Adaptive Retrieval for Reasoning-Intensive Retrieval

Anonymous ACL submission

Abstract

We study leveraging adaptive retrieval to ensure sufficient "bridge" documents are retrieved for reasoning-intensive retrieval. Bridge documents are those that contribute to the reasoning process yet are not directly relevant to the initial query. While existing reasoning-based reranker pipelines attempt to surface these documents in ranking, they suffer from bounded recall. Naive solution with adaptive retrieval into these pipelines often leads to planning error propagation. To address this, we propose REPAIR, a framework that bridges this gap by repurposing reasoning plans as dense feedback signals for adaptive retrieval. Our key distinction is enabling mid-course correction during reranking through selective adaptive retrieval, retrieving documents that support the pivotal plan. Experimental results on reasoning-intensive retrieval and complex QA tasks demonstrate that our method outperforms existing baselines by 5.6%pt.[1]

1 Introduction

The central challenge of reasoning-intensive retrieval (Su et al., 2024) lies in the similarity gap between the user's query and the necessary evidence for its resolution. To illustrate, consider the query q: *"At home, after I water my plants, the water goes to plates below the pots. Can I reuse it for my plants next time?"* A similarity-based retriever would search for documents containing terms like "reuse plant water", while the gold document should "bridge" between the premise (collecting runoff water) to the conclusion (it is harmful to reuse). Such bridging documents are often not similar to the query and ranked low.

Existing approaches rely on *reasoning-based reranking* to surface these bridging documents higher in ranking. As illustrated in Figure 1(a), these rerankers generate a reasoning trace for a given query and prioritize documents that appear to support this trace. While effective at improving precision within a fixed candidate set, reasoning-based rerankers fundamentally suffer from the *bounded recall* problem: any relevant document missing from the initial retrieval pool $\mathcal{D}$ cannot be recovered by reranking alone.

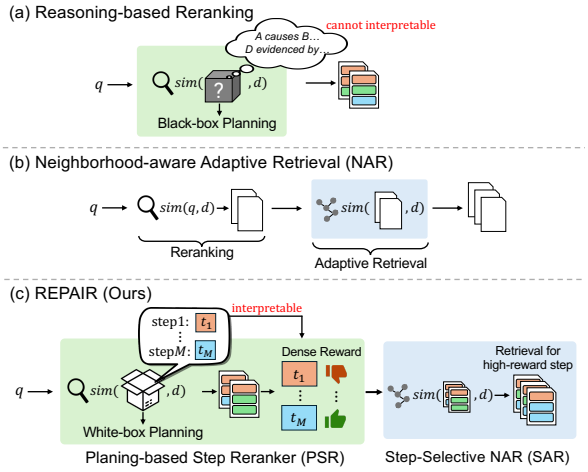

Figure 1: A comparison of reranking and adaptive retrieval paradigms. (a) Reasoning-based reranking first generates a reasoning trace and ranks documents based on the trace, but is limited by bounded recall since it operates on a fixed candidate set. (b) NAR addresses bounded recall by expanding candidates through document neighborhoods, but retrieval is guided solely by top-ranked documents, without awareness of reasoning steps. (c) REPAIR tightly integrates reranking and adaptive retrieval by transforming planning steps into dense, step-level rewards and selectively performing adaptive retrieval for mid-course correction.

Neighborhood-aware adaptive retrieval (NAR) (Rathee et al., 2025a,b) offers a principled solution to bounded recall by expanding candidates through document neighborhoods, as shown in Figure 1(b). A seemingly natural approach for reasoning-intensive retrieval is therefore to

[1]Source code: https://anonymous.4open.science/r/RIFT-BEB0/README.md

combine reasoning-based rerankers with NAR.

However, such a naive integration is insufficient and can even be detrimental, because effectively combining the two requires identifying which reasoning step should guide expansion—a challenge that neither component can address on its own. First, reasoning-based rerankers typically generate unstructured reasoning traces and remain a black box to NAR module. These traces make it difficult for the NAR process to interpret which reasoning steps are under-supported and should trigger expansion. Second, standard NAR relies solely on top-ranked documents. When these documents are biased toward an incorrect or weak reasoning direction, adaptive retrieval amplifies this bias by expanding their neighbors.

To address this gap, we propose **REPAIR** (**RE**asoning-aware **P**lan-**A**dapt**I**ve **R**etrieval), a framework that enables adaptive retrieval to leverage reasoning signals through an explicit interface. As illustrated in Figure 1(c), REPAIR exposes planning steps as interpretable signals, transforms them into dense, step-level rewards, and selectively performs adaptive retrieval for the most pivotal reasoning steps. Guided by these signals, REPAIR selectively performs adaptive retrieval for the most pivotal reasoning steps, enabling mid-course correction during retrieval.

