TRACE-RAG: TRACE-CONSISTENT RETRIEVAL WITH CLAIM-GATED MEMORY

OptimalScale AIScientist

ABSTRACT

Multi-step retrieval-augmented research agents often suffer from evidence drift: an early unsupported intermediate statement is written into persistent memory, then reused to generate follow-up queries and plans, creating a self-reinforcing trajectory that wastes limited tool calls and tokens and can culminate in seemingly well-cited but poorly grounded outputs. This problem is important because users increasingly expect not only fluent synthesis but also auditable evidence trails, and it is challenging because verification is imperfect, intermediate claims are compositional, and exhaustive checking and repair can exceed fixed budgets. We propose TRACE-RAG, a trace-consistent retrieval framework that treats intermediate claims as first-class objects, verifies each proposed claim against retrieved passages while extracting minimal supporting spans, and enforces claim-gated memory so that only verified claims may condition subsequent planning and query generation. To operate under strict budgets, TRACE-RAG adds dependency-aware impact scoring to prioritize selective repair of the most consequential failing claims, and introduces anti-confirmation controls for high-impact claims via mandatory counterevidence attempts and source diversity requirements before admission to verified memory. We will evaluate TRACE-RAG under identical per-question tool-call and token envelopes against single-shot RAG, an iterative agent with free-form memory, and an iterative agent with post-hoc final verification, using multi-hop question answering benchmarks that stress long-horizon reasoning and support evaluation of grounding. Our experimental plan reports end-task answer quality alongside trace-level grounding and citation faithfulness metrics, includes budget sweeps to produce accuracy-versus-cost curves, and performs ablations that isolate the effects of memory gating, selective repair, disconfirming queries, verifier choice, and source diversity thresholds.

1 Introduction

Retrieval-augmented generation (RAG) and tool-using language model agents are increasingly deployed for multi-step research workflows that require planning, iterative querying, and synthesis with citations. Yet over long horizons, these systems must manage a persistent internal state under fixed budgets: early unsupported intermediate statements can be written into scratchpad or memory, then reused as premises for subsequent query generation and planning. TRACE-RAG changes this dynamic by verifying intermediate claims before they are allowed to persist and by gating memory so that only admissible, evidence-supported premises can steer later retrieval. Here, a claim is any intermediate statement the agent intends to reuse as a premise, verified means the statement is supported by retrieved text spans under an explicit check, and the gate permits only verified claims to be written to persistent state while blocking unverified ones from influencing later planning. The concrete benefit is trajectory control: preventing early unsupported premises from steering downstream retrieval and planning when tool calls and tokens are limited.

This problem matters because users increasingly demand not only fluent responses but also auditable evidence trails, especially in settings where decisions depend on the reliability of intermediate reasoning. Analyses of RAG attribution have shown that citations can be unfaithful to what the model relied on, motivating evaluation beyond surface-level support and toward evidence-grounded generation [1, 2]. In multi-step agents, the risk is amplified: once an ungrounded premise enters persistent state, it can steer many downstream actions, making later correction expensive and difficult.

Naive mitigations often fail because they do not control the agents trajectory. Post-hoc checking of the final response can detect some errors, but it provides weak guidance about which earlier premise caused the drift and does not recover

the budget already spent exploring an incorrect branch. Simply doing more retrieval is also not a solution if the agent continues to reuse unsupported intermediate text as if it were evidence. Hallucinations can arise at multiple stages of an agent workflow, including planning, tool use, and memory updates, and can propagate across steps if not explicitly constrained [3]. Without an operational rule governing what may be reused as a premise, the agents own intermediate text can become an unvetted source of facts.

Prior work has made important progress in characterizing these issues and proposing mitigation strategies, but much of it is framed as stage-specific detection or correction rather than as a control mechanism over persistent state [3]. Separately, structured multi-hop reasoning representations can make intermediate steps more interpretable and easier to refine, supporting error localization [4]. Relatedly, multi-hop question answering settings explicitly require chaining intermediate facts across steps, making them a natural stress test for how unsupported intermediate claims can derail downstream reasoning [5, 6]. However, interpretability alone does not ensure that intermediate claims used for future retrieval are actually supported by retrieved evidence, nor does it directly address budgeted settings where verification must be selective and trajectory-shaping.

We propose TRACE-RAG, a trace-consistent retrieval framework that treats intermediate claims as first-class objects and enforces a simple principle: any claim that will be reused for planning or query generation must be explicitly supported by retrieved evidence, otherwise it is blocked and repaired. Verification produces a label and confidence for each claim based on whether the retrieved spans are sufficient to support it; biases or misalignment can arise when the verifier over-trusts plausible text, misses needed context, or is overly conservative under ambiguity. In particular, we emphasize span minimality and overhead as first-order design constraints: the system aims to admit only the minimal supporting spans needed to justify a claim, while keeping verification and repair costs within a fixed budget. The intended failure mode is precision-first admission: when evidence is ambiguous or compositional, TRACE-RAG prefers to block the claim rather than allow a weakly supported premise to persist. This conservative gating can reduce coverage, and TRACE-RAG mitigates the trade-off by prioritizing selective repair of high-impact blocked claims and by adding anti-confirmation checks intended to prevent the agent from repeatedly reintroducing the same unsupported premise when budgets are tight. The proposal targets text-based multi-hop question answering first; extensions to other modalities or languages are future work rather than a current claim. More broadly, retrieval-augmented methods have also been used to extract and externalize algorithmic logic from neural models, reinforcing the value of grounding intermediate structure in retrieved evidence rather than relying on opaque internal state [7]. We will evaluate the approach by measuring evidence faithfulness and downstream task quality under controlled budgets, with an explicit focus on expected trade-offs between robustness and cost.

