STAGERAG: ANSWER-SUPERVISED CASCADE OPTIMIZATION FOR RETRIEVAL-AUGMENTED GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented generation (RAG) is commonly deployed as a cascade: a retriever recalls candidates, a reranker selects a compact context, and a language model generates the answer. Optimizing this cascade from answer feedback alone faces two challenges. Corpus-specific supporting chains are costly and ambiguous to annotate, creating a *Gold-Evidence Bottleneck*; meanwhile, final-answer feedback does not reveal where evidence was lost, creating *Cascade Credit Ambiguity*. We introduce STAGERAG, an answer-supervised framework that improves both the retriever and reranker without gold supporting-document annotations. STAGERAG records candidate progression in a logged RAG execution, uses an answer-and-citation verifier to determine answer correctness and support attribution, converts the resulting states into query-level preferences, and balances them with per-query caps. We also formalize a pipeline-aware ranking objective, derive a pointwise rule under idealized conditions, and bound its regret under bounded evidence interactions. On HotpotQA, 2WikiMultiHopQA, and TAT-QA, STAGERAG improves F1 over the strongest compared answer-supervised baseline by 5.65, 12.44, and 6.43 points, respectively. Cascade diagnostics show improved alignment with downstream trajectory preferences and recovery of supporting evidence into the final context.

1 INTRODUCTION

Retrieval-augmented generation (RAG) grounds language-model outputs in external evidence (Lewis et al., 2020). Deployed RAG systems are commonly cascades: a lightweight retriever searches a large corpus, a stronger reranker selects a compact context, and a language model generates the answer. This design improves efficiency but makes evidence selection irreversible. A document missed by retrieval cannot be recovered downstream, and a recalled document excluded from the final context cannot inform generation. Final answer quality therefore depends on how evidence progresses through the complete cascade rather than on any component in isolation. However, final-answer-supervised optimization for RAG is still underexplored, and existing works generally suffer from the following two limitations.

Firstly, practical RAG development faces a *Gold-Evidence Bottleneck*. Reference answers are often available from question–answer collections or task records, but they do not reveal which corpus documents form a complete supporting chain. As illustrated in Figure 1(a), the evidence leading to the same available answer may form a long multi-hop chain, appear redundantly across overlapping passages, contain missing links, or involve conflicting documents. Constructing gold evidence therefore requires annotators to search the deployed corpus and adjudicate among multiple, sometimes incomplete or inconsistent, document combinations. Such annotations are costly, potentially non-unique, and corpus-dependent, and may need revision after the corpus is updated. Although prior work derives retrieval signals from language-model feedback (Shi et al., 2024; Yu et al., 2023; Lin et al., 2024) or improves reranking and relevance modules (Kim & Lee, 2024; Wu et al., 2026), it typically provides only a single answer-level target for one component.

Secondly, final-answer feedback creates *Cascade Credit Ambiguity*. Figure 1(b) visualizes this endpoint ambiguity: an incorrect answer does not reveal whether useful evidence failed to enter

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.

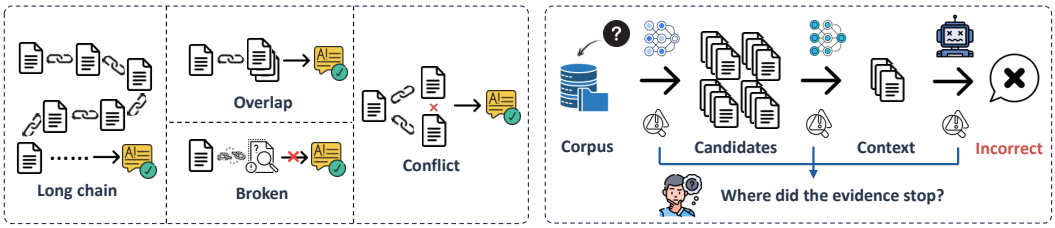


Figure 1: Two challenges in answer-supervised RAG cascade optimization. (a) A reference answer may be available even when its corpus-specific evidence chain is long, overlapping, broken, or conflicting. (b) An incorrect answer exposes only the cascade endpoint, without revealing whether evidence was lost during retrieval, reranking, or generation.

the candidate pool, was removed before reaching the final context, or reached the generator but still failed to produce a correct answer. Answer-level feedback therefore observes only the cascade endpoint, not where each document stopped. An observed trajectory provides finer distinctions: a document may be missed by retrieval, filtered by reranking, retained without verified support, or verified as supporting an answer that is nevertheless incorrect. Treating all such documents as equally negative discards this progression information and can penalize valid evidence when the remaining failure lies in composition or generation. Here, "credit" refers to preserving observable document progression for learning, not causal root-cause attribution. Cascaded ranking work has shown that upstream items should account for downstream selection (Qin et al., 2022; Zheng et al., 2024); this principle remains underexplored when answer feedback is transferred across a RAG cascade.

To this end, we introduce STAGERAG, an answer-supervised cascade optimization framework that improves both the retriever and reranker **without** gold supporting-document labels. Specifically, given questions and reference answers, STAGERAG runs the base cascade once and records whether each candidate is retrieved, retained, and verified by an answer-and-citation verifier as supporting the generated answer. STAGERAG converts the resulting trajectory states into within-query preferences, applies per-query capped balancing to avoid domination by frequent state types, and separately optimizes the retriever and reranker. The trained components replace their original counterparts without changing the inference-time topology. We give both theoretical analysis and empirical study to validate the capability of STAGERAG in jointly optimizing the retrieval system of RAG without any document relevance labels.

Our main contributions are as follows:

- We formulate *answer-supervised cascade optimization* for jointly improving the retriever and reranker of a fixed RAG pipeline without gold supporting-document annotations, and propose **STAGERAG** to learn both stages from verifier-observed trajectories.
- STAGERAG is a RAG-pipeline-agnostic framework, and is compatible with different modern RAG pipelines without changing their inference-time topology.
- We provide theoretical analysis on STAGERAG by co-relating intermediate decisions to expected final performance. We derive an optimal pointwise ranking rule under idealized conditions, and thus bound its regret under uniformly bounded evidence interactions.
- STAGERAG achieves SOTA performance against existing baselines on three public benchmarks, showing its superior performance in optimizing the retrieval system of RAG without any document relevance labels

2 RELATED WORK

Answer-supervised retrieval optimization. Several methods derive retrieval supervision from language-model feedback. W-RAG scores candidates by reference-answer likelihood (Nian et al., 2025); REPLUG distills a frozen language model's document preferences into a retriever (Shi et al., 2024); and AAR transfers language-model preferences to retrievers (Yu et al., 2023). RA-DIT

jointly adapts retrieval and generation components (Lin et al., 2024). These methods reduce reliance on annotated evidence, but generally compress each candidate to one utility or optimize particular components without preserving its progress through a fixed retriever–reranker–generator cascade.

Generation-aware reranking and retrieval augmentation. RE-RAG introduces a relevance estimator for downstream generation (Kim & Lee, 2024), RRPO learns from generation-quality rewards (Wu et al., 2026), and SARK studies passage- and list-level reranking (Zhang et al., 2026). Query2Doc, HyDE, and RAG-Fusion instead modify inference-time retrieval through generated content or query reformulations (Wang et al., 2023; Gao et al., 2023; Rackauckas, 2024). These approaches provide component-level objectives or alternative retrieval paths. We retain a fixed topology and supervise both retrieval stages from each candidate's observed stopping point. Query augmentation and cascade optimization can also be combined: the former expands retrieval through reformulated queries, while the latter improves document selection within the cascade. We examine this combination by applying STAGERAG alongside HyDE and RAG-Fusion.

Cascade-aware learning. RankFlow transfers supervision across stage-specific candidate distributions (Qin et al., 2022), while FS-LTR models downstream selection effects from examples collected throughout a recommendation cascade (Zheng et al., 2024). RAG differs because evidence may be complementary and a generator may fail despite receiving valid evidence. STAGERAG combines document progression with verifier judgments of answer correctness and document support to construct preferences for training both the retriever and reranker. Re^{2}G (Glass et al., 2022) combines retrieval, reranking, and generation through end-to-end training and online distillation, with gold provenance supervision used in its initial retriever and reranker training stages. In contrast, STAGERAG constructs offline document preferences from logged cascade trajectories, optimizing the retriever and reranker without task-specific supporting-document annotations while keeping the generator fixed.

3 METHOD

3.1 OVERVIEW

STAGERAG optimizes both retrieval stages using question–answer pairs, **without gold supporting-document annotations**. As shown in Figure 2, we run the base cascade and record each candidate's progression. An answer-and-citation verifier uses the reference answer to assess answer correctness and attributes support to cited documents within the selected context. These judgments and the observed trajectory define document states; gold supporting-document identities are not supplied to the verifier or used to construct training preferences.

We order the states into within-query preferences and cap each pair type per query to limit frequent transitions. The same preferences separately optimize the retriever for candidate availability and the reranker for final-context selection. The answer model and inference topology remain unchanged.