Empirical results on reasoning-intensive retrieval and multi-hop QA benchmarks demonstrate that REPAIR consistently outperforms reasoning-based rerankers and naive extension methods, validating the importance of plan-adaptive retrieval.

Our contributions are threefold:

- We clarify the role of reranker and NAR in reasoning-intensive retrieval, and show that their naive combination is detrimental.
- We propose REPAIR, a plan-adaptive retrieval framework that exposes reasoning steps as interpretable signals to focus adaptive retrieval on pivotal reasoning steps, as dense step-level rewards.
- We demonstrate the effectiveness of REPAIR on reasoning-intensive retrieval and multi-hop question answering benchmarks, achieving consistent improvements over strong reranking and adaptive retrieval baselines by up to 5.6%pt.

2 Preliminaries

We formalize reasoning-intensive retrieval (§2.1) and review prior methods using reranking (§2.2) and adaptive retrieval (§2.3).

2.1 Reasoning-Intensive Retrieval in RAG

We consider reasoning-intensive retrieval under the RAG framework, where given a complex query q, a first-stage retriever (e.g., BM25) returns an initial candidate set $\mathcal{D}$. As motivated in Section 1, answering q in reasoning-intensive retrieval requires bridge documents that support intermediate reasoning steps, which are often only indirectly related to q and thus ranked low by the retriever. As a result, effective systems must enhance the precision and recall of the initial ranking, using reranking and NAR, respectively.

2.2 Improving Precision: Reranking

A listwise (or setwise) reranker ϕ is typically used to take a query q and a document list to output a ranked order by estimating collective relevance for reasoning. Figure 1(a) illustrates such reranking, commonly implemented by prompting an LLM with a ranking prompt p. Since the candidate set is typically large (e.g., top-100), reranking is performed over overlapping sliding windows $\mathcal{W}$:

$$\begin{aligned}\mathcal{W}' &= \phi\left(p, q, \mathcal{W}\right) \\ &= [d_{r_1} > \cdots > d_{r_{|\mathcal{W}|}}],\end{aligned} \tag{1}$$

where $\{r_1, \ldots, r_{|\mathcal{W}|}\}$ is a permutation indicating the estimated ranking.

While effective at promoting documents directly relevant to q, rerankers suffer from *bounded recall*: any relevant document missing from $\mathcal{D}$ cannot be recovered. Hence, we turn to adaptive retrieval for recall.

2.3 Improving Recall: NAR

Figure 1(b) illustrates how NAR, following reranking, addresses its bounded recall by expanding candidates using a pre-computed corpus graph G (Rathee et al., 2025a,b). After reranking a window $\mathcal{W}$, neighbors of the top-k documents are retrieved and merged into the next window:

$$\begin{aligned}\mathcal{W}' &= \phi\left(p, q, \mathcal{W}\right) \\ \mathcal{N} &= \text{Neighbors}(G, \mathcal{W}'[:k]) \\ \mathcal{W}_{\text{next}} &\leftarrow \mathcal{W}'[:k] \oplus \mathcal{N}[:b-k],\end{aligned} \tag{2}$$

where b is the window size and $\oplus$ denotes concatenation. In conventional prior works (Rathee

et al., 2025a), Neighbors retrieves and ranks neighbors using document–document similarity from G.[2] Under the Clustering Hypothesis (Jardine and van Rijsbergen, 1971), documents similar to highly ranked ones are likely relevant to the same query. By iteratively expanding the candidate set, NAR improves recall by recovering documents missed by the first-stage retriever.

2.4 Baseline: Reranking and NAR

A naive baseline to integrate them would be simply appending standard NAR to the reasoning rerankers as in Figure 1(b). While reasoning-based rerankers such as Rank-R1 (Zhuang et al., 2025) and Reason-Rank (Liu et al., 2025) internally perform planning-like reasoning, their unstructured reasoning traces are produced solely for ranking and cannot be used by NAR, as we later propose to reuse them for identifying under-supported steps.

This leads to a shared challenge in determining which reasoning step should guide expansion, as both the reranker and the adaptive retrieval component lack the necessary information to make this decision. From the reranking perspective, its reasoning process remains a black box, making it unclear which parts of the reasoning steps are under-supported and should guide expansion. Existing rerankers are optimized for final ranking rather than step-wise interpretability, so their intermediate traces are often not reliably structured. From the NAR perspective, NAR expands candidates based on top-ranked documents as shown in the second line in Eq. 2, without knowing which reasoning steps they support. Consequently, the two components operate largely independently, limiting the effectiveness of adaptive retrieval in reasoning-intensive settings. These limitations motivate a tighter integration between reranking and adaptive retrieval, where reasoning signals are made explicit and interpretable for retrieval.