In summary, TRACE-RAG contributes:

- **Claim-level verification with claim-gated memory:** verify each reusable intermediate statement against retrieved spans before it can enter persistent state, preventing evidence drift and keeping unsupported premises from steering later retrieval and planning.
- **Budget-aware selective repair:** allocate verification and correction to the highest-impact blocked claims, reusing compute that would otherwise be spent exploring an incorrect branch to improve reliability under fixed budgets.
- **Anti-confirmation controls:** block self-reinforcing reintroduction of previously rejected premises, reducing the chance that early errors become entrenched when budgets are tight.

2 Related Work

Hallucinations and Error Propagation in LLM Agents A growing body of work has characterized how LLM-based agents hallucinate across the agent workflow, including during planning, tool use, and memory updates, and has organized mitigation strategies into taxonomies of causes and interventions [3]. This line of research motivates our focus on long-horizon failure modes where early unsupported statements can silently become "facts" that steer subsequent actions. However, much of the surveyed mitigation landscape is framed as detection or correction at particular stages (e.g., prompting, verification, or tool-use safeguards) rather than as an explicit operational constraint that prevents ungrounded intermediate claims from being reused as premises for future retrieval and planning under a fixed budget [3]. TRACE-RAG is positioned as a complementary control-oriented approach: instead of only diagnosing hallucinations, it aims to reduce cascading drift by gating what can enter and influence the agent's persistent state. Related retrieval-augmented systems have also been explored for claim retrieval and nuanced claim analysis, motivating our emphasis on trace-level auditing and evidence packaging [8, 9, 10].

Faithfulness and Attribution in Retrieval-Augmented Generation Recent work has emphasized that even when RAG outputs are correct, their citations may not be faithful indicators of what the model actually relied on, highlighting

post-rationalization and the need to evaluate attribution beyond surface-level support [1, 2]. This distinction is closely related to the "citation laundering" failure mode in multi-step settings: an agent can arrive at a conclusion via unsupported intermediate reasoning and later attach plausible sources that appear to justify it. While attribution-focused analyses expose why correctness-only evaluation is insufficient and propose desiderata for trustworthy citations [1], they do not by themselves prescribe how to control an agent's intermediate state so that future steps cannot be conditioned on unverified premises. TRACE-RAG directly targets this gap by enforcing trace consistency at the level of intermediate claims, aiming to improve both end-task accuracy and the faithfulness of the evidence trail by construction rather than post hoc. For broader context on RAG design patterns and evaluation, we refer to recent surveys [11].

Structured Multi-Hop Reasoning with Entailment Trees and Iterative Feedback For complex multi-hop question answering, structured reasoning representations such as entailment trees provide interpretable intermediate steps and can be refined through iterative feedback mechanisms, improving robustness to irrelevant evidence and making errors easier to localize [4, 12]. These approaches align with our emphasis on explicit intermediate reasoning objects, but they are typically designed around generating a coherent reasoning structure for answering (including multimodal settings) rather than enforcing a retrieval-and-memory discipline where only entailed intermediate claims are allowed to influence subsequent query generation and planning decisions under strict tool/token budgets [4]. TRACE-RAG builds on the general insight that interpretable intermediate structures aid debugging, but differs in treating intermediate claims as first-class control variables whose admissibility into memory is gated by entailment, thereby aiming to prevent drift before it shapes downstream retrieval trajectories. This is complementary to recent work on multi-hop retrieval and logic-aware RAG that explicitly explores multi-hop neighborhoods or aligns planning with the retrieval set [13, 14, 15, 16].

Agentic and Domain-Specific RAG Systems Beyond QA benchmarks, recent work has explored agentic retrieval interfaces and domain-specific RAG systems that emphasize transparency and traceability of retrieval–generation dynamics [15, 2]. Such systems motivate our focus on trace-level logging and controllable memory updates. In addition, retrieval-augmented frameworks have been applied to explainable analysis in specialized domains (e.g., code and security), illustrating how RAG can bridge natural-language queries and structured artifacts while producing human-readable reports [17].

3 Method

Problem definition (budgeted multi-step RAG with trace consistency). Let x denote a user query. An agent interacts with a retrieval tool over T steps to produce a final response y and an auditable trace. At each step $t \in \{1, \dots, T\}$, the agent issues a retrieval query $q_t \in \mathcal{Q}$, receives a set of passages $P_t = \{p_{t,1}, \dots, p_{t,k}\}$ with provenance metadata (e.g., document identifiers and offsets), and updates an internal state. We assume a fixed budget B that constrains total tool calls and language-model tokens, so the agent must decide where computation is spent: which intermediate statements to verify, and which failing statements to repair, under limited resources. The core failure mode we target is *evidence drift*: an intermediate statement is written into persistent state without being supported by retrieved evidence, then reused as a premise for later query generation and planning, causing compounding errors over long horizons [3]. Our goal is to learn a policy that maps (x, B) to (y, τ) where τ is a structured trace, while enforcing *trace consistency*: any intermediate claim that is reused to condition future actions must be explicitly supported by retrieved evidence, improving the faithfulness of the evidence trail rather than relying on post-hoc attribution [1].