3.2 PROBLEM FORMULATION

Let q be a question, a^* its reference answer, and $\mathcal{D}$ a document corpus. A retriever R_θ assigns a score $s_{\mathrm{ret}}(q, d)$ and returns a candidate set $\mathcal{C}_q$ containing the top K_r documents. A reranker C_ϕ assigns a score $s_{\mathrm{rank}}(q, d)$ to each retrieved candidate and selects a compact context $\mathcal{E}_q \subset \mathcal{C}_q$ of size K_c. The answer model G then produces

$$\hat{a} = G(q, \mathcal{E}_q). \tag{1}$$

Our default pipeline uses $K_r = 20$ and $K_c = 3$.

The verifier returns $y_q \in \{0, 1\}$ for answer correctness and, for each $d \in \mathcal{E}_q$, $h_q(d) \in \{0, 1\}$ for support of the generated answer.

These decisions construct trajectory states; EM and F1 are reserved for model selection and final evaluation.

Our training data contain (q, a^*) pairs but no gold supporting-document annotations. The objective is to improve the complete cascade by training both R_θ and C_ϕ from its observed behavior, rather than fitting either component to an isolated gold-document distribution.

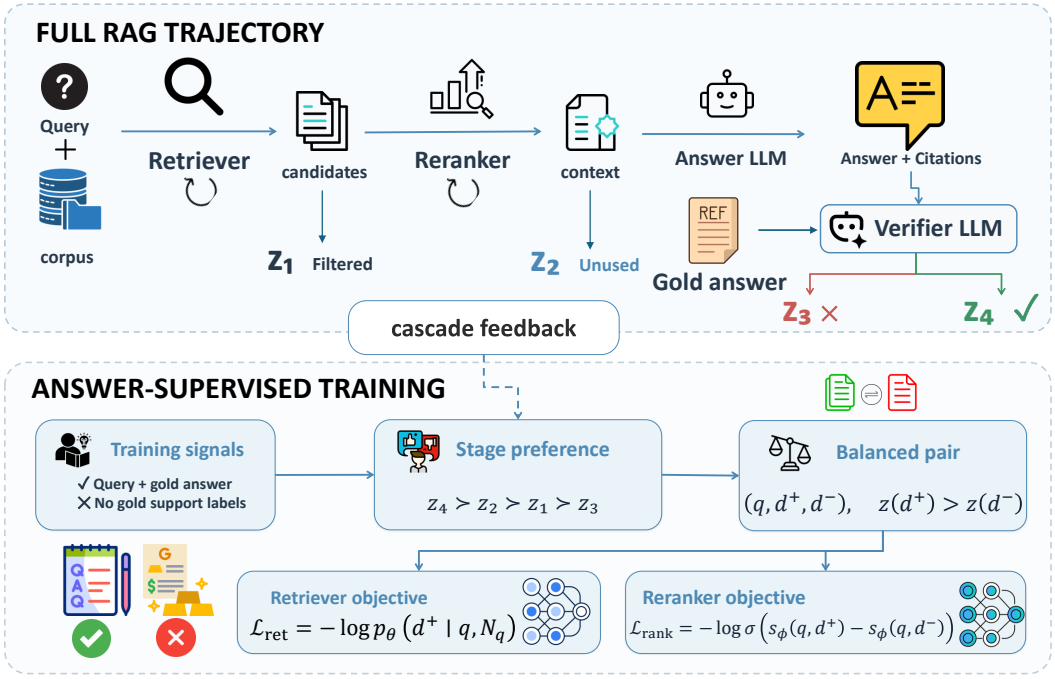


Figure 2: Overview of StageRAG. Questions and reference answers supervise an ordinary RAG execution without gold supporting-document annotations. Observable cascade trajectories induce document states, which are converted into per-query balanced preferences for training the retriever and reranker. The inference topology remains unchanged.

3.3 Cascade-Trajectory State Labeling

For each document, define a retrieval indicator $r_q(d) = \mathbb{I}[d \in \mathcal{C}_q]$ and a retention indicator $k_q(d) = \mathbb{I}[d \in \mathcal{E}_q]$. Support attribution implies retention, so $h_q(d) = 1$ only if $k_q(d) = 1$. We run the base cascade once and assign each observed candidate the state

$$\begin{aligned} z_1 &: r_q(d) = 1,\ k_q(d) = 0, \\ z_2 &: k_q(d) = 1,\ h_q(d) = 0, \\ z_3 &: h_q(d) = 1,\ y_q = 0, \\ z_4 &: h_q(d) = 1,\ y_q = 1. \end{aligned} \tag{2}$$

The four states respectively represent candidates that are filtered after retrieval, retained without support attribution, attributed in a failed trajectory, and attributed in a successful trajectory. Documents outside $\mathcal{C}_q$ may optionally be assigned z_0 during data construction; the primary experiments construct pairs among observed candidates.

These are *cascade-trajectory states*, not semantic relevance grades. They retain more information than the final answer alone by recording where a candidate stops in the cascade. We distinguish an unambiguous core progression relation

$$z_4 \succ_{\text{core}} z_2 \succ_{\text{core}} z_1, \qquad z_3 \text{ incomparable}, \tag{3}$$

from the total ordering used by our default implementation:

$$z_4 \succ z_2 \succ z_1 \succ z_3. \tag{4}$$

The practical placement of z_3 is a risk-averse empirical extension: attribution in a failed trajectory may reinforce an error or reflect valid evidence when another hop is missing or generation fails. Alternative placements and removal of z_3 are evaluated in Appendix A.7.

3.4 CASCADE-AWARE PREFERENCE LEARNING

Preference construction. For each question, two documents ordered by Eq. (3) form a core preference triple (q, d^+, d^-); ties and incomparable states are omitted. Training adds z_3 pairs under Eq. (4), yielding query-level comparisons without globally fixed relevance scores.

Per-query capped balancing. Enumerating all legal pairs yields a highly skewed training set, dominated by frequent transitions such as $z_2 \succ z_1$. Conversely, making pair types globally uniform can repeatedly sample rare states from only a few questions. For pair type t, let $\mathcal{P}_{q,t}$ be its legal pairs for question q and B_t a type-specific cap. We sample

$$\widetilde{\mathcal{P}}_{q,t} = \mathrm{Sample}(\mathcal{P}_{q,t}, \min(|\mathcal{P}_{q,t}|, B_t)) \tag{5}$$

and aggregate $\widetilde{\mathcal{P}} = \bigcup_{q,t} \widetilde{\mathcal{P}}_{q,t}$. The operation retains scarce query-level comparisons while limiting redundant frequent pairs. The pair-type caps B_t are provided in Appendix A.5.

Joint supervision of the cascade. The same trajectory preferences supervise the two ranking components, but each model retains its own architecture and candidate scope. The retriever uses dual-encoder similarity,

$$s_{\mathrm{ret}}(q, d) = \mathrm{sim}(E_q(q), E_d(d)), \tag{6}$$

whereas the cross-encoder reranker uses $s_{\mathrm{rank}}(q, d) = C_\phi(q, d)$. For the retriever, let $\mathcal{N}_q^+$ contain the paired lower-preference document and the in-batch negatives for d^+, and write $u_\tau(q, d) = s_{\mathrm{ret}}(q, d)/\tau$. We use the multi-negative contrastive objective

$$\begin{aligned} \mathcal{L}_{\mathrm{ret}} &= -\mathbb{E}\big[u_\tau(q, d^+) - \log Z_\tau(q, d^+)\big], \\ Z_\tau(q, d^+) &= \exp(u_\tau(q, d^+)) + \sum_{d^- \in \mathcal{N}_q^+} \exp(u_\tau(q, d^-)). \end{aligned} \tag{7}$$

where τ is the contrastive temperature. The reranker is optimized with the pairwise logistic objective

$$\mathcal{L}_{\mathrm{rank}} = -\mathbb{E}_{\widetilde{\mathcal{P}}} \log \sigma\big(s_{\mathrm{rank}}(q, d^+) - s_{\mathrm{rank}}(q, d^-)\big). \tag{8}$$

With one negative, Eq. (7) reduces to a temperature-scaled pairwise logistic loss. Retriever training promotes stronger trajectories into $\mathcal{C}_q$, and reranker training promotes them into $\mathcal{E}_q$. The separately optimized models are recomposed without additional inference modules or generation.

Although supervision is constructed within logged candidate pools, the optimized retriever can recover evidence missed by the original retriever, as shown by our held-out evidence-transition analysis (Figure 3(b)).

3.5 THEORETICAL ANALYSIS

We formalize **pipeline-aware evidence ranking**, which evaluates intermediate document selection by expected final-answer utility under a fixed downstream policy. Our analysis establishes three results: **local relevance alone can be insufficient**, with an arbitrarily poor worst-case approximation when downstream realization probabilities differ; under pointwise downstream behavior and additive evidence contributions, **optimal selection ranks documents by pipeline value**, $v_i^\pi(q, d) = p_i^\pi(q, d) r(q, d)$, combining potential evidence contribution with its probability of downstream realization; and when the interaction residual is uniformly bounded by ε_i over feasible document sets, the resulting pointwise selection has **regret at most** $2\varepsilon_i$. These results motivate accounting for downstream processing when learning intermediate rankings. Due to space constraints, the full formulation, assumptions, and proofs are presented in Appendix A.1.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Datasets and evaluation protocol. We evaluate on HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), and RAGBench-TATQA (Zhu et al., 2021). Their contexts are merged

Table 1: End-to-end retrieval and answer-generation results on HotpotQA, 2WikiMultiHopQA, and TAT-QA. Best results are in **bold**, and second-best results are underlined.