3 Proposed: Dense PSR and SAR

We propose REPAIR combining PSR and SAR that plan, rerank, and adaptively retrieve for each window. Our key innovation is that the results from each window are leveraged as a dense reward signal for the next window. Based on the densified reward, REPAIR corrects high-ranked documents for each window, by selectively retrieving for the step with high reward. The overall algorithm is presented in Algorithm 1. We elaborate on these modules below.

[2]Neighbor expansion is typically applied every other window.

Algorithm 1 REPAIR

Input: The given query q, i-th window $\mathcal{W}_i$, window size b, the precomputed neighborhood graph G, and the predefined number of iterations I.

1: **for** $i \in [\lceil I/2 \rceil, I)$ **do**
2: ▷ Planning-based Step Reranker:
3: ▷ Planning
4: $\mathcal{R}_i = \{t_{i,1}, \ldots, t_{i,M}\}, \mathcal{R}_i \leftarrow \phi(p_{\text{plan}}, q, \mathcal{W}_i)$
5: ▷ Reranking
6: $\mathcal{W}'_i \leftarrow \phi(p_{\text{plan}}, q, \mathcal{R}, \mathcal{W}_i)$
7: ▷ Compute reward for each reasoning step $t_{i,\ell} \in \mathcal{R}_i$
8: $r_{i,\ell} \leftarrow f(t_{i,\ell}, \mathcal{W}'_i)$
9: ▷ Step-Adaptive NAR:
10: ▷ Take the step with the maximum reward
11: $t_{i,\star} \leftarrow \arg\max_{t_{i,\ell} \in \mathcal{R}_i} \; r_{i,\ell}$
12: ▷ Retrieve neighbors for the step $t_{i,\star}$
13: $\mathcal{N}_i \leftarrow \text{Neighbors}(G, \mathcal{W}'_i[:k])$
14: $\mathcal{W}_{i+1} \leftarrow \mathcal{W}'_i[:k] \oplus \{d \in \mathcal{N} \cup \mathcal{W}_{i+1} \mid \text{rank}_{\text{sim}}(d) \leq b - k\}$
15: **end for**
16: **return** $\mathcal{W}'_I$

3.1 Planning-based Step Reranker (PSR)

As shown in line 2-6 in Algorithm 1, PSR is a list-wise reranker that first generates a plan comprising reasoning steps to answer q. It then uses the plan as guidance to rerank $\mathcal{W}$, producing $\mathcal{W}'$.

Our goal is to build a deliberately simplified planning reranker interface that can better leverage the planning signal in prior work. Reasoning-based rerankers already emit planning as part of their reasoning (as illustrated in Appendix D.6), but these traces are discarded after ranking. Instead, we repurpose trace as the signal an adaptive retriever needs to decide where to expand.

To this end, we train a lightweight LLM-based reranker that takes $(p_{\text{plan}}, q, \mathcal{W})$ as input and outputs a re-ordered document list $\mathcal{W}'$ and plan $\mathcal{R}$, where p_{plan} is LLM prompt for planning (presented in Appendix B). We emphasize that this reranker is not intended to outperform prior planning-based rerankers in isolation. Rather, it reproduces their core behavior through a minimal interface that enables tight integration with adaptive retrieval.

Specifically, at each window, we formalize the interface as:

$$\begin{aligned}(\mathcal{W}', \mathcal{R}) &= (\mathcal{W}', \{t_1, \ldots, t_\ell, \ldots, t_M\}), \\ &= \phi(p_{\text{plan}}, q, \mathcal{W})\end{aligned} \tag{3}$$

where $\mathcal{W}$ denotes the window. We use $\mathcal{R}$ as an explicit planning interface, making intermediate

reasoning steps interpretable for subsequent modules.

For reranking, $\mathcal{R}$ replaces a single query q with a set of intermediate information requirements. This design closely follows existing reasoning-based rerankers, but is deliberately simplified to ensure that the planning signal is interpretable.

3.2 Step Selection Reward

While prior works naively expand documents for every individual step, our distinction is to compute which steps we should selectively target retrieval to maximize the answer coverage of q. The computation is shown in line 8 in Algorithm 1. Unlike *sparse* signals (V et al., 2025) that only emerge after all the windows are processed, we evaluate each step in $\mathcal{R}$ within the current window $\mathcal{W}'$ to guide selective expansion during the process. We use two dense reward functions, elaborated on below.

Baseline Reward Similar to prior work (V et al., 2025) that improves similarity for each sub-query, we compute a baseline reward r^{base} based on how much individual steps are similar to the top-ranked documents in the current window. Given t_ℓ, the reward can be formulated as:

$$r_\ell^{\text{base}} = -\frac{1}{|\mathcal{W}'|} \sum_{d \in \mathcal{W}'} \sigma\left(\text{sim}(t_\ell, d)\right), \quad (4)$$

where $\text{sim}(t_\ell, d)$ measures embedding similarity between the step description and a document, and σ denotes the sigmoid function for normalization. A higher r_ℓ^{base} indicates that a step t_ℓ has not yet been sufficiently covered by the top-ranked documents observed so far, suggesting remaining room for improvement.