Figure 1 illustrates the overall TRACE-RAG pipeline and how selective verification plus claim-gated memory prevent unsupported intermediate statements from steering downstream retrieval and planning.

Claim-centric state representation (why structured objects). TRACE-RAG represents intermediate knowledge as a set of atomic claims with explicit dependencies, inspired by the general utility of structured intermediate reasoning objects for localization and refinement of errors in multi-hop settings [4, 12]. Concretely, we maintain a claim set $\mathcal{C}$ where each claim is an object

$$c = \langle s, \pi, \ell, \gamma, \mathsf{span} \rangle, \tag{1}$$

with claim text s, provenance identifiers π (document ids and offsets for retrieved passages), a verifier label $\ell \in \{\textsc{Entailed}, \textsc{Contradicted}, \textsc{NEI}\}$, a confidence score $\gamma \in [0, 1]$, and supporting span(s) span when $\ell = \textsc{Entailed}$. We also maintain a directed dependency graph $G = (V, E)$ whose nodes include claims and actions (queries/subgoals), and whose edges record when a claim was used as a premise for generating a downstream query or subgoal. This explicit bookkeeping is designed to make drift attributable to specific unsupported premises, addressing the long-horizon propagation concerns highlighted in agent hallucination taxonomies [3].

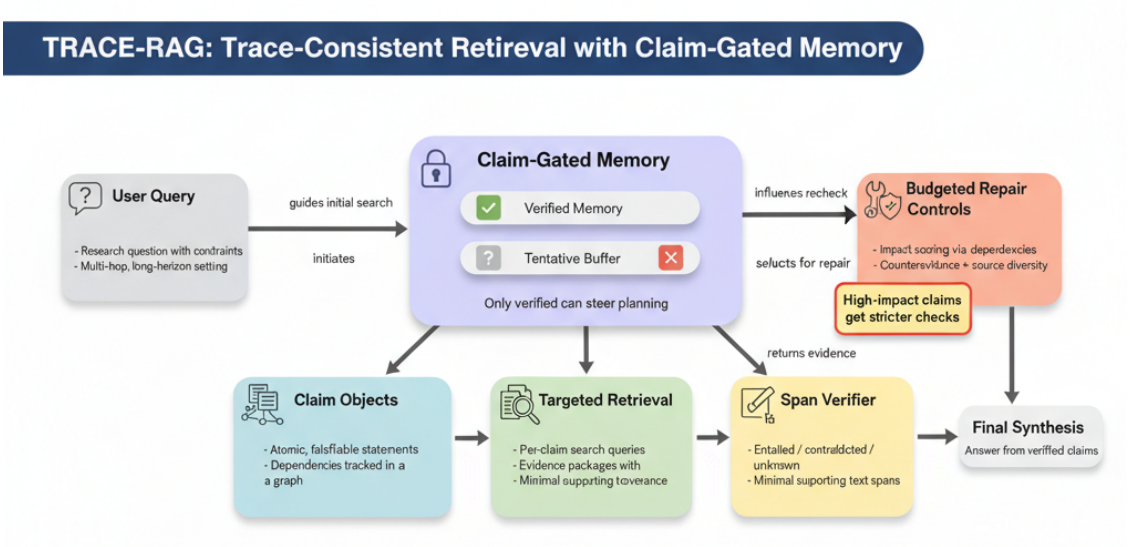

Figure 1: Overview of the proposed method. TRACE-RAG runs a closed-loop retrieve–verify–gate–repair pipeline that proposes atomic claims with per-claim queries, verifies them with entailment labels and span provenance, gates memory so only entailed claims can condition future planning, and prioritizes selective repair using a dependency-aware impact score.

Trace-consistent memory gating (what is allowed to influence future steps, and why). We partition claims into three disjoint memories: verified memory M, tentative buffer T, and refuted memory R. The key operational constraint is that only verified memory may be used as factual premises for planning and query generation:

$$M = \{c \in \mathcal{C} : \ell(c) = \textsc{Entailed}\}. \tag{2}$$

$$T = \{c \in \mathcal{C} : \ell(c) = \text{NEI}\}. \tag{3}$$

$$R = \{c \in \mathcal{C} : \ell(c) = \textsc{Contradicted}\}. \tag{4}$$

Let Plan be a planner that proposes the next action (e.g., a subgoal and retrieval queries) given the user query and current state. TRACE-RAG enforces

$$a_{t+1} \sim \mathsf{Plan}(x, M, R, B_t), \tag{5}$$

where B_t is the remaining budget, and where T is *not* provided as admissible factual context. The rationale is control-oriented: by preventing unsupported intermediate statements from entering the set of premises that can steer future retrieval, we aim to reduce cascading drift rather than merely detecting unfaithful citations after generation [1]. This gating rule operationalizes the intuition that persistent state should be treated as a controlled resource in long-horizon agents, since hallucinations can arise during memory updates and then propagate across steps if unconstrained [3]. When verification is noisy, this gate degrades gracefully: conservative admission (treating borderline cases as NEI) reduces coverage but preserves precision by keeping uncertain claims out of M.