	HotpotQA				2Wiki				TAT-QA			
Method	**R@20**	**R@3**	**EM**	**F1**	**R@20**	**R@3**	**EM**	**F1**	**R@20**	**R@3**	**EM**	**F1**
Naive	0.6427	0.5945	0.3580	0.4425	0.5585	0.5095	0.2460	0.2938	0.4307	0.3232	0.0360	0.4191
W-RAG	0.6700	0.6220	0.3780	0.4708	0.5560	0.5000	0.2120	0.2597	0.3998	0.3468	0.0400	0.3958
REPLUG	0.6480	0.6040	0.3580	0.4489	0.5500	0.5030	0.2300	0.2773	0.4553	0.3368	0.0400	0.4279
Query2Doc	0.7690	0.6700	0.4040	0.5281	0.6450	0.5445	0.2560	0.3020	0.4222	0.3348	0.0360	0.4155
HyDE	0.7790	0.6780	0.4060	0.5253	0.6485	0.5565	0.2760	0.3235	0.4483	0.3348	0.0360	0.4290
RAG-Fusion	0.6870	0.6250	0.3740	0.4710	0.6185	0.5530	0.2740	0.3229	0.4460	0.3322	0.0400	0.4179
Naive + StageRAG	0.7310	0.6770	0.4340	0.5273	0.6850	0.6205	0.3500	0.4017	**0.5673**	0.4548	**0.0580**	**0.4922**
HyDE + StageRAG	**0.8140**	**0.7270**	**0.4560**	**0.5734**	**0.7195**	0.6235	0.3560	0.4042	0.5663	**0.4588**	**0.0580**	0.4886
RAG-Fusion + StageRAG	0.7660	0.6960	0.4260	0.5301	0.6880	**0.6255**	**0.3620**	**0.4128**	0.5353	0.4458	0.0540	0.4813

and deduplicated into global corpora, so retrieval is not restricted to question-specific contexts. All methods use the same fixed-seed random samples of 1,000 development and 500 test examples. We select checkpoints by development F1 (EM tie-breaker). Gold supporting-document annotations are used only for Recall and post-hoc analyses, never for trajectory construction, training, tuning, or model selection. Corpus statistics and preprocessing are in Appendix A.2.

Metrics. For gold-document set G_i and top-K results R_i^K, Recall@K averages $|G_i \cap R_i^K|/|G_i|$ across questions. Recall@20 measures candidate retrieval and Recall@3 the retriever–reranker cascade; normalized EM and token-level F1 measure answer quality. *All Gold@K* is the fraction of questions whose complete annotated evidence set appears in the top K.

Baselines. Trainable baselines share the retriever–reranker–generator backbone and complete training split. *Naive* is the frozen cascade. W-RAG (Nian et al., 2025) learns from gold-answer likelihood, whereas REPLUG (Shi et al., 2024) matches component scores to frozen-LM document preferences. We also compare Query2Doc (Wang et al., 2023), two-view HyDE (Gao et al., 2023), and RAG-Fusion (Rackauckas, 2024), which modify inference-time retrieval and are not compute-matched. Unless stated otherwise, methods share initialization, corpus, splits, retrieval depths, and answer model; implementation details are provided in Appendix A.3.

Backbones and implementation. Our primary cascade uses `all-MiniLM-L6-v2` and `ms-marco-MiniLM-L-6-v2` as retriever and reranker, with GPT-4o-mini for answer generation and verification (temperature zero), $K_r = 20$, $K_c = 3$, and contrastive temperature $\tau = 0.07$. Training and hardware details appear in Appendix A.4.

4.2 Overall Performance

Table 1 compares StageRAG with the original cascade, answer-supervised trainable baselines, and inference-time query-augmentation methods on all three benchmarks under the protocol above.

First, answer-supervised baselines. StageRAG outperforms both W-RAG and REPLUG on every reported metric across all three datasets. Compared with the stronger of these two baselines on each dataset, StageRAG improves F1 by 5.65 points on HotpotQA, 12.44 points on 2Wiki, and 6.43 points on TAT-QA, alongside improvements in both Recall@20 and Recall@3.

Second, gains throughout the retrieval cascade. StageRAG also consistently improves over the original Naive cascade. Across the three benchmarks, Recall@20 increases by 8.83–13.66 points, Recall@3 by 8.25–13.16 points, and answer F1 by 7.31–10.79 points.

Third, compatibility with query augmentation. StageRAG is complementary to inference-time query augmentation. On HotpotQA, HyDE + StageRAG achieves the best F1 (57.34). On 2Wiki, HyDE + StageRAG obtains the best Recall@20, whereas RAG-Fusion + StageRAG achieves the best Recall@3, EM, and F1. On TAT-QA, StageRAG alone retains the best Recall@20 and F1.

Table 2: Component ablation across three benchmarks; *w/o Both* denotes the unoptimized Naive cascade (**bold**: best; underlined: second-best).

	HotpotQA				2Wiki				TAT-QA			
Method	**R@20**	**R@3**	**EM**	**F1**	**R@20**	**R@3**	**EM**	**F1**	**R@20**	**R@3**	**EM**	**F1**
StageRAG	**0.7310**	**0.6770**	**0.4340**	**0.5273**	**0.6850**	**0.6205**	**0.3500**	**0.4017**	**0.5673**	**0.4548**	**0.0580**	**0.4922**
w/o Rerank Opt.	**0.7310**	0.6620	0.4000	0.4961	**0.6850**	0.5950	0.3120	0.3611	**0.5673**	0.3668	0.0360	0.4527
w/o Ret. Opt.	0.6427	0.6080	0.3800	0.4680	0.5585	0.5180	0.2440	0.2917	0.4307	0.3817	0.0460	0.4414
w/o Both	0.6427	0.5945	0.3580	0.4425	0.5585	0.5095	0.2460	0.2938	0.4307	0.3232	0.0360	0.4191

Table 3: Supervision and sampling ablations on HotpotQA test@500. StageRAG uses full-state preferences with per-query caps; its row is shared across all blocks and reproduces the main-table result (**bold**: best; underlined: second-best).

Variant	**R@20**	**R@3**	**EM**	**F1**
Cascade states (StageRAG)	**0.7310**	**0.6770**	**0.4340**	**0.5273**
Supervision scheme				
Binary z_4	0.6940	0.6550	0.4280	0.5216
Answer-only	0.7290	0.6760	0.4000	0.4987
Random pairs	0.3760	0.1320	0.1260	0.1620
Direct preference removal				
w/o $z_2 \succ z_1$ pairs	0.6880	0.6490	0.4280	0.5175
w/o $z_4 \succ z_2$ pairs	0.7200	0.6610	0.4180	0.5110
Pair sampling				
Uniform pair-type	0.6410	0.5020	0.3420	0.4174
Natural distribution	0.7160	0.6600	0.4020	0.5026

4.3 Ablation Studies

We conduct two controlled studies. Table 2 removes retriever and reranker optimization individually or jointly, producing StageRAG, w/o Rerank Opt., w/o Ret. Opt., and w/o Both. Table 3 compares supervision schemes and individual preference relations. Binary z_4, answer-only, and random preferences replace cascade-state supervision with the initialization, trainable components, pair budget, and optimization configuration held fixed. Two additional variants remove direct $z_2 \succ z_1$ or $z_4 \succ z_2$ pairs from the training preferences with the same training budget as the main experiment (Appendix A.6).

Complementary stage improvements. The two components address different cascade bottlenecks. Retriever optimization alone raises Recall@20 over Naive by 8.83–13.66 points, whereas adding reranker optimization further raises Recall@3 by 1.50–8.80 points and F1 by 3.12–4.06 points. Reranker-only training cannot change Recall@20 and yields smaller gains. Thus, retriever training increases candidate evidence availability, while reranker training improves final-context selection.

Benefits of cascade-state supervision. Compared with binary z_4 supervision, the full states improve Recall@20 and Recall@3 by 3.70 and 2.20 points. Answer-only supervision nearly matches retrieval recall but trails by 3.40 EM and 2.86 F1 points, showing that similar aggregate recall can yield contexts of different utility. Random preferences sharply degrade every metric, confirming that pairwise optimization depends on the extracted trajectory structure. Removing direct $z_2 \succ z_1$ pairs reduces Recall@20 and Recall@3 by 4.30 and 2.80 points relative to the full-method reference, with a 0.98-point F1 decrease. Removing $z_4 \succ z_2$ pairs reduces Recall@3 by 1.60 points and F1 by 1.63 points. These ablations support retaining direct preferences from both reranker selection and verified support in successful trajectories, with the larger recall decrease occurring when selection preferences are removed.