However, not all reasoning steps contribute equally to answering the query (Prasad et al., 2023). A pivotal step may exist upon which other steps are dependent. Nevertheless, r^{base} measures step-level similarity gaps in isolation, over-prioritizing less important steps, while overlooking critical ones that substantially affect final ranking.

Consistency Reward Therefore, we introduce a consistency reward r^{con} that evaluates whether a step in the current window i consistently aligns with the reranker's preferences across previous window iterations $(1 \ldots i-1)$. This reward captures a longer-term notion of contribution by aggregating evidence from past ranking behaviors.

Specifically, we interpret the iterative reranking process as generating implicit pairwise preferences between documents over time. Rather than relying on single-window outcomes, we aggregate these preferences using the Bradley–Terry model (Bradley and Terry, 1952) to estimate a consensus preference distribution P:

$$P(d_p \succ d_q) = \frac{e^{\gamma_{pq}}}{e^{\gamma_{pq}} + e^{\gamma_{qp}}}, \quad (5)$$

where γ_{pq} denotes how frequently document d_p has been ranked above d_q across past iterations.

Let $t_{i,\ell}$ a step at i-th window and $\mathcal{W}'_{t_{i,\ell}}$ the document ranking produced using only step $t_{i,\ell}$. We define the set of all ordered document pairs $\mathcal{T}_{t_{i,\ell}}$ induced by this ranking as:

$$\mathcal{T}_{t_{i,\ell}} = \{(d_p, d_q) \mid d_p > d_q \text{ in } \mathcal{W}'_{t_{i,\ell}}\} \quad (6)$$

We then define the consistency reward for a step as the average alignment between this step-specific ranking $\mathcal{T}_{t_{i,\ell}}$ and the consensus model P:

$$r_{i,\ell}^{\text{con}} = \frac{1}{|\mathcal{T}_{t_{i,\ell}}|} \sum_{(d_p, d_q) \in \mathcal{T}_{t_{i,\ell}}} P(d_p \succ d_q), \quad (7)$$

A low $r_{i,\ell}^{\text{con}}$ indicates that the ranking induced by $t_{i,\ell}$ contradicts the established consensus from earlier windows, suggesting that $t_{i,\ell}$ may negatively contribute to the reasoning chain.

Finally, our goal is to identify a step that has maximum reward. We define the final reward $r_{i,\ell}$, for the step $t_{i,\ell}$, by combining two rewards[3]:

$$r_{i,\ell} = f(r_{i,\ell}^{\text{base}}, r_{i,\ell}^{\text{con}}) \quad (8)$$

We highlight that this reward formulation is computationally efficient. A standard reranker makes one LLM call per window to produce a single ranked list, yielding only one piece of feedback. In contrast, our approach extracts multiple signals from that same output. For a given window with M reasoning steps, we generate M distinct reward signals. By the i-th iteration, this accumulates $i \times M$ signals, forming a much denser feedback loop without incurring additional LLM cost.

3.3 Step-Adaptive NAR (SAR)

In contrast to standard NAR (§ 2.3), which indiscriminately retrieves neighbors based on the top-k

[3]For simplicity, we implemented f as linear combination $(r_{i,\ell}^{\text{base}} + r_{i,\ell}^{\text{con}})$.

documents from the current window, our SAR selectively constructs the next window to focus on the highest-reward step $t_{i,\star}$ using the generated reward.

With $r_{i,\ell}$, we compare each step's score and identify the step $t_{i,\star}$ that should be selected for the candidate expansion.

$$t_{i,\star} = \arg\max_{t_{i,\ell} \in \mathcal{R}_i} r_{i,\ell} \tag{9}$$

We then gather additional candidate documents from the neighbors of top-k documents in the graph G, following standard NAR. To enlarge the candidate pool for $t_{i,\star}$, we combine these neighbor documents with the next-window candidates $\mathcal{W}_{i+1}$ that the first-stage retriever would have retrieved. From this combined pool, we select the top-$(b-k)$ documents most relevant to $t_{i,\star}$:

$$\begin{aligned} \mathcal{N} =& \text{Neighbors}(G, \mathcal{W}'_i[:k]) \\ \mathcal{W}_{i+1} \leftarrow& \mathcal{W}'_i[:k] \oplus \\ & \{d \in \mathcal{N} \cup \mathcal{W}_{i+1} \mid \text{rank}_{\text{sim}}(d) \leq b-k\} \end{aligned} \tag{10}$$

The resulting $\mathcal{W}_{i+1}$ serves as the input window for the next iteration of PSR. To ensure a fair comparison with standard NAR (§ 2.3) in retrieval frequency, we apply adaptive retrieval only in the latter half of iterations. This scheduling also allows the preference model based on the accumulated reranking results (§3.2) to become sufficiently stable before guiding adaptive retrieval.