Constrained claim proposal and query generation (atomicity and falsifiability). At each step, the planner proposes a small set of candidate atomic claims $\{s_{t,1}, \ldots, s_{t,K}\}$ and associated retrieval queries $\{q_{t,1}, \ldots, q_{t,K}\}$, following the broader idea that query generation quality can be a bottleneck for retrieval effectiveness in RAG pipelines [18, 19, 20]. We treat claim proposal as a constrained generation problem:

$$(s_{t,i}, q_{t,i}) \sim \mathsf{Propose}(x, M, R, B_t) \quad \text{for } i = 1, \ldots, K, \tag{6}$$

subject to the constraint that each $s_{t,i}$ is (i) falsifiable, (ii) scoped to explicit entities and relations, and (iii) intended to be verifiable from short retrieved passages. This design is motivated by multi-hop QA settings where intermediate facts must be chained; making those facts explicit and checkable improves interpretability and supports iterative refinement [4]. Additionally, by generating queries per claim rather than a single monolithic query, we align retrieval with the granularity at which verification and memory updates occur, which is critical for preventing unsupported premises from being reused.

Targeted retrieval and evidence packaging (what is retrieved, and why). Given each claim-query pair $(s_{t,i}, q_{t,i})$, the retriever returns a ranked list of passages $P_{t,i} = \{p_{t,i,1}, \ldots, p_{t,i,k}\}$ with provenance. TRACE-RAG treats retrieval as producing an *evidence package* for each claim:

$$e_{t,i} = \langle s_{t,i}, P_{t,i}, \pi(P_{t,i}) \rangle, \tag{7}$$

where $\pi(P_{t,i})$ denotes the provenance metadata for the retrieved passages. The motivation is to decouple (a) evidence acquisition from (b) claim acceptance: retrieval provides candidate support, but verification is applied selectively to authorize which claims may enter M and condition future steps. This separation directly targets the attribution gap in RAG, where citations may be present but not faithful to the actual premises used during reasoning [1].

Entailment-based verification with span provenance (formalizing labels, confidence, and span sufficiency). For each evidence package $e_{t,i}$, a verifier produces (i) an entailment label, (ii) a confidence score, and (iii) supporting span provenance when applicable:

$$(\ell_{t,i}, \gamma_{t,i}, \mathsf{span}_{t,i}) = \mathsf{Verify}(s_{t,i}, P_{t,i}). \tag{8}$$

Operationally, Verify consumes the claim text $s_{t,i}$ and the retrieved passages $P_{t,i}$ and returns a distribution over labels plus candidate supporting spans. We define $\gamma_{t,i}$ as the verifier's confidence in its predicted label (e.g., the maximum label probability after calibration), and we map confidence to decisions using fixed thresholds: if the top label is ENTAILED and $\gamma_{t,i} \geq \theta_E$ and a sufficient span package exists, we accept entailment; if the top label is CONTRADICTED and $\gamma_{t,i} \geq \theta_C$, we mark contradiction; otherwise we return NEI.

Span sufficiency is a hard constraint for ENTAILED: the extracted span(s) must explicitly mention the key entities and the asserted relation, and must include any qualifiers needed for the claim's meaning (e.g., scope, time, or comparison class). If the best spans are ambiguous (entity mismatch, underspecified relation, or conflicting phrasing across passages), the verifier returns NEI so the claim remains in T and cannot enter verified memory. Some claims require evidence distributed across multiple passages (or documents). In these cases, Verify may return a small set of spans $\mathsf{span}_{t,i} = \{\sigma_1, \sigma_2\}$ and require that their joint content covers the claim; if no such compact multi-span package exists within the retrieval budget, the claim is treated as NEI.

These outputs feed directly into control: the label determines memory placement ($M/T/R$), and the confidence and span sufficiency determine whether a claim is admissible as a premise for planning. Conservative thresholds (larger θ_E, θ_C) reduce false admissions to M at the cost of leaving more claims in T.

Dependency-aware impact scoring and selective repair (making weights and objective explicit). Verification introduces the possibility that some proposed claims are NEI or CONTRADICTED. Under a fixed budget, repairing every failing claim is infeasible, so TRACE-RAG prioritizes repairs using the dependency graph. Let $\mathsf{dep}(c)$ denote the set of downstream actions that depend on claim c (via edges in G). We define an impact score

$$I(c) = \sum_{a \in \mathsf{dep}(c)} w(a), \tag{9}$$

where $w(a)$ is a nonnegative action weight computed from the trace structure, for example as a product of (i) a depth-based discount (actions closer to final synthesis receive larger weight) and (ii) a fan-out term (actions that spawn more subsequent queries/subgoals receive larger weight). Intuitively, $I(c)$ estimates how much downstream computation and final-answer content would be affected if c is wrong; we expect it to correlate with downstream cost and accuracy sensitivity, without requiring a proof.