Additional pair-sampling analysis. The sampling block of Table 3 compares three strategies under the same protocol. Per-query caps improve F1 over natural and globally uniform pair-type sampling by 2.47 and 10.99 points, respectively, balancing frequent relations against repeated sampling from a few queries.

4.4 Robustness and Transferability

Table 4 evaluates StageRAG under three retrieval budgets and a separately initialized Qwen model family. We vary $K_r \in \{10, 20, 50\}$ while holding the remaining HotpotQA pipeline fixed, and then replace the MiniLM cascade with Qwen3 embedding, reranking, and answer models on TAT-QA.

Table 4: Robustness across retrieval budgets and model families. HotpotQA uses the MiniLM cascade, whereas TAT-QA uses the Qwen cascade; *w/o Both* denotes the corresponding unoptimized cascade (**bold**: best; underlined: second-best).

(a) Retrieval budget: HotpotQA, MiniLM

K_r	Method	R@K_r	R@3	EM	F1
10	w/o Both	0.5950	0.5670	0.3340	0.4221
	StageRAG	**0.6850**	**0.6590**	**0.4060**	**0.5046**
20	w/o Both	0.6427	0.5945	0.3580	0.4425
	StageRAG	**0.7310**	**0.6770**	**0.4340**	**0.5273**
50	w/o Both	0.7000	0.6280	0.3760	0.4720
	StageRAG	**0.7850**	**0.7070**	**0.4560**	**0.5580**

(b) Backbone: TAT-QA, Qwen

Metric	w/o Both	StageRAG
R@20	0.6417	**0.6627**
R@3	0.4178	**0.4768**
EM	0.0520	**0.0700**
F1	0.4859	**0.5040**

Robustness to retrieval budget. StageRAG improves F1 over w/o Both by 8.25–8.60 points for all three values of K_r. With only ten retrieved candidates, StageRAG also exceeds the baseline at $K_r = 50$ in both Recall@3 (65.90 vs. 62.80) and F1 (50.46 vs. 47.20), indicating that cascade optimization can reduce the candidate-processing budget.

Transfer across model families. Replacing the MiniLM cascade with Qwen3 models preserves the improvement: StageRAG raises TAT-QA Recall@3 from 41.78 to 47.68 and F1 from 48.59 to 50.40. The complete Qwen comparison, full retrieval-budget results, and sensitivity to the rare z_3 state are reported in Appendices A.9, A.8, and A.7, respectively.

4.5 Verifier Robustness and Annotation Cost

Table 5 evaluates verifier substitution on the same 5,000 HotpotQA training queries. We keep GPT-4o-mini answer generation fixed, construct supervision using each verifier, and train with matched budgets. The resulting models are evaluated on the same test@500 subset.

Table 5: Verifier substitution on HotpotQA. Cost estimates cover answer generation and verification only; **bold**/underline mark best/second-best values.

Verifier	Cost ($)↓	EM↑	F1↑
GPT-4o-mini	**0.799**	**0.4100**	**0.5120**
GPT-5.6 Luna	0.934	**0.4100**	0.5082
Grok 4.3	10.597	0.4080	0.5062

Stability across verifiers. F1 ranges from 50.62 to 51.20 and EM from 40.80 to 41.00. On the same training subset, 88.43% of GPT-4o-mini's z_4 documents match annotated gold evidence, covering 37.02% of all gold documents. Although 41.03–49.39% of queries contain no z_4 document, every query yields at least one preference pair. The remaining stage relations thus extend supervision beyond queries answered correctly with verified support by the base cascade. Appendix A.12 reports label coverage, cross-verifier agreement, and a three-dataset audit.

Annotation efficiency. StageRAG limits verification to explicitly cited documents within the top-three context, rather than labeling all retrieved candidates. With comparable document lengths and full-text inputs, labeling all top-20 candidates requires approximately $20/3 \approx 6.7$ times the document-input tokens of verifying three cited documents, even when the judgments are batched

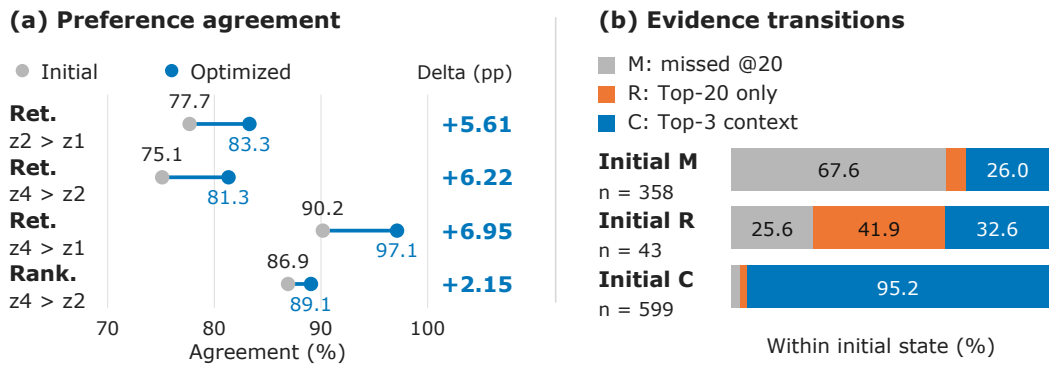


Figure 3: Preference transfer and evidence-state repair on HotpotQA. (a) Agreement with downstream preferences on fixed logged pairs before and after optimization, without gold labels. (b) Naive-to-StageRAG transitions of 1,000 gold documents among missing Top-20 (M), Top-20 only (R), and Top-3 context (C); gold is used only post hoc.

into a single call. This estimate concerns document-input tokens rather than total annotation cost. Under the provider pricing used in our experiments, GPT-4o-mini answer generation and verification cost an estimated $0.799 for the 5,000 queries. Relative to Grok 4.3 verification, this reduces verification cost by 95.8% and total annotation cost by 92.5%, with no lower observed EM or F1 in this comparison. These costs cover offline supervision construction; no verifier is added to the inference pipeline.

4.6 Cascade Diagnostics

Figure 3 examines whether learning downstream preferences translates into improved evidence coverage on HotpotQA. Panel (a) measures agreement with fixed trajectory preferences, while panel (b) tracks gold-document transitions from the original to the optimized cascade on the same held-out queries.

First, downstream preferences transfer across stages. In Figure 3(a), retriever agreement with the three downstream relations rises by 5.61–6.95 points after optimization, and reranker agreement on $z_4 > z_2$ rises by 2.15 points.

Second, cascade optimization recovers missed and filtered evidence. In Figure 3(b), of 358 gold documents missed by the Naive Top-20, the full cascade recovers 116 into Top-20, including 93 directly into Top-3. It also promotes 14 of the 43 documents previously truncated before Top-3 and preserves 570 of 599 documents already retained. At the stricter query level, All Gold@3 rises from 30.00% to 41.80%. Together, the two panels show improved alignment with downstream preferences alongside greater gold-evidence coverage in the final context. Exact query-level results and complete paired transition matrices are provided in Appendix A.10.

5 Conclusion

We present StageRAG, an answer-supervised framework for optimizing RAG cascades without gold supporting-document annotations. It converts observable document progression, verifier-derived support, and answer outcomes into balanced preferences for training both the retriever and reranker. Our theoretical analysis derives a cascade-aware pointwise ranking rule under idealized conditions and bounds its regret under bounded evidence interactions. Experiments across three benchmarks show improvements in retrieval and answer quality, while cascade diagnostics reveal better alignment with downstream preferences and recovery of supporting evidence into the final context. Together, these findings establish trajectory-derived supervision as a practical approach to adapting both retrieval stages while keeping the answer model and inference topology unchanged.

AI USE STATEMENT

Generative AI tools assisted with code development, manuscript drafting and language editing, internal review, and discussions of theoretical analysis and experimental design. The authors take responsibility for the final manuscript and reported results. Language models used as components of the experimental pipeline are documented in the method and experimental setup.

REFERENCES

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1762–1777, 2023. doi: 10.18653/v1/2023.acl-long.99. URL https://aclanthology.org/2023.acl-long.99/.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. Re2G: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2701–2715. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-main.194. URL https://aclanthology.org/2022.naacl-main.194/.

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/.

Kiseung Kim and Jay-Yoon Lee. RE-RAG: Improving open-domain QA performance and interpretability with relevance estimator in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 22149–22161, 2024. doi: 10.18653/v1/2024.emnlp-main.1236. URL https://aclanthology.org/2024.emnlp-main.1236/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvasy, Mike Lewis, Luke Zettlemoyer, and Scott Yih. RA-DIT: Retrieval-augmented dual instruction tuning. In *International Conference on Learning Representations*, pp. 19138–19162, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/536d18fbb454f80221465f1a42c6f389-Abstract-Conference.html.

Jinming Nian, Zhiyuan Peng, Qifan Wang, and Yi Fang. W-RAG: Weakly supervised dense retrieval in RAG for open-domain question answering, 2025. URL https://arxiv.org/abs/2408.08444.

Jiarui Qin, Jiachen Zhu, Bo Chen, Zhirong Liu, Weiwen Liu, Ruiming Tang, Rui Zhang, Yong Yu, and Weinan Zhang. Rankflow: Joint optimization of multi-stage cascade ranking systems as flows. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 814–824, 2022. doi: 10.1145/3477495.3532050. URL https://doi.org/10.1145/3477495.3532050.