4 Experiments

4.1 Experimental Setting

Dataset We evaluate our method on both reasoning-intensive information retrieval (IR) multi-hop question answering (QA) tasks.

For information retrieval evaluation, we used BRIGHT (Su et al., 2024), which consists of 12 subsets. In BRIGHT, the gold document is often dissimilar to the given query, yet it is necessary to traverse the reasoning steps to answer the query (i.e., bridge documents). For example, in StackExchange subsets, the target document is defined by whether the document is cited in the reasoning to reach the answer.

For multi-hop QA benchmarks, we include HotpotQA (Yang et al., 2018), 2WIKI (Ho et al., 2020), and MusiQue (Trivedi et al., 2022), where answering a query requires composing multiple intermediate facts. We randomly sample 500 queries for each dataset due to the cost issue. We use nDCG@10 for the IR task and Exact Match (EM) and F1 scores for QA, following the common practice.

Baselines We compare REPAIR against the following baselines. BM25 (Robertson and Zaragoza, 2009) is a traditional lexical matching retriever widely used in IR. RankZephyr (Pradeep et al., 2023) is a GPT4-distilled listwise reranker. SUNAR$_R$ (V et al., 2025) is a NAR method that decomposes the query and applies reranker with NAR for each sub-query. Rank-R1 (Zhuang et al., 2025), REARANK (Zhang et al., 2025), and ReasonRank (Liu et al., 2025) are reasoning-based setwise and listwise rerankers, respectively, that perform intermediate reasoning before generating the ranked output. For rerankers, including ours, we apply reranking to the top 100 documents retrieved in the first stage.

Implementation Details See Appendix C.

4.2 Results and Analysis

We address the following research questions:

- **RQ1 (Performance)**: How does the REPAIR affect the performance?
- **RQ2 (Ablation study)**: Does each module in REPAIR contribute to performance?
- **RQ3 (Efficiency)**: How efficient is REPAIR in terms of computational cost?

4.2.1 How does the REPAIR affect the performance?

Effectiveness on Reasoning-Intensive Retrieval Table 1 compares our method on BRIGHT, against various conventional rerankers (RankZephyr, REARANK, Rank-R1, ReasonRank) and a NAR baseline (SUNAR$_R$) under a BM25 first-stage retrieval setting. The underperformance of SUNAR$_R$ compared to conventional rerankers highlights that a naive NAR approach can be detrimental. On the other hand, REPAIR consistently outperforms baselines. Notably, REPAIR achieves the highest average nDCG@10 gain of 10.2%pt (14.5 vs 24.7) from BM25 and of 5.6%pt (19.1 vs 24.7) from the best-performing baseline, demonstrating its effectiveness in reasoning-intensive retrieval. We note that REPAIR also outperforms ReasonRank, which is trained on reasoning-intensive data. However, since such training may introduce potential data contamination with the BRIGHT benchmark [4], we focus subsequent analyses on models trained with

[4]The analysis for the potential data contamination is described in Appendix D.5.

Model	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Pony	LC.	AoPS	TheoT.	TheoQ.	
Zero-shot													
BM25†	18.7	27.2	14.9	12.5	13.6	**18.4**	15.0	7.9	24.4	6.2	10.4	4.9	14.5
SUNAR$_R$	16.9	23.2	9.4	10.5	10.0	11.2	12.9	6.3	9.5	2.7	5.4	3.6	10.1
QwenRerank-7B	28.0	40.8	16.7	26.4	24.8	24.0	24.9	12.1	24.8	6.7	12.3	13.3	21.2
Training with knowledge-intensive data													
RankZephyr-7B	19.2	21.9	13.9	10.8	13.9	14.9	15.0	6.4	22.6	4.4	9.1	4.6	13.1
Rank-R1-7B	29.2	29.8	17.2	24.2	22.8	11.2	26.6	8.3	16.3	3.2	12.2	13.5	17.9
REARANK-7B	22.6	25.4	15.9	21.4	22.6	17.5	27.0	14.0	27.8	6.2	16.6	12.6	19.1
(Ours) REPAIR-7B	**39.6**	**38.1**	**21.4**	**32.6**	**24.6**	17.2	**33.1**	**15.6**	**29.0**	**7.2**	**16.9**	**21.2**	**24.7**
Training with reasoning-intensive data													
ReasonRank-7B	36.1	44.0	24.7	30.4	**31.1**	25.3	33.1	**17.7**	30.0	**11.5**	**18.7**	14.8	26.5
(Ours) REPAIR-7B	**43.1**	**45.0**	**32.4**	**32.5**	30.9	**26.3**	**36.5**	15.2	**31.4**	8.3	17.9	**19.8**	**28.3**

Table 1: nDCG@10 performance on BRIGHT. † indicates initial retrieval. All other models show reranking performance on the top-100 BM25 results. The best performing results are highlighted in bold.