The repair policy selects a subset of failing claims

$$\mathcal{R}_t = \arg\max_{\mathcal{S} \subseteq (T \cup R)} \sum_{c \in \mathcal{S}} I(c) \quad \text{s.t.} \quad \mathsf{Cost}(\mathcal{S}) \leq B_t^{\text{repair}}, \tag{10}$$

where B_t^{repair} is the reserved repair budget and $\mathsf{Cost}(\cdot)$ upper-bounds additional retrieval and verification calls induced by attempting to fix each claim (including any counterevidence attempt). This makes the control logic explicit: verify to label claims, gate to prevent uncertain claims from steering planning, score failures by downstream impact, then spend the reserved repair budget on the most consequential ones.

Budget and complexity accounting (bounding added calls). The additional overhead relative to a baseline RAG agent comes from (i) verification calls, (ii) optional counterevidence attempts for high-impact claims, and (iii) repair loops that trigger extra retrieval and re-verification. With at most K proposed claims per step, verification adds at most $T \cdot K$ verifier calls. Retrieval is already required for evidence packages; counterevidence adds at most one extra retrieval per high-impact claim, and selective repair adds at most B^{repair} repair attempts overall, where each attempt is

bounded by a fixed retrieval depth and a single re-verification. The knobs that keep this bounded are K, retrieval depth k, the fraction of claims treated as high-impact (e.g., top-p by $I(c)$), and reserving B^{repair} so repair cannot consume the full budget. Conservative gating further limits repeated work by excluding uncertain claims from M rather than repeatedly rechecking them.

Anti-confirmation constraints for high-impact claims (reducing self-reinforcing drift). To mitigate self-reinforcing retrieval loops, TRACE-RAG applies stricter acceptance criteria to high-impact claims (as measured by $I(c)$). For such claims, we require evidence diversity and an explicit attempt to retrieve counterevidence before admission to M. Formally, let $\mathsf{doc}(p)$ map a passage to its source document. For a high-impact claim c, we require

$$\ell(c) = \textsc{Entailed}, \tag{11}$$

$$|\{\mathsf{doc}(p) : p \in P(c) \ \wedge \ p \text{ supports } c\}| \geq 2, \tag{12}$$

where $P(c)$ denotes the retrieved passages considered for c. In practice, these checks target anticipated failure modes without expanding the budget: verifier false entailment from topical similarity (mitigated by span sufficiency constraints and conservative NEI), false NEI when evidence is compositional across passages (handled by allowing compact multi-span support for high-impact claims), and retriever misses that return only superficially related passages (mitigated by diversity requirements and counterevidence attempts). The rationale is that ungrounded premises can bias subsequent retrieval toward superficially supportive content, amplifying drift across steps [3]; enforcing diversity and counterevidence attempts is a lightweight control mechanism that reduces self-confirmation.

Trace-constrained synthesis (final generation from verified premises). The final response y is generated from verified memory only:

$$y \sim \mathsf{Synthesize}(x, M), \tag{13}$$

with the additional constraint that each factual statement in y must be traceable to one or more claims in M and their supporting spans. This directly targets the concern that citations can be unfaithful or post-rationalized in RAG outputs [1] by making the evidence trail a first-class input to generation. Tentative claims in T may be mentioned only as explicitly uncertain hypotheses and are excluded from core conclusions, ensuring that unsupported intermediate content cannot silently become asserted fact.

Algorithm summary. Overall, TRACE-RAG implements a closed-loop retrieve–verify–gate–repair cycle: it proposes atomic claims and per-claim queries, retrieves evidence packages, verifies selected claims with span provenance, admits only entailed claims into a gated memory that is allowed to condition future planning, and uses dependency-aware selective repair to allocate limited budget to the most consequential failures. By treating intermediate claims as explicit control variables and enforcing trace consistency, the method is designed to reduce long-horizon error propagation in agents [3] while improving the faithfulness of evidence attribution in retrieval-augmented generation [1].

4 Experimental Setup

4.1 Overview, Experimental Factors, and Budget Protocol

We evaluate TRACE-RAG as a budgeted multi-step retrieval agent whose key intervention is *trace-consistent memory gating*: only intermediate claims verified as entailed (with span provenance) may be reused as premises for subsequent planning and query generation (Section 3). The setup tests whether this control reduces long-horizon error propagation in agent workflows [3] and improves evidence attribution faithfulness in retrieval-augmented generation [1] under fixed tool-call and token budgets.

Across all experiments, we vary (i) the agent policy (TRACE-RAG vs baselines), (ii) the budget envelope, and (iii) key TRACE-RAG hyperparameters controlling claim granularity, retrieval depth, and repair selectivity. We keep the underlying language model used for planning and synthesis fixed across methods to isolate the effect of trace consistency and repair policies.

Fixed per-question budget envelope. All methods are evaluated under the same per-question budget envelope, which constrains both tool calls and language-model tokens:

- Total retrieval calls: 12
- Total verification calls: 12
- Total LLM tokens (input + output across planning, verification prompts if applicable, and synthesis): 12,000

- Maximum steps: $T_{\max} = 6$

To make overhead comparable, we report budgets both as totals and as an implied per-step allowance (e.g., 12 retrieval and 12 verification calls over $T_{\max} = 6$ corresponds to up to 2 retrieval and 2 verification calls per step on average, with unused calls carried forward). TRACE-RAG must fit within this envelope by limiting K, k, reranking, and repair retries.