Zackary Rackauckas. RAG-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, 13(1):37–47, 2024. doi: 10.5121/ijnlc.2024.13103. URL https://doi.org/10.5121/ijnlc.2024.13103.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, 2024. doi: 10.18653/v1/2024.naacl-long.463. URL https://aclanthology.org/2024.naacl-long.463/.

Liang Wang, Nan Yang, and Furu Wei. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9414–9423, 2023. doi: 10.18653/v1/2023.emnlp-main.585. URL https://aclanthology.org/2023.emnlp-main.585/.

Yuhang Wu, Xiangqing Shen, Fanfan Wang, Cangqi Zhou, Zhen Wu, Xinyu Dai, and Rui Xia. Optimizing RAG rerankers with LLM feedback via reinforcement learning. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 30474–30490, 2026. doi: 10.18653/v1/2026.acl-long.1406. URL https://aclanthology.org/2026.acl-long.1406/.

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/.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2421–2436, 2023. doi: 10.18653/v1/2023.acl-long.136. URL https://aclanthology.org/2023.acl-long.136/.

Ruwen Zhang, Bo Liu, Zhang Sheng Xiang, Yida Chen, Hantao Zhao, Ding Ding, Jiahui Jin, and Jiuxin Cao. Don't be misled by style: A style-adaptive reranker for capturing effective knowledge in retrieval-augmented generation. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6665–6681, 2026. doi: 10.18653/v1/2026.acl-long.302. URL https://aclanthology.org/2026.acl-long.302/.

Kai Zheng, Haijun Zhao, Rui Huang, Beichuan Zhang, Na Mou, Yanan Niu, Yang Song, Hongning Wang, and Kun Gai. Full stage learning to rank: A unified framework for multi-stage systems. In *Proceedings of the ACM Web Conference 2024*, pp. 3621–3631, 2024. doi: 10.1145/3589334.3645523. URL https://doi.org/10.1145/3589334.3645523.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3277–3287, 2021. doi: 10.18653/v1/2021.acl-long.254. URL https://aclanthology.org/2021.acl-long.254/.

A TECHNICAL APPENDIX

A.1 THEORETICAL ANALYSIS

We formalize a pipeline-aware ranking objective and analyze pointwise selection under separable and bounded-interaction models.

A.1.1 PIPELINE-AWARE EVIDENCE RANKING

At stage i of an m-stage cascade, select K_i documents from S_{i-1}. Let $\mathcal{T}^{\pi}_{i+1:m}(q, S; \omega)$ denote the final context after setting stage i's output to S and executing fixed downstream policy $\pi_{i+1:m}$, with ω capturing candidate, inference, and generation randomness. For answer utility $U(\hat{a}, a^*) \in [0, 1]$, define

$$Y^{\pi}_i(q, S; \omega) = U\big(G\big(q, \mathcal{T}^{\pi}_{i+1:m}(q, S; \omega)\big), a^*\big) \tag{9}$$

and its expectation

$$J^{\pi}_i(q, S) = \mathbb{E}_{\omega}[Y^{\pi}_i(q, S; \omega)]\,. \tag{10}$$

The ideal output of stage i is

$$S^*_i = \arg \max_{\substack{S \subseteq S_{i-1} \\ |S| = K_i}} J^{\pi}_i(q, S). \tag{11}$$

This *pipeline-aware evidence ranking principle* values intermediate decisions by their downstream answer utility.

Proposition S1 (Failure of isolated relevance ranking)**.** *If downstream realization probabilities vary across documents, ranking only by local evidence utility can have an approximation ratio arbitrarily close to zero relative to Eq.* (11)*.*

For $K_i = 1$, documents with local utilities $(1, 1 - \delta)$ and realization probabilities $(\epsilon, 1)$ give local ranking an approximation ratio $\epsilon/(1 - \delta) \to 0$. This establishes a worst-case separation. Proofs are provided in Section A.1.4 below.

A.1.2 POINTWISE PIPELINE VALUE

For $d \in S$, the latent indicator $A^{\pi}_i(q, d; S, \omega) \in \{0, 1\}$ records survival through the fixed downstream policy and contribution to the final outcome; it is distinct from verifier attribution. Define

$$p^{\pi}_i(q, d \mid S) = \mathbb{E}_{\omega}[A^{\pi}_i(q, d; S, \omega)]. \tag{12}$$

We first assume set-independent contribution probabilities and additive evidence contributions to isolate the effect of downstream processing. We then capture departures from this separable model through an interaction residual.

Assumption 1 (Pointwise downstream behavior)**.** *For every feasible S containing d, $p^{\pi}_i(q, d \mid S) = p^{\pi}_i(q, d)$.*

Assumption 2 (Additive evidence contribution)**.** *Conditioned on the effective-contribution indicators,*

$$\mathbb{E}[Y^{\pi}_i(q, S; \omega) \mid q, S, \mathbf{A}_i] = \sum_{d \in S} A^{\pi}_i(q, d; S, \omega) r(q, d), \tag{13}$$

where $r(q, d) \geq 0$ is the potential contribution of d.

Theorem 1 (Pointwise pipeline value)**.** *Under Assumptions 1 and 2,*

$$J^{\pi}_i(q, S) = \sum_{d \in S} p^{\pi}_i(q, d) r(q, d). \tag{14}$$

Consequently, Eq. (11) *is solved by selecting the K_i documents with the largest*

$$v^{\pi}_i(q, d) = p^{\pi}_i(q, d) r(q, d). \tag{15}$$

This factorization parallels GPRP (Zheng et al., 2024), replacing item exposure with downstream evidence contribution. It separates a document's potential contribution from its probability of realization.

A.1.3 Robustness to Bounded Evidence Interactions

To quantify a controlled departure from additivity, write

$$J_i^\pi(q,S) = \widetilde{J}_i^\pi(q,S) + \Delta_i(q,S),$$
$$\widetilde{J}_i^\pi(q,S) = \sum_{d\in S} p_i^\pi(q,d) r(q,d), \tag{16}$$

where Δ_i collects non-additive evidence interactions.

Proposition S2 (Approximation under bounded interactions)**.** *Let S_i^* maximize J_i^π and let $\widetilde{S}_i$ maximize $\widetilde{J}_i^\pi$ over the same feasible sets. If $|\Delta_i(q,S)| \le \varepsilon_i$ for every feasible S, then*

$$J_i^\pi(q,S_i^*) - J_i^\pi(q,\widetilde{S}_i) \le 2\varepsilon_i. \tag{17}$$

The bound relates the loss of pointwise selection to the magnitude of the interaction residual, recovering exact optimality when the residual vanishes. The proof is provided in Section A.1.4 below.

A.1.4 Proofs of the Theoretical Results

Proposition S3 (Reduction to local relevance ranking)**.** *Suppose no downstream component further filters the output of stage i and $J_i^\pi(q,S) = \sum_{d\in S} r(q,d)$. Then the pipeline-aware objective defined in Eq.* (11) *is solved by selecting the K_i documents with the largest values of $r(q,d)$.*

Proof of Proposition S3. Fix a question and let T contain the K_i candidates with the largest values of $r(q,d)$. Suppose a size-K_i set $S \neq T$ is optimal. There must exist $d^- \in S \setminus T$ and $d^+ \in T \setminus S$ such that $r(q,d^+) \ge r(q,d^-)$. Under the assumed additive utility, replacing d^- with d^+ changes the objective by

$$J_i^\pi(q, S - \{d^-\} + \{d^+\}) - J_i^\pi(q,S)$$
$$= r(q,d^+) - r(q,d^-) \ge 0. \tag{18}$$

Repeated exchanges transform S into T without decreasing utility. Hence a top-K_i set under $r(q,d)$ is optimal; it is unique when the scores at the selection boundary are distinct. □

Proof of Proposition S1. It suffices to construct a two-document instance with $K_i = 1$. Let

$$r(q,d_a) = 1, \qquad r(q,d_b) = 1-\delta, \tag{19}$$

where $0 < \delta < 1$. Local evidence utility ranks d_a above d_b. Now consider a fixed downstream pipeline in which the probabilities that the two documents survive and are effectively used are ϵ and 1, respectively, with $0 < \epsilon < 1-\delta$. If effective use realizes the local utility and filtering realizes zero utility, then

$$J_i^\pi(q,\{d_a\}) = \epsilon, \qquad J_i^\pi(q,\{d_b\}) = 1-\delta. \tag{20}$$

The local rule selects d_a, whereas the pipeline-aware rule selects d_b. Their utility ratio is $\epsilon/(1-\delta)$, which approaches zero as $\epsilon \to 0$ for any fixed δ. Moreover, the absolute gap $1-\delta-\epsilon$ approaches one as both δ and ϵ approach zero. Thus no positive constant-factor end-to-end guarantee exists for local ranking without additional restrictions on downstream behavior. □

