We estimate Pass@K as $1 - \binom{64-c_q}{K} / \binom{64}{K}$ and average over questions, taking the numerator to be zero when $64 - c_q < K$. Thus Pass@1

averages correctness over all 64 samples per question, whereas the WebQA single-rollout references evaluate sample 0. SFT coverage uses all generated samples before selector filtering; normalized empty predictions count as incorrect.

The Qwen2.5-7B and Qwen2.5-14B SFT policies are trained with full-parameter updates for three epochs on 3,083 successful NQ/HotpotQA search trajectories generated by Qwen3.6-35B-A3B. Each policy is evaluated on 64 samples for each of the same 3,125 WebQA questions. In the table, Q2.5 and Q3 abbreviate Qwen2.5 and Qwen3.

Table D.1: Question-weighted Pass@K (%) for the policies in the parallel-search study.

Policy	1	2	4	8	16	32	64
Q2.5-7B Base	24.9	35.3	44.8	52.6	58.6	63.4	67.4
Q2.5-14B	30.5	40.1	48.0	54.3	59.3	63.6	67.3
SearchR1 RL	43.7	46.5	48.8	50.6	52.1	53.3	54.2
Q2.5-32B	30.5	39.7	47.6	54.2	59.4	63.6	67.4
Q3-32B	36.1	43.8	50.0	54.7	58.5	61.7	64.2
SFT (7B)	36.3	43.4	49.4	54.4	58.5	62.1	65.3
SFT (14B)	38.6	44.4	49.3	53.5	57.4	60.9	64.0

E Additional WebQA Rollout Policies

Table E.1 reports all six WebQA rollout policies with Qwen3-32B generative aggregation. The main WebQA table uses Qwen3-32B aggregators; Appendix F gives the comparison by dataset. TRACE improves over majority and weighted voting across all six policies. Table E.1 presents these voting comparisons as transfer across rollout generators.

Table E.1: WebQA EM (%) at $K = 16$ across six rollout policies, weighted by question count over the seven datasets. SolAgg, SummAgg, and AggAgent use Qwen3-32B. Bold marks the best result in each column.

	Qwen2.5-7B rollouts			Qwen2.5-14B rollouts		
Method	Base	SFT	RL	Base	SFT	RL
Single rollout	24.6	36.6	44.0	29.5	38.8	46.9
Majority Voting	38.1	46.0	45.1	42.6	47.2	47.6
Weighted Voting	38.5	46.1	45.3	43.0	47.3	47.7
Fewest Tools	27.2	42.0	45.3	33.2	44.4	47.1
SolAgg	42.4	48.1	**46.5**	43.9	48.0	**48.5**
SummAgg	**42.9**	46.8	44.9	43.7	46.7	46.4
AggAgent	40.7	43.3	44.2	41.5	43.6	46.0
TRACE	41.2	**48.3**	45.8	**45.2**	**49.2**	47.9

The RL pools contain much more repetitive search evidence. We measure overlap using the Jaccard similarity between the retrieved-chunk sets of two retained trajectories, averaging over pairs within each question and then over questions with at least two retained trajectories. Mean overlap reaches 0.87 and 0.83 for 7B and 14B RL, compared with 0.36 and 0.50 for Base and 0.56 and 0.60 for SFT. The difference persists on matched questions where both policies have correct and incorrect candidates. Repetition creates many graph links, but these links connect fewer distinct evidence contexts. This reduced complementarity is consistent with the smaller gains over voting on RL rollouts: 0.7 points at 7B and 0.3 points at 14B.

For 7B Base, candidate availability is a more immediate constraint. Of the first 16 samples per question, 27.2% are excluded because they lack a nonempty answer or retrieved evidence, compared with 21.1% for 14B Base. Zero-search trajectories are especially common: 18.1% at 7B versus 6.0% at 14B. Some already answer correctly, but cannot enter the graph under the evidence requirement. Filtering reduces the fraction of questions with a correct candidate from 59.0% to 56.5% at 7B,

compared with 59.2% to 58.2% at 14B. The resulting 7B pool contains 11.6 candidates per question on average. TRACE still gains 3.0 points over majority voting within this smaller pool, recovering useful distinctions among the remaining trajectories.