Budget accounting. We log and enforce the following counters per example: retrieval calls, reranker calls (if used), verification calls, and LLM tokens. For baselines that do not use intermediate verification, we still reserve the verification-call budget (i.e., it is not automatically converted into extra retrieval) unless explicitly enabled in a separate cost-curve experiment; this keeps comparisons focused on trace-consistent gating rather than simply spending more retrieval budget.

Cost curves. In addition to the default envelope above, we sweep smaller and larger budgets to produce accuracy-versus-cost curves. We report both quality and latency on these curves to reveal regimes where verification and repair overhead outweighs gains, and to test whether TRACE-RAG improves grounding and end-task performance *under equal budgets* rather than by spending more resources.

4.2 Tasks and Datasets

Primary benchmarks (multi-hop QA). We use multi-hop question answering benchmarks because they require chaining intermediate facts across steps, making them a natural stress test for whether unsupported intermediate premises can derail downstream retrieval and reasoning [5, 13, 14]. Concretely, we run experiments on:

- **HotpotQA (fullwiki)** for multi-hop QA with annotated supporting facts, enabling evaluation of whether retrieved and cited spans align with gold supporting sentences.
- **2WikiMultiHopQA** to stress longer reasoning chains and cross-article evidence aggregation.
- **MuSiQue** to test compositional multi-hop questions that encourage explicit intermediate premises.

Corpus construction. For Wikipedia-based benchmarks, we build a retrieval corpus from the benchmark-aligned Wikipedia snapshot used by each dataset (to avoid leakage across versions). Documents are segmented into overlapping chunks of 200 tokens with 50-token overlap, and each chunk stores metadata: page title, section heading (if available), and stable document identifiers. We additionally sentence-split each chunk and retain sentence offsets to support span-level provenance extraction during verification and citation.

Optional domain shift (biomedical QA). If licensing and retrieval access permit, we include a biomedical QA subset (e.g., abstract-only retrieval) as a contingent extension to test whether trace-consistent gating remains effective when evidence is denser and terminology is specialized. If full-text retrieval is unavailable, we restrict evidence to abstracts and evaluate entailment and citation precision within the retrieved abstract spans.

Scope note. The initial evaluation focuses on English text-only QA; non-English and multimodal generalization are out-of-scope for this experimental plan.

4.3 Systems Compared, Retrieval, Verification, and Memory Gating Details

We compare four systems under identical budget envelopes, including representative recent RAG variants that emphasize agentic retrieval interfaces and multi-hop retrieval reasoning:

1. **Single-shot RAG**: one retrieval round for the user question followed by answer generation with citations.
2. **Iterative RAG agent (free-form memory)**: multi-step retrieve–write loop where subsequent queries may condition on any prior scratchpad/memory text, including unverified intermediate statements.
3. **Iterative RAG + post-hoc final verification**: the same iterative agent, augmented with a final answer verification pass that may trigger one regeneration of the final response.
4. **TRACE-RAG (ours)**: claim decomposition, entailment verification with span provenance, claim-gated memory, dependency-aware selective repair, and anti-confirmation constraints for high-impact claims (Section 3).

This comparison tests whether controlling *intermediate* state (TRACE-RAG) yields benefits beyond post-hoc attribution checks, which are known to be insufficient when citations can be unfaithful to the models actual reasoning process [1].

Retriever. We implement a hybrid retriever combining dense retrieval with BM25. Dense embeddings are computed for all chunks using a fixed sentence-embedding model (e.g., E5-base-v2), and BM25 is built over the same chunk text. For each query, we retrieve k candidates from each retriever and merge by weighted score:

$$\mathsf{score}(p \mid q) = (1-\alpha)\,\mathsf{score}_{\text{dense}}(p \mid q) + \alpha\,\mathsf{score}_{\text{bm25}}(p \mid q), \tag{14}$$

with $\alpha \in \{0.0, 0.3, 0.5\}$ swept in ablations.

Reranker. We optionally apply a cross-encoder reranker to the merged candidate set and keep the top k_r passages per claim query. Unless otherwise stated, we use $k = 5$ and $k_r = 3$ for TRACE-RAG and for iterative baselines that support per-step retrieval. For single-shot RAG, we retrieve k_{single} passages once (set so that total retrieval calls match the budget protocol above).

Deduplication and source diversity. To support TRACE-RAGs diversity constraints for high-impact claims, we deduplicate near-identical chunks using a similarity threshold over dense embeddings and drop duplicates from the candidate pool. Each passage retains a document identifier $\mathsf{doc}(p)$ so that we can enforce minimum distinct-document support when accepting high-impact claims into verified memory.

Claim proposal. At each step t, TRACE-RAGs planner proposes K atomic claims and associated per-claim retrieval queries. We set $K = 3$ by default and sweep $K \in \{2, 3, 4\}$. Claims are constrained to be falsifiable and scoped to explicit entities and relations to make entailment checking feasible from short passages, aligning with the motivation for structured intermediate reasoning objects in multi-hop settings [4]. Under the fixed envelope, this implies that TRACE-RAG can afford only a small number of claim-level retrieval and verification actions per step, so ablations explicitly test whether performance is stable when K, retrieval depth, and repair selectivity are varied.