Proof of Theorem 1. Applying iterated expectation under the additive evidence contribution assumption gives

$$\begin{aligned} J_i^\pi(q,S) &= \mathbb{E}_\omega\left[\sum_{d\in S} A_i^\pi(q,d;S,\omega) r(q,d)\right] \\ &= \sum_{d\in S} \mathbb{E}_\omega[A_i^\pi(q,d;S,\omega)] r(q,d) \\ &= \sum_{d\in S} p_i^\pi(q,d \mid S) r(q,d) \\ &= \sum_{d\in S} p_i^\pi(q,d) r(q,d), \end{aligned} \tag{21}$$

where the final equality follows from the pointwise downstream behavior assumption. The objective is now additive in $v_i^\pi(q,d) = p_i^\pi(q,d)r(q,d)$. The exchange argument used in the proof of Proposition S3 therefore shows that selecting the K_i largest values of v_i^π is optimal. □

Proof of Proposition S2. By the optimality of $\widetilde{S}_i$ for the pointwise surrogate,

$$\widetilde{J}_i^\pi(q, S_i^*) - \widetilde{J}_i^\pi(q, \widetilde{S}_i) \le 0. \tag{22}$$

Using the decomposition $J_i^\pi(q,S) = \widetilde{J}_i^\pi(q,S) + \Delta_i(q,S)$,

$$\begin{aligned} &J_i^\pi(q, S_i^*) - J_i^\pi(q, \widetilde{S}_i) \\ &= \widetilde{J}_i^\pi(q, S_i^*) - \widetilde{J}_i^\pi(q, \widetilde{S}_i) + \Delta_i(q, S_i^*) - \Delta_i(q, \widetilde{S}_i) \\ &\le |\Delta_i(q, S_i^*)| + |\Delta_i(q, \widetilde{S}_i)| \le 2\varepsilon_i. \end{aligned} \tag{23}$$

□

A.2 Dataset Construction and Evaluation Splits

Table 6: Dataset split and retrieval-corpus statistics.

Dataset	Train	Dev	Test pool	Evaluated test	Corpus
HotpotQA	80,000	10,447	7,405	500	507,494
2WikiMultiHopQA	150,000	17,454	12,576	500	414,743
RAGBench-TATQA	26,430	3,336	3,338	500	12,714

The test pool is the complete available evaluation split, whereas evaluated test denotes the common 500-example subset used for every reported method. For HotpotQA, we take the first 80,000 examples of the official 90,447-example training split for training and use the remaining 10,447 as internal development data. The official 7,405-example validation split forms the test pool. We merge passages from the official training and validation splits and deduplicate them by title, producing 507,494 retrieval passages. For 2WikiMultiHopQA, the first 150,000 examples of the 167,454-example official training split are used for training and the remaining 17,454 for development; the 12,576-example official development split forms the test pool. Deduplication by passage title and content produces 414,743 retrieval passages.

For RAGBench-TATQA, we retain the standardized RAGBench splits containing 26,430 training, 3,336 validation, and 3,338 test examples. Deduplicating the contexts across these splits yields 12,714 retrieval passages. This benchmark differs from the two Wikipedia-based datasets in requiring joint reasoning over financial tables and text, often including arithmetic and percentage answers.

For every dataset, model selection uses a fixed-seed random sample of 1,000 examples from the development split, and final evaluation uses a fixed-seed random sample of 500 examples from the test pool. These subsets are sampled once before evaluation and shared by all methods. Supporting-document annotations are excluded from label generation, training, hyperparameter tuning, and model selection; they are consulted only for retrieval evaluation and post-hoc diagnostics.

A.3 Baseline Implementations

All standalone baselines use the same initial retriever, reranker, corpus, splits, top-20 retrieval depth, top-3 reranking depth, and final answer model. The trainable answer-supervised baselines use the complete training split of each dataset. In the main MiniLM comparisons, W-RAG, REPLUG, and StageRAG all use GPT-4o-mini to produce their respective supervision signals. For each dataset and trainable component, the three methods share the same initialization, learning rate, batch size, optimizer, learning-rate schedule, and optimizer-update budget, as detailed in Appendix A.4. The methods differ in how they construct supervision and in their training objectives, not in the supervision model or training budget.

W-RAG. We retrieve 20 candidates and use GPT-4o-mini to score the average token log-likelihood of the reference answer conditioned on each document. The highest-scoring document is treated as positive and up to eight low-scoring documents as negatives. These pairs supervise both retrieval components: the retriever is optimized with contrastive loss and the reranker with pairwise logistic loss. Both components follow the shared training configuration and matched update budgets described above.

REPLUG. Our pipeline-matched implementation uses GPT-4o-mini to compute reference-answer likelihoods for all 20 candidates and transform them into a teacher distribution with temperature 0.2. The retriever and reranker are both optimized to match these induced soft document preferences; retriever similarities use a student temperature of 0.07, and cross-encoder scores are optimized over the same candidate lists. Both components follow the same shared training configuration and matched update budgets as W-RAG and STAGERAG.

Inference-time query augmentation. Query2Doc generates one hypothetical document, concatenates it with the original question, and performs one retrieval. Two-view HyDE retrieves separately with the original question and the hypothetical document, merges duplicates by their maximum dense score, and truncates the union to 20 candidates. It uses GPT-4o-mini with temperature 0.2 and a 160-token limit; the TAT-QA prompt contains two financial evidence-sketch demonstrations. RAG-Fusion generates three rewrites with GPT-4o-mini at temperature 0.2 and a 96-token limit. Together with the original question, they produce four retrieval lists, which are fused by

$$\mathrm{RRF}(d) = \sum_{q_i} \frac{1}{60 + \mathrm{rank}_{q_i}(d)}. \tag{24}$$

The fused list is truncated to 20 candidates before standard reranking.

A.4 BACKBONES AND TRAINING CONFIGURATION

The MiniLM retriever is `sentence-transformers/all-MiniLM-L6-v2`, a six-layer encoder with 384-dimensional embeddings. We use mean pooling, L2 normalization, and cosine similarity implemented by `FAISS IndexFlatIP`. Its maximum inference length is 256 and maximum training length is 128. The reranker is `cross-encoder/ms-marco-MiniLM-L-6-v2` with maximum length 512. GPT-4o-mini is accessed through an OpenAI-compatible API and is used for answer generation and verification with temperature zero and no API seed.

Table 7: Training configurations for the MiniLM cascade. Pair counts and training schedules are reported for STAGERAG; W-RAG and REPLUG use the same hyperparameters and match its optimizer-update budget for each dataset and component.

Dataset	Component	Pairs	Batch	LR	Warmup	Training
HotpotQA	Retriever	412,338	64	1×10^{-6}	0.10	10 epochs
HotpotQA	Reranker	412,338	32	5×10^{-6}	0.10	1 epoch
2Wiki	Retriever	557,638	32	2×10^{-5}	0.10	4,000 steps
2Wiki	Reranker	557,638	16	1×10^{-5}	0.10	7,000 steps
TAT-QA	Retriever	97,956	32	1×10^{-5}	0.05	3,062 steps
TAT-QA	Reranker	15,780	16	1×10^{-5}	0.05	987 steps

For STAGERAG, retriever optimization uses multi-negative contrastive loss with temperature 0.07, whereas reranker optimization uses pairwise logistic loss. All MiniLM runs use FP32, AdamW, weight decay 0.01, cosine decay, and random seed 42. Every run completes its matched optimizer-update budget without terminating early; Table 7 specifies the reference schedules for STAGERAG. We save intermediate checkpoints and select the one with the highest development F1, using EM as a tie-breaker. The selected checkpoint is evaluated once on the fixed test subset.

For the Qwen compatibility experiment, the retriever is `Qwen/Qwen3-Embedding-0.6B`, using last-token pooling, L2 normalization, and a retrieval instruction; the reranker is `Qwen/Qwen3-Reranker-0.6B`; and `Qwen/Qwen3-8B` serves as the answer and judge model.

Retriever LoRA uses rank 16, alpha 32, dropout 0.05, all attention and MLP projections, BF16, batch size 64, learning rate 5×10^{-6}, and 1,200 steps. Reranker LoRA uses rank 8, alpha 32, dropout 0.05, Q/K/V/O projections, BF16, batch size 32, learning rate 2×10^{-6}, and 1,000 steps.

Experiments run on a server with eight NVIDIA H100 80GB HBM3 PCIe GPUs. Each retriever or reranker run uses one GPU; independent hyperparameter configurations are executed in parallel.

A.5 Generation, Verification, and Cascade-State Construction

Cascade states are generated once offline and remain fixed during training. Each training question is executed once through the retriever, reranker, answer model, and verifier. The answer model generates an answer with citations; the verifier receives the question, reference answer, generated answer, and cited document texts. It returns a Boolean answer-correctness decision and the indices of citations that support the predicted answer. The structured output contains two fields: `answer_correct`, a Boolean, and `supported_citations`, a list of cited-document indices. Only explicitly cited documents verified as supporting the prediction receive support attribution. The same fixed generation and verification protocol is used for all methods that require trajectory construction. Calls are retried up to three times; a persistent failure is conservatively treated as incorrect and unsupported. This verifier decision constructs z_3 and z_4; evaluation EM and F1 do not enter cascade-state construction.