Model	HotpotQA		2WIKI		MuSiQue	
	EM	F1	EM	F1	EM	F1
Contriever†	26.4	37.4	15.8	28.4	2.6	10.2
SUNAR$_R$	28.2	39.4	17.0	27.9	4.8	12.9
QwenRerank-7B	27.8	37.9	16.4	27.2	4.0	10.8
RankZephyr-7B	28.2	38.8	17.0	29.3	5.2	13.1
Rank-R1-7B	27.0	37.1	16.4	28.0	3.2	10.8
REARANK-7B	30.2	42.0	16.6	30.1	5.0	12.2
(Ours) REPAIR-7B	**32.2**	**43.0**	**19.4**	**31.8**	**7.6**	**15.4**

Table 2: Performance comparison on QA task. Each model returns the top-10 documents used as context for answer generation, with Llama-3.1-8B-Instruct serving as the answer generator. † indicates initial retrieval. All other models rerank the top-100 documents retrieved by Contriever. The **bold**-faced indiciates the best results.

Reranker	nDCG@10
Conventional Reranking	
REARANK	19.1
Conventional NAR	
SUNAR$_R$	10.1
SUNAR$_R$ (w/ our $\mathcal{R}$)	11.0
Planning-based Reranking	
(Ours) PSR	**20.2**

Table 3: Comparison between rerankers on BRIGHT.

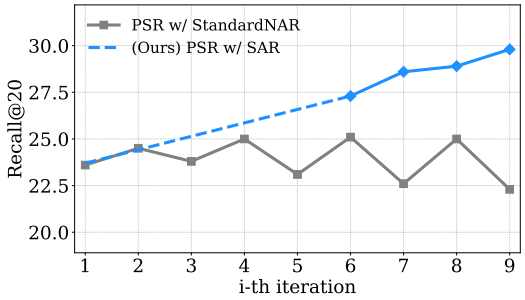

Figure 2: Comparison between retrievers on BRIGHT. $i = 1$ corresponds to the first-stage retrieval (BM25). For (Ours) PSR w/ SAR, we used BM25 ranking results up to $i < 6$ (See Figure 4 for related discussion.).

knowledge-intensive data to ensure a fair comparison.

Effectiveness on Multi-Hop Question Answering Beyond retrieval metrics, we demonstrate that the improved document sets provided by REPAIR also lead to substantial gains in downstream QA performance. As shown in Table 2, REPAIR consistently outperforms all baseline models across three datasets. Our method achieves the highest scores in both EM and F1 across the table. This strong performance underscores the importance of retrieving bridge documents to form a complete reasoning chain.

4.2.2 Does each module in REPAIR contribute to performance?

We validate the individual contributions of PSR and SAR, regarding improving precision of ranked lists and improving recall via retrieval, respectively, and finally the synergy between the two.

PSR as a representative planning baseline (Table 3). We validate REPAIR as a representative implementation of the planning paradigm by evaluating its precision against established models like REARANK. As the official implementation of SUNAR$_R$ uses a query decomposition as planning model, for a fair comparison, we also compare SUNAR$_R$ (w/ $\mathcal{R}$) using the same plan generated by

Reranker	nDCG@10	
	w/o NAR	w/ NAR
RankZephyr-7B	13.1	9.8 (-3.3)
REARANK-7B	19.1	13.3 (-5.8)
(Ours) REPAIR-7B	20.2	**24.7 (+4.5)**

Table 4: Naive baselines for evaluating the compatibility between rerankers and NAR on BRIGHT. Each row compares the same reranker with and without NAR. REPAIR w/o NAR corresponds to PSR.

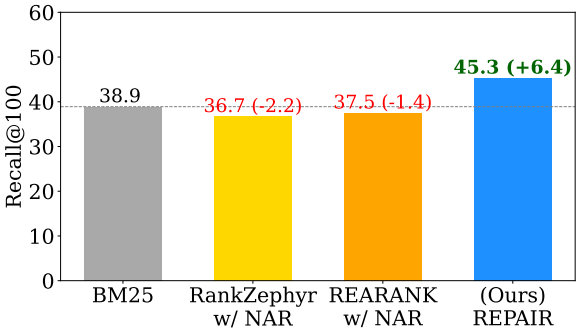

Figure 3: Compatibility comparison between different rerankers and NAR on BRIGHT benchmark.

PSR. We report average nDCG@10 on BRIGHT subsets, which measures precision on the top-10 ranked documents. Results are presented in Table 3.