Verifier and span provenance. For each claim s and its retrieved passages P, the verifier outputs a label $\ell \in \{\text{ENTAILED}, \text{CONTRADICTED}, \text{NEI}\}$, a confidence $\gamma \in [0, 1]$, and supporting span(s) when entailed. We implement two verifier configurations:

- **NLI verifier**: an off-the-shelf NLI model that scores entailment/contradiction/neutral for (s, p) pairs, followed by sentence-level span selection (choose the highest-scoring sentence(s) within p) and minimal span trimming.
- **LLM verifier (fallback)**: a constrained prompt that forces a discrete label and requires extraction of the minimal supporting text span(s) when labeling ENTAILED.

A claim is admitted to verified memory only if $\ell = \text{ENTAILED}$ and the extracted span explicitly mentions the key entities and the asserted relation; otherwise it is downgraded to NEI. This rule targets false support driven by topical similarity, a known contributor to unfaithful attribution in RAG [1].

Thresholds and calibration. We use an entailment acceptance threshold τ_e and contradiction threshold τ_c on verifier confidence, with defaults $\tau_e = \tau_c = 0.7$ and sweeps over $\{0.6, 0.7, 0.8\}$. These sweeps are designed to test robustness to implementation variation (verifier choice and decision thresholds) and to interpret automated grounding metrics in light of verifier imperfections by (i) checking calibration on a held-out validation set (e.g., temperature scaling when training/adapting the verifier) and (ii) running targeted error analysis on a small stratified sample of verifier decisions (false entailments from topical similarity vs false NEI when multi-source aggregation is needed).

High-impact claim constraints. For claims whose dependency-aware impact score $I(c)$ falls in the top p fraction (default $p = 0.3$), we require evidence diversity before admission:

$$|\{\mathsf{doc}(p) : p \in P(c) \ \wedge \ p \text{ supports } c\}| \geq d_{\min}, \tag{15}$$

with $d_{\min} = 2$ by default and an ablation with $d_{\min} = 1$. We also require at least one disconfirming query attempt for high-impact claims to reduce self-reinforcing retrieval loops, consistent with the motivation to control error propagation in agent workflows [3].

4.4 Evaluation Metrics

End-task QA metrics. For each benchmark, we report standard answer quality metrics (Exact Match and token-level F1) on the datasets evaluation split.

Trace grounding and citation faithfulness. To directly test trace consistency and attribution faithfulness, we compute:

- **Premise Support Rate**: the fraction of premises used for planning, query generation, or synthesis that are labeled ENTAILED by the verifier. This metric operationalizes whether the agents intermediate state is grounded before reuse, addressing long-horizon hallucination propagation concerns [3].
- **Citation Precision**: for each cited span–claim pair in the final answer, we re-run the verifier on $(\text{claim}, \text{cited span})$ and count it correct only if it is ENTAILED and contains explicit entity+relation match, targeting the attribution faithfulness issues highlighted for RAG [1].
- **Citation Recall**: the fraction of entailed claims expressed in the final answer that have at least one correct supporting span citation.
- **Memory Contradiction Rate**: the rate at which verified memory contains mutually contradictory claims, detected by running the verifier on selected claim pairs (e.g., same subject-relation with conflicting objects) or by tracking contradiction labels over time.
- **Support Sentence Recall** (when gold supporting facts are available): recall of gold supporting sentences among retrieved passages and among cited spans.

Efficiency metrics. We report total tool calls, total LLM token usage, and wall-clock latency per question. These metrics are necessary to interpret whether improvements come from better control of the trajectory rather than increased computation.

4.5 Experimental Plan

We will conduct the following experiments, each run on the same dataset splits and under the same budget protocol:

1. **Main comparison under fixed budgets**: evaluate TRACE-RAG against the three baselines on HotpotQA, 2WikiMultiHopQA, and MuSiQue using the default budget envelope, reporting end-task metrics, trace grounding metrics, and efficiency.
2. **Budget sweep**: repeat the main comparison across multiple budget envelopes to produce accuracy-versus-cost curves and identify regimes where trace-consistent gating is most beneficial.
3. **Ablations of TRACE-RAG controls**:
   - *No claim gating*: allow NEI claims into memory and permit the planner to use them as premises, testing whether gating is the key mechanism for reducing drift.
   - *No selective repair*: disable repair and proceed using only currently verified claims, testing the trade-off between precision and coverage under fixed budgets.
   - *No disconfirming queries*: remove the mandatory disconfirming query attempt for high-impact claims, testing sensitivity to confirmation bias and self-reinforcing retrieval loops [3].
   - *Verifier swap*: compare NLI-only, LLM-only, and hybrid verification to quantify robustness and cost trade-offs.
   - *Source diversity threshold*: sweep $d_{\min} \in \{1, 2\}$ for high-impact claims to measure the effect on citation precision and recall.
4. **Metric alignment check**: under equal budgets, test whether improvements in grounding metrics (premise support and citation precision) track improvements in end-task QA metrics, to detect metric overfitting.
5. **Failure analysis**: categorize errors into verifier false positives (topic similarity), verifier false negatives (multi-source aggregation needed), retriever misses, claim decomposition errors, and repair loops exceeding budget, using the logged traces and dependency graphs for attribution [4].