The main experiments use four observed states: retrieved but filtered (z_1), retained but not attributed as support (z_2), attributed as support in an unsuccessful trajectory (z_3), and attributed as support in a successful trajectory (z_4). These attributions are produced by the answer and citation verifier and do not reveal the generator's internal use of a document. The per-query caps are $B_{4,2} = 2$, $B_{4,1} = 4$, $B_{2,1} = 2$, $B_{2,3} = 2$, and $B_{1,3} = 2$, where $B_{u,v}$ caps pairs of type $z_u \succ z_v$. Retriever and reranker training share the same offline state records. TAT-QA reranker training additionally restricts pair construction to the stages associated with the retained top-3 context.

A.6 Construction of the Supervision Ablations

The supervision ablations reported in the main paper use the same 80,000 HotpotQA training trajectories as the complete method. Each variant is sampled to 412,338 pairs with random seed 42 and contributes at most 12 pairs per question. The model initialization, trainable components, optimization objectives, and training configuration are otherwise held fixed; only the rule used to construct preference pairs changes.

Binary z_4. We retain questions containing at least one z_4 document and construct preferences

$$d^+ \in z_4, \qquad d^- \in \{z_1, z_2, z_3\}. \tag{25}$$

The variant therefore distinguishes the successful outcome state from all other observed states but discards the differences among z_1, z_2, and z_3. It covers 41,947 questions and produces 503,364 pairs before downsampling.

Answer-only. We retain trajectories that the answer verifier judges to be correct. Each document in the final top-three context is paired as preferred against the remaining retrieved candidates from the same question. This construction uses answer correctness but ignores citation attribution, the cascade-state ordering, and gold supporting-document labels. It covers 42,594 questions and produces 511,128 pairs before downsampling.

Random pairs. For each question, we sample two distinct observed candidates and randomly assign their preference direction. This matched-size variant contains no meaningful trajectory supervision and serves as a negative control.

Relation removal. Two variants remove direct $z_2 \succ z_1$ pairs or direct $z_4 \succ z_2$ pairs, respectively, while retaining the other preference relations. Both use the same training budget and optimization configuration as the main HotpotQA experiment; only the available preference relations change.

A.7 Sensitivity to the Ambiguous State z_3

Table 8: Sensitivity to z_3 treatment on HotpotQA test@500 (**bold**: best; underlined: second-best).

z_3 **Treatment**	**R@20**	**R@3**	**EM**	**F1**
Failure-sensitive (default)	0.7310	**0.6770**	0.4180	0.5208
Progression-sensitive	**0.7320**	0.6740	**0.4300**	**0.5308**
Drop z_3	0.7250	0.6740	0.4180	0.5223

Table 8 compares the default failure-sensitive ordering with a progression-sensitive ordering, $z_4 > z_3 > z_2 > z_1$, and a variant that excludes all pairs involving z_3. The progression-sensitive variant obtains the highest test F1, exceeding the default by 1.00 point, but slightly reduces Recall@3. The answer-generation metrics in this table come from the common evaluation run used for this sensitivity comparison and may differ slightly from the independently evaluated main-table result. Removing z_3 also produces comparable results. On development data, paired bootstrap 95% confidence intervals for the F1 differences of progression-sensitive minus failure-sensitive $[-0.0145, 0.0064]$, drop-z_3 minus failure-sensitive $[-0.0181, 0.0051]$, and drop-z_3 minus progression-sensitive $[-0.0122, 0.0070]$ all include zero. The state is also rare: z_3 accounts for 825 of 1.6 million candidate records (0.052%) and occurs in 646 of 80,000 training questions (0.81%). These results indicate that performance is largely insensitive to its precise treatment. The main gains therefore arise primarily from the more frequent z_4, z_2, and z_1 supervision rather than from a particular placement of z_3.

Interpretation. The theoretical analysis does not derive or validate the core relation $z_4 \succ_{\text{core}} z_2 \succ_{\text{core}} z_1$. Instead, StageRAG uses it as an empirical ordinal surrogate for the latent pipeline value. The full ordering in the main experiments instantiates a failure-sensitive operating preference rather than a universal semantic relevance hierarchy. In particular, z_3 is ambiguous because a document attributed as support in an unsuccessful trajectory may still provide valid evidence. The sensitivity results support treating its placement as an empirical extension rather than a source of gold semantic relevance.

A.8 Additional Robustness Results

Table 9: Retrieval-budget sensitivity on HotpotQA test@500; methods are compared at the same K_r (**bold**: best; underlined: second-best).

Method	K_r	**Recall@**K_r	**R@3**	**EM**	**F1**
Naive	10	0.5950	0.5670	0.3340	0.4221
StageRAG	10	**0.6850**	**0.6590**	**0.4060**	**0.5046**
Naive	20	0.6427	0.5945	0.3580	0.4425
StageRAG	20	**0.7310**	**0.6770**	**0.4340**	**0.5273**
Naive	50	0.7000	0.6280	0.3760	0.4720
StageRAG	50	**0.7850**	**0.7070**	**0.4560**	**0.5580**

Table 10: Component ablation on TAT-QA test@500 with a Qwen-based pipeline; *w/o Both* denotes the unoptimized Naive cascade (**bold**: best; underlined: second-best).

Method	**R@20**	**R@3**	**EM**	**F1**
StageRAG	**0.6627**	**0.4768**	**0.0700**	**0.5040**
w/o Rerank Opt.	**0.6627**	0.4398	0.0580	0.4935
w/o Ret. Opt.	0.6417	0.4528	**0.0700**	0.5038
w/o Both	0.6417	0.4178	0.0520	0.4859

Table 12: Cross-stage preference agreement on held-out HotpotQA trajectories.

Component	Transferred signal	Preference	#Queries / #Pairs	Initial	Optimized	Δ
Retriever	Reranker retention	$z_2 > z_1$	498 / 20,009	77.68	**83.29**	**+5.61**
Retriever	Successful generation	$z_4 > z_2$	233 / 466	75.11	**81.33**	**+6.22**
Retriever	Full cascade	$z_4 > z_1$	235 / 5,440	90.17	**97.13**	**+6.95**
Reranker	Successful generation	$z_4 > z_2$	233 / 466	86.91	**89.06**	**+2.15**

Table 13: Query-level pipeline funnel on HotpotQA test@500. All Gold@K requires all annotated supporting documents in the top K (**bold**: best; underlined: second-best).

Method	All Gold in Top-20	All Gold in Top-3	Correct | All Gold in Top-3
Naive	0.3760 (188/500)	0.3000 (150/500)	0.6400 (96/150)
Ours w/o Rerank Opt.	**0.5020 (251/500)**	0.3840 (192/500)	0.6406 (123/192)
Ours w/o Ret. Opt.	0.3760 (188/500)	0.3200 (160/500)	**0.6562 (105/160)**
StageRAG	**0.5020 (251/500)**	**0.4180 (209/500)**	0.6316 (132/209)

Table 9 provides the complete retrieval-budget results summarized in the main paper. Table 10 separates retriever and reranker optimization in the Qwen-based TAT-QA pipeline.

A.9 Full Cross-Model Compatibility Results

Table 11: Full comparison on TAT-QA test@500 with a Qwen-based pipeline (**bold**: best; underlined: second-best).

Method	R@20	R@3	EM	F1
Naive	0.6417	0.4178	0.0520	0.4859
W-RAG	0.6437	0.4178	0.0500	0.4839
REPLUG	0.6387	0.4178	0.0500	0.4862
Query2Doc	**0.6713**	0.4228	0.0560	0.4866
HyDE	0.6593	0.4118	0.0540	0.4874
RAG-Fusion	0.6337	0.4158	0.0380	0.4623
StageRAG	0.6627	0.4768	**0.0700**	0.5040
HyDE + StageRAG	0.6703	**0.4828**	**0.0700**	**0.5071**
RAG-Fusion + StageRAG	0.6447	0.4688	0.0460	0.4854

A.10 Extended Cascade Diagnostics

Cross-stage preference transfer. For observed pair set $\mathcal{P}_{u>v}$ and component score s_j, we compute pairwise agreement as

$$\mathrm{Agr}^{(j)}_{u>v} = \frac{1}{|\mathcal{P}_{u>v}|} \sum_{(q,d_u,d_v)\in\mathcal{P}_{u>v}} \mathbf{1}[s_j(q,d_u) > s_j(q,d_v)].$$

Pairs are fixed when scoring initial and optimized components, and no gold supporting-document labels enter this diagnostic. Table 12 shows that retriever agreement increases for all three downstream relations, while reranker agreement also increases for $z_4 > z_2$. Reranker comparisons against z_1 are omitted because z_1 is defined by exclusion from the initial reranker's top three. Relations involving z_3 are too sparse for a reliable held-out agreement estimate.

Pipeline funnel. Table 13 traces the stricter All Gold measure. Retriever optimization improves complete evidence availability at top-20, and reranker optimization promotes available evidence into the top-three context. Complete evidence remains insufficient for some questions, reflecting the fixed answer model's remaining reasoning and generation errors.