REPAIR achieves precision comparable to the reasoning reranker, REARANK. This suggests that our interpretable planning implementation is a valid reproduction of the planning paradigm and yields a modest ranking gain. In contrast, the results that SUNAR$_R$ underperforms across settings indicate that incorporating naive NAR without step-aware rewards can be harmful. These results show that our approach of making planning steps explicit establishes a robust and interpretable interface, which supports the opportunity for a tighter integration with NAR.

SAR improves recall (Figure 2). We compare SAR with the standard NAR on the recall metric. Both employing expansion using a neighbor graph, SAR uses informed, selective expansion, whereas the standard NAR employs indiscriminate expansion. For a fair comparison, we used PSR as the reranker for both retrievers.

Figure 2 compares how the recall performance of the two retrievers evolves over iterations. $i = 1$ represents the performance of the first-stage retriever (BM25), which our model used up to $i < 6$. The performance of the standard NAR fluctuates, and it often underperforms BM25. We conjecture that the naive expansion often suffers from semantic shift, where the expanded set deviates from the original query (Rocchio Jr, 1971; Zighelnic and Kurland, 2008). In contrast, SAR shows consistent improvements across iterations, as it successfully excludes erroneous steps from the expansion and continues to uncover missing relevant documents.

Method	nDCG@10	Recall@100
REPAIR-7B	**24.7**	**45.3**
w/o r^{base}	24.3	44.8
w/o r^{con}	23.6	43.6
w/o r^{con} & r^{base}	20.2	38.9

Table 5: Ablation study of the reward components in REPAIR on BRIGHT.

Synergy between PSR and SAR (Table 4, Figure 3) Our key contribution is the effective integration of a reranker with NAR. To validate the synergy between the two, we compare REPAIR with naive integration of existing rerankers and NAR. Results on precision and recall metrics are presented in Table 4 and Figure 3, respectively.

Table 4 shows that simply applying a naive NAR to existing rerankers does not guarantee a performance boost. In fact, RankZephyr and REARANK see a significant drop in nDCG@10 when paired with NAR, by -3.3%pt and -5.8%pt, respectively. In contrast, REPAIR shows substantial gains when enhanced with our SAR module.

Figure 3 compares Recall@100 before and after applying NAR. While baselines combined with NAR see their recall decrease below the initial BM25 baseline (dashed gray line), our method is the only one to successfully improve recall.

These results demonstrate that conventional rerankers, which are not designed for adaptive retrieval, fail to provide a strong enough signal to guide the NAR process effectively. On the other hand, REPAIR is synergistic, where the reranker provides rewards that allow the NAR module to effectively retrieve for high-reward reasoning steps, leading to a significant improvement in both recall (Figure 3) and final ranking performance (Table 4).

Effectiveness of rewards (Table 5, Figure 4) We present an ablation study on reward functions, validating the contribution of base reward (r^{base}) and consistency reward (r^{con}). Table 5 shows

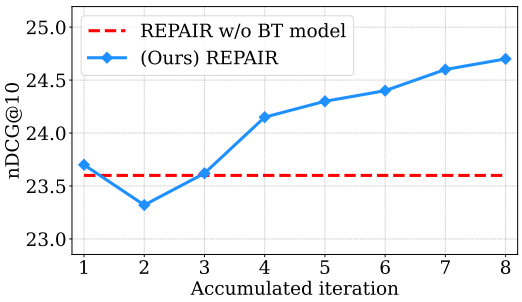

Figure 4: nDCG@10 vs. accumulated iterations used for the Bradley-Terry (BT) model on BRIGHT.

Model	nDCG@10	Avg. # Tokens (↓)
Rank-R1-7B	17.9	2622.0
REARANK-7B	19.1	7088.2
REPAIR-7B	**24.7**	<u>1803.2</u>
REPAIR-7B w/ reuse	<u>23.8</u>	**550.7**

Table 6: nDCG@10 and the average number of generated tokens for reranking on BRIGHT. 'w/ reuse' denotes the variant of REPAIR that reuses the same plan $\mathcal{R}$ for all windows. **Bold**-faced and <u>underlined</u> numbers denote the best and the second-best, respectively.

each reward is crucial for achieving optimal performance. Notably, removing both rewards (w/o r^{base} & r^{con}) results in the most significant performance drop, confirming that the two are synergistic.

Figure 4 provides a deeper analysis of r^{con}, illustrating how the final ranking performance (y-axis) changes as the Bradley-Terry (BT) preference model is built upon ranking feedback accumulated across iterations (x-axis). As more iterations are aggregated, performance steadily improves, surpassing the no-BT baseline (red-dashed line) around the 3–4 iterations. In practice, this recommends warming up or delaying the activation of r^{con} until adequate history is available, motivating us to use it after 5 iterations.