F QWEN3-32B AGGREGATION ON WEBQA

Table F.1 evaluates Qwen3-32B aggregation on the same Qwen2.5-14B Base and SFT pools as Table 1. Replacing Qwen2.5-14B with Qwen3-32B improves all three generative methods on these pools. On SFT rollouts, SolAgg rises from 33.9% to 48.0% Overall EM, SummAgg from 17.7% to 46.7%, and AggAgent from 29.3% to 43.6%. TRACE reaches 49.2%, retaining the highest Overall EM in this comparison without an additional autoregressive aggregation stage.

Table F.1: WebQA EM (%) with Qwen3-32B generative aggregation at $K = 16$ with Qwen2.5-14B Base and SFT rollouts. SolAgg, SummAgg, and AggAgent use Qwen3-32B. Overall is question-weighted within each rollout policy. Bold marks the best result in each column.

	NQ		HotpotQA		TriviaQA		PopQA		2Wiki		MuSiQue		Bamboogle		Overall	
Method	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT	Base	SFT
Single rollout	29.4	41.4	27.6	36.8	49.6	64.2	35.2	38.6	24.6	31.4	10.4	18.4	31.2	45.6	29.5	38.8
Majority Voting	44.6	46.6	40.4	46.8	63.8	72.2	46.0	46.2	40.8	46.2	18.4	22.4	48.8	59.2	42.6	47.2
Weighted Voting	44.4	46.6	40.6	47.2	64.8	72.6	46.0	46.2	41.6	45.6	19.0	22.4	48.8	60.0	43.0	47.3
Fewest Tools	36.0	45.0	30.2	42.6	58.2	71.2	39.4	43.4	26.0	41.0	9.6	20.2	32.8	57.6	33.2	44.4
SolAgg	43.2	43.4	42.6	**49.0**	65.0	72.4	45.0	46.4	45.8	**48.6**	19.6	25.0	52.8	**60.8**	43.9	48.0
SummAgg	41.2	41.6	**44.2**	48.0	62.2	71.6	43.6	44.0	**48.0**	47.2	21.2	24.6	51.2	58.4	43.7	46.7
AggAgent	41.0	39.6	37.8	44.0	63.6	68.0	40.4	42.2	44.8	43.2	17.0	21.0	**60.0**	59.2	41.5	43.6
TRACE	**47.4**	**48.4**	43.2	48.2	**65.6**	**73.2**	**47.6**	**49.2**	44.8	47.2	**21.4**	**26.0**	51.2	60.0	**45.2**	**49.2**

G CANDIDATE COVERAGE AND SELECTION

G.1 PARALLEL SEARCH AND EVALUATION

We distinguish the opportunities created by repeated search from the ability to select a successful trajectory (Brown et al., 2024). For a question q and a fixed search policy π, let $\mathcal{T}_K = (\tau_1, \ldots, \tau_K)$ contain K completed rollouts, with final answers a_i and correctness labels $y_i \in \{0, 1\}$. Define $E_K = \{\max_i y_i = 1\}$ as the event that the pool contains a correct answer. Candidate coverage is $C_K(\pi) = \Pr(E_K)$, or Pass@K. A selector S returns an index $\hat{\imath} = S(q, \mathcal{T}_K)$ without observing the correctness labels. Its final-answer accuracy satisfies, for $C_K(\pi) > 0$,

$$A_K(\pi, S) = \Pr(y_{\hat{\imath}} = 1) = C_K(\pi)\, \Pr(y_{\hat{\imath}} = 1 \mid E_K). \tag{G.1}$$

Probabilities range over questions and sampled rollouts. This decomposition separates candidate coverage from selection success conditional on a correct answer being available. On fixed candidate pools, oracle accuracy is the fraction of questions containing a correct candidate. Comparisons with this oracle concern selection from the same pools, rather than methods that may generate new answers.