Table 14: Post-hoc cascade repair on HotpotQA test@500. Queries are grouped by their Naive evidence status; All Gold@3 requires complete annotated evidence in the final context. Cells report percentages; gold labels are used only for analysis.

Naive status	**#**	**Naive**	**StageRAG**	**Δ**
Missing ≥1 gold from Top-20	312	0.00	19.87	+19.87
Complete Top-20, incomplete Top-3	38	0.00	28.95	+28.95
Complete Top-3	150	100.00	90.67	-9.33
Overall	**500**	**30.00**	**41.80**	**+11.80**

Table 15: Document-level gold-evidence transitions from the Naive cascade to retriever-only optimization. M, R, and C denote missed from Top-20, retrieved but excluded from Top-3, and retained in Top-3, respectively. Each cell reports count (row-normalized percentage).

Naive / Retriever-only	M	R	C
M	242 (67.60%)	34 (9.50%)	82 (22.91%)
R	11 (25.58%)	30 (69.77%)	2 (4.65%)
C	16 (2.67%)	5 (0.83%)	578 (96.49%)

Table 16: Document-level gold-evidence transitions from the Naive cascade to reranker-only optimization. M, R, and C denote missed from Top-20, retrieved but excluded from Top-3, and retained in Top-3, respectively. Each cell reports count (row-normalized percentage).

Naive / Reranker-only	M	R	C
M	358 (100.00%)	0 (0.00%)	0 (0.00%)
R	0 (0.00%)	25 (58.14%)	18 (41.86%)
C	0 (0.00%)	9 (1.50%)	590 (98.50%)

Table 17: Document-level gold-evidence transitions from the Naive cascade to the full StageRAG cascade. M, R, and C denote missed from Top-20, retrieved but excluded from Top-3, and retained in Top-3, respectively. Each cell reports count (row-normalized percentage).

Naive / StageRAG	M	R	C
M	242 (67.60%)	23 (6.42%)	93 (25.98%)
R	11 (25.58%)	18 (41.86%)	14 (32.56%)
C	16 (2.67%)	13 (2.17%)	570 (95.16%)

Gold-evidence flow. Table 14 groups queries by the status of their annotated evidence under the Naive cascade. Here we provide the corresponding document-level transitions. We denote a gold document missed from Top-20 by M, one retrieved but excluded from Top-3 by R, and one retained in Top-3 by C. The matrices use the same 500 aligned test queries and 1,000 gold documents; each cell gives a count and its rate conditional on the Naive row state. They reveal both recoveries and regressions. For example, retriever-only optimization recovers 116 initially missed gold documents into Top-20 ($M \rightarrow R/C$), while reranker-only optimization leaves Top-20 unchanged and promotes 18 initially truncated documents to Top-3 ($R \rightarrow C$). The full cascade directly moves 93 initially missed gold documents into Top-3, but also demotes 13 initially retained documents from Top-3 to Top-20. As throughout the paper, gold annotations are used only for post-hoc analysis.

A.11 Relation to Prior Cascade Optimization

StageRAG shares with RankFlow and FS-LTR the general principle that an upstream component should account for decisions made later in a cascade. However, the three methods differ in their supervision, optimization mechanisms, and target systems.

RankFlow (Qin et al., 2022) jointly trains multiple ranking stages through iterative self-learning and tutor-learning flows. A succeeding ranker serves as a teacher by providing document-level scores to its preceding stage, while logged impression feedback supplies additional supervision. This mechanism assumes that adjacent components are rankers whose outputs can be compared or distilled. A RAG generator does not provide such a document-ranking interface: it produces an answer from a set of documents and may fail even when valid evidence is available. STAGERAG therefore does not distill one component into another. Instead, it converts the observable progression of documents through a heterogeneous retriever–reranker–generator cascade into a shared set of pairwise preferences. These preferences are collected once from the base cascade and used to optimize the retriever and reranker, while the generator remains fixed.

FS-LTR (Zheng et al., 2024) assigns scalar labels according to the deepest ranking stage reached by an item and its eventual user feedback. Its theoretical motivation combines downstream selection probability with item-level user utility. STAGERAG adopts the related intuition that later cascade behavior can inform earlier stages, but the RAG setting introduces a different supervision boundary. Its states depend on retrieval, final-context retention, verifier-derived support attribution, and answer correctness, and require only reference answers rather than gold supporting-document annotations. Moreover, RAG utility is defined over evidence sets: documents may be complementary, conflicting, or individually insufficient, and a generator may produce an incorrect answer despite receiving valid evidence. For this reason, our pointwise analysis is presented only under explicit approximation conditions, and the ambiguous failure state z_3 is treated as an empirically evaluated design choice rather than a theoretically guaranteed relevance level.

Thus, the contribution of STAGERAG is not generic joint optimization of a cascade, which has been studied by prior work. It is an answer-supervised mechanism for transferring trajectory information across the heterogeneous components of a fixed RAG cascade without gold document labels or additional inference-time modules.

A.12 VERIFIER AGREEMENT, COVERAGE, AND COST

Matched-query substitution. We use the same 5,000 HotpotQA training queries and fixed GPT-4o-mini-generated answers to compare GPT-4o-mini, GPT-5.6 Luna, and Grok 4.3 as verifiers. Each verifier supplies answer-correctness and support-attribution decisions for preference construction. Training budgets are matched, while the number of resulting preference pairs varies with the verifier's decisions. Table 18 reports document counts and query-level coverage.

Let $Z_4(q)$ denote the documents assigned to z_4, G_q the gold-document set, and $\mathcal{E}_q$ the logged Top-3 context. Gold agreement is measured over query–document instances:

$$P_4 = \frac{\sum_q |Z_4(q) \cap G_q|}{\sum_q |Z_4(q)|}, \qquad R_4 = \frac{\sum_q |Z_4(q) \cap G_q|}{\sum_q |G_q|}, \tag{26}$$

$$R_{4|\mathcal{E}} = \frac{\sum_q |Z_4(q) \cap G_q|}{\sum_q |G_q \cap \mathcal{E}_q|}. \tag{27}$$

Gold-set sizes may vary across queries. These metrics measure alignment with annotated evidence, whereas the verifier evaluates support for the generated answer. Correctness agreement is the fraction of queries with identical binary answer judgments; support-set agreement requires identical sets of attributed documents.

Cross-model audit across datasets. GPT-5.6 Luna audits the original GPT-4o-mini judgments on 200 queries per dataset (600 total). Support confirmation is computed only over identifiable originally cited documents that GPT-4o-mini judged supportive, not over all Top-3 documents. Separately, we compare the original attributed support set with gold evidence, including support from both correct and incorrect answer trajectories rather than restricting evaluation to z_4. Table 19 shows that answer judgments agree more closely than support attribution, with the lowest support confirmation on 2Wiki. Cross-model agreement measures consistency, not human-verified accuracy.

Cost accounting. Estimated verification costs are \$0.434, \$0.569, and \$10.231 for GPT-4o-mini, GPT-5.6 Luna, and Grok 4.3, respectively. Including the shared GPT-4o-mini answer-generation stage gives total annotation costs of \$0.799, \$0.934, and \$10.597. Estimates use the service-provider

Table 18: Verifier-derived labels on the matched HotpotQA training subset. Agreement is measured against GPT-4o-mini.

Metric	GPT-4o-mini	GPT-5.6 Luna	Grok 4.3
z_4 documents	4,185	3,696	3,511
Gold documents within z_4	3,701	3,335	3,175
P_4 (%)	88.43	90.23	90.43
R_4 (%)	37.02	33.36	31.76
$R_{4\|\mathcal{E}}$ (%)	63.91	57.59	54.83
Answer-judgment agreement (%)	—	98.82	98.70
Exact support-set agreement (%)	—	90.64	87.04
Queries with z_4 (%)	58.97	52.43	50.61
Queries without z_4 (%)	41.03	47.57	49.39
Queries with a preference pair (%)	100.00	100.00	100.00
Training pairs	27,674	25,672	25,022

Table 19: Cross-model audit on 200 queries per dataset. All values are percentages. Gold precision evaluates GPT-4o-mini's original support set.

Dataset	Answer agreement	Support confirmation	Gold precision
HotpotQA	99.50	92.25	89.80
2Wiki	100.00	77.38	88.78
TAT-QA	97.00	95.29	46.74

pricing applied to these runs; they exclude model training, test evaluation, and the separate audit. These are subset-level measurements rather than full-dataset cost estimates.

A.13 LIMITATIONS

STAGERAG inherits the support of its logging retriever: evidence absent from logged candidate pools receives no direct state label, so its recovery depends on representation generalization. Support attribution is verifier-derived and does not reveal the generator's internal or causal use of a document; verifier errors can therefore introduce preference noise. The practical treatment of the rare state z_3 is an empirical design choice rather than a universal relevance relation. The theoretical pointwise and additive conditions are idealizations: multi-hop complementarity, conflict, and order sensitivity may violate them. The bounded-interaction proposition only bounds the resulting regret when the residual interaction is uniformly bounded; it does not establish that this condition holds on the evaluated tasks. Finally, the answer generator remains fixed, and complete annotated evidence does not eliminate reasoning and generation errors.