Implementation and reproducibility. We run each configuration with three random seeds (1, 2, 3) and keep decoding settings fixed (temperature 0.2, top-p 0.95, and a maximum of 800 output tokens for each planning or synthesis call). For each example, we log: all planner prompts, proposed claims, per-claim queries, retrieved passage identifiers and text offsets, verifier outputs (labels, confidences, extracted spans), memory updates (which claims entered M, T, R), and the dependency graph edges indicating which claims were used as premises for which downstream actions. Training is optional and limited to components whose calibration materially affects gating decisions.

Statistical testing. We use paired bootstrap resampling over questions to compute 95% confidence intervals for key metrics (answer F1, premise support rate, citation precision), and we report mean, median, and effect sizes for TRACE-RAG versus each baseline.

References

[1] Jonas Wallat, Maria Heuss, Maarten de Rijke, and Avishek Anand. Correctness is not faithfulness in retrieval augmented generation attributions. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR)*, ICTIR '25, page 22–32. ACM, July 2025.

[2] Sizhe Cheng, Jiaping Li, Huanchen Wang, and Yuxin Ma. Ragtrace: Understanding and refining retrieval-generation dynamics in retrieval-augmented generation, 2025.

[3] Xixun Lin, Yucheng Ning, Jingwen Zhang, Yan Dong, Yilong Liu, Yongxuan Wu, Xiaohua Qi, Nan Sun, Yanmin Shang, Kun Wang, Pengfei Cao, Qingyue Wang, Lixin Zou, Xu Chen, Chuan Zhou, Jia Wu, Peng Zhang, Qingsong Wen, Shirui Pan, Bin Wang, Yanan Cao, Kai Chen, Songlin Hu, and Li Guo. Llm-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions, 2025.

[4] Qing Zhang, Haocheng Lv, Jie Liu, Zhiyun Chen, Jianyong Duan, Hao Wang, Li He, and Mingying Xv. An entailment tree generation approach for multimodal multi-hop question answering with mixture-of-experts and iterative feedback mechanism. *arXiv preprint arXiv:2412.05821*, 2024.

[5] Vaibhav Mavi, Anubhav Jangra, and Adam Jatowt. Multi-hop question answering, 2022.

[6] Manil Shrestha and Edward Kim. Efficient multi-hop question answering over knowledge graphs via llm planning and embedding-guided search, 2025.

[7] Waleed Khalid, Dmitry Ignatov, and Radu Timofte. A retrieval-augmented generation approach to extracting algorithmic logic from neural networks, 2025.

[8] Ivan Vykopal, Martin Hyben, Robert Moro, Michal Gregor, and Jakub Simko. A generative-ai-driven claim retrieval system capable of detecting and retrieving claims from social media platforms in multiple languages, 2025.

[9] Priyanka Kargupta, Runchu Tian, and Jiawei Han. Beyond true or false: Retrieval-augmented hierarchical analysis of nuanced claims, 2025.

[10] Mohammad Mahdi Abootorabi, Alireza Ghahramani Kure, Mohammadali Mohammadkhani, Sina Elahimanesh, and Mohammad Ali Ali Panah. Multimind at semeval-2025 task 7: Crosslingual fact-checked claim retrieval via multi-source alignment, 2025.

[11] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2023.

[12] Jinyuan Fang, Zaiqiao Meng, and Craig Macdonald. Trace the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation, 2024.

[13] Hao Liu, Zhengren Wang, Xi Chen, Zhiyu Li, Feiyu Xiong, Qinhan Yu, and Wentao Zhang. Hoprag: Multi-hop reasoning for logic-aware retrieval-augmented generation, 2025.

[14] Yuxin Wang, Shicheng Fang, Bo Wang, Qi Luo, Xuanjing Huang, Yining Zheng, and Xipeng Qiu. Multi-hop reasoning via early knowledge alignment, 2025.

[15] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Shaohan Wang, Pengyu Wang, Xiaorui Wang, and Zhendong Mao. A-rag: Scaling agentic retrieval-augmented generation via hierarchical retrieval interfaces, 2026.

[16] Mengqi Zhang, Bowen Fang, Qiang Liu, Pengjie Ren, Shu Wu, Zhumin Chen, and Liang Wang. Enhancing multi-hop reasoning through knowledge erasure in large language model editing, 2024.

[17] Guangyu Zhang, Xixuan Wang, Shiyu Sun, Peiyan Xiao, Kun Sun, and Yanhai Xiong. Tracerag: A llm-based framework for explainable android malware detection and behavior analysis, 2025.

[18] Hamin Koo, Minseon Kim, and Sung Ju Hwang. Optimizing query generation for enhanced document retrieval in rag, 2024.

[19] Weihua Sun, Run-An Wang, and Zhaonian Zou. Query generation based on generative adversarial networks, 2023.

[20] Kaustubh D. Dhole, Shivam Bajaj, Ramraj Chandradevan, and Eugene Agichtein. Queryexplorer: An interactive query generation assistant for search and exploration, 2024.