G.2 POLICY CHOICE UNDER PARALLEL SEARCH

Table D.1 compares Base, full-SFT, and Search-R1 RL (Jin et al., 2025) with the same Qwen2.5-7B backbone on seven WebQA datasets. Results are weighted by question count; Appendix D gives the SFT protocol and full coverage results. The policy ordering changes with the sampling budget. Equation G.1 further shows that coverage must be paired with successful selection, which we learn from completed rollouts while holding their generating policy fixed.

G.3 A Probabilistic View of the Coverage–Voting Gap

For a fixed policy π, assume rollouts are i.i.d. conditional on q over finitely many normalized answer classes. Let $\mathcal{A}_q^+$ denote the correct classes, p_q their total probability, and $a_q^\star$ the unique modal class. Majority voting S_{MV} selects the most frequent class. Independence and the law of large numbers give

$$\begin{aligned} C_K(\pi) &= \mathbb{E}_q\big[1-(1-p_q)^K\big] \xrightarrow{K\to\infty} \Pr_q(p_q>0), \\ A_K(\pi, S_{\mathrm{MV}}) &\xrightarrow{K\to\infty} \Pr_q(a_q^\star \in \mathcal{A}_q^+). \end{aligned} \tag{G.2}$$

The limiting gap is therefore $\Pr_q(p_q > 0,\, a_q^\star \notin \mathcal{A}_q^+)$: sampling eventually finds any reachable correct answer, whereas voting converges to the most probable class, even if incorrect. For probabilities $(0.2, 0.5, 0.3)$ with only the first class correct, coverage tends to one and voting accuracy to zero. This gap requires no dependence between rollouts.

H Efficiency Measurements

We measure post-rollout runtime on the full 3,125-question WebQA evaluation with Qwen2.5-14B SFT rollouts and NVIDIA H20 GPUs. The first 16 samples provide 35,503 retained candidates across 2,939 nonempty pools; all questions remain in the denominator. Table 4 compares H20 GPU-hours for TRACE and the Qwen3-32B generative aggregators. Rollout generation and correctness judging are excluded.

TRACE profiling covers text encoding, graph construction, and answer selection from completed candidate texts. The 68,664 distinct node texts are encoded once each with Qwen3-Embedding-8B, using BF16, batches of 16, a 4,096-token limit, and 4,096-dimensional outputs. The selector scores graphs in batches of 256. GPU timings use CUDA synchronization. Table H.1 includes the frozen encoder, which accounts for most of the inference time.

Table H.1: Measured TRACE runtime on one H20 for the full 3,125-question pool. Text embeddings are recomputed, not loaded from a previous run.

Stage	Seconds
Read and parse candidate texts	23.6
Fresh text encoding	905.2
Assemble tensors and construct graphs	38.2
Score and extract answers	7.2
Total	974.2

Model initialization adds 52.8 seconds to the reported runtime. When graph tensors are reused, batched selector inference takes a median of 1.09 seconds over three measurements, including batching and device transfer.

The Qwen3-32B SummAgg evaluation generates 35,503 trajectory summaries and 2,939 final integrations. With two H20s using tensor parallelism, the complete pipeline takes 59,322 seconds (16.5 hours). This includes summary generation, final integration, auxiliary confidence scoring, and client scheduling, and excludes rollout generation and correctness judging.

GPU-hours in Table 4 sum elapsed time multiplied by the number of H20 GPUs used in each interval. SolAgg takes approximately 98.5 minutes on two H20s (3.28 GPU-hours), reconstructed from the first non-pilot request to the final confidence output; the two pilot questions are reused. SummAgg uses 32.96 GPU-hours. AggAgent uses an estimated 69.93 GPU-hours, integrating its initial two-GPU allocation and later four-GPU allocation through the terminal cutoff. This deployment estimate includes restarts and abnormal waits, with two unfinished questions counted as failures. SolAgg and SummAgg timings include auxiliary confidence scoring; concurrent request durations are not summed as batch runtime.

A three-epoch selector fit on 101,323 training questions, with validation on 5,309 questions, takes 4.17 minutes on one H20 using precomputed embeddings; including data loading and setup, it takes 6.42 minutes.