4.2.3 How efficient is REPAIR in terms of computational cost?

Token efficiency of PSR To evaluate the token efficiency of PSR, we compare the average number of generated tokens per query on the BRIGHT benchmark. The results in Table 6 highlight that PSR achieves higher nDCG while using about 4× fewer tokens than REARANK-7B and 30% fewer than Rank-R1-7B. The 'w/ reuse' variant further reduces token generation by over 69% with only a marginal performance drop.

Model	TFLOPS
PSR	188.0 (x1.00)
PSR + SAR (REPAIR)	191.6 (x1.02)

Table 7: FLOPS efficiency of SAR. We compare the computational cost (TFLOPS) of the single window, between the full framework (REPAIR) against PSR, demonstrating the low overhead of SAR.

FLOPS efficiency of SAR To evaluate the computational overhead of reaward computation and candidate expansion in SAR, we analyze the FLOPS efficiency in Table 7. For a single window, PSR requires 188.0 TFLOPS, while our full framework with SAR requires 191.6 TFLOPS. The 2% increase indicates that SAR 's cost is negligible compared to the LLM-based reranking pass, confirming the overall efficiency of our approach.

5 Conclusion

This paper addressed the limitations of naively combining reasoning-based reranking with neighborhood-aware adaptive retrieval in reasoning-intensive retrieval by introducing REPAIR. PSR provides an interpretable planning interface to enable tighter integration between reranking and adaptive retrieval, while the SAR uses a densified reward for targeted, mid-course correction. Our experiments show that REPAIR is an effective and efficient solution, successfully bringing bridge documents through selective adaptive retrieval in reasoning-intensive retrieval.

6 Limitation

We designed a simplified PSR to emit an interpretable plan instead of leveraging existing rerankers. Although many rerankers generate planning-like traces, their output formats and contents differ substantially, which makes direct plug-and-play integration non-trivial and potentially confounded by additional engineering choices. Developing general conversion or alignment techniques to make diverse reranker outputs compatible with SAR remains open.

Also, our framework relies on the quality of the planning steps exposed by the reranker, which are used as an interface to guide adaptive retrieval. Although the proposed step-level rewards help mitigate the impact of weak or misleading steps, errors in the initial planning can still affect retrieval decisions. An important direction for future work is to go beyond using a fixed plan per window and ex-

plore mechanisms for revising or refining planning steps during retrieval.

References

Ralph Allan Bradley and Milton E. Terry. 1952. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

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

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

N. Jardine and C.J. van Rijsbergen. 1971. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7(5):217–240.

Dayoon Ko, Jinyoung Kim, Hahyeon Choi, and Gunhee Kim. 2024. Growover: How can llms adapt to growing real-world knowledge? *arXiv preprint arXiv:2406.05606*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2025. Reasonrank: Empowering passage ranking with strong reasoning ability. *Preprint*, arXiv:2508.07050.

Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive re-ranking with a corpus graph. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1491–1500.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *arXiv preprint arXiv:2312.02724*.

Archiki Prasad, Swarnadeep Saha, Xiang Zhou, and Mohit Bansal. 2023. Receval: Evaluating reasoning chains via correctness and informativeness. *arXiv preprint arXiv:2304.10703*.

Xubo Qin, Jun Bai, Jiaqi Li, Zixia Jia, and Zilong Zheng. 2025. Reinforced query reasoners for reasoning-intensive retrieval tasks. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21250–21263, Suzhou, China. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, and 25 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025a. Guiding retrieval using llm-based listwise rankers. *Preprint*, arXiv:2501.09186.

Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025b. Quam: Adaptive retrieval through query affinity modelling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining*, pages 954–962.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Joseph John Rocchio Jr. 1971. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. Reasonir: Training retrievers for reasoning tasks. *arXiv preprint arXiv:2504.20595*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S Siegel, Michael Tang, and 1 others. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *CoRR*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Venktesh V, Mandeep Rathee, and Avishek Anand. 2025. Sunar: Semantic uncertainty based neighborhood aware retrieval for complex qa. In *North American Chapter of the Association for Computational Linguistics*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380.

Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. 2025. Rearank: Reasoning reranking agent via reinforcement learning. *Preprint*, arXiv:2505.20046.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning. *arXiv preprint arXiv:2503.06034*.

Liron Zighelnic and Oren Kurland. 2008. Query-drift prevention for robust query expansion. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 825–826.

Appendix

A Related Works

A.1 Reasoning-Intensive Retrieval

BRIGHT (Su et al., 2024) first introduced the reasoning-intensive retrieval, which necessitates in-depth reasoning to identify relevant documents that go beyond surface form matching. Prior works involve three directions: reranker, query-rewriter, and retriever. Reasoning-based rerankers such as Rank-R1 (Zhuang et al., 2025), ReasonRank (Liu et al., 2025), and REARANK (Zhang et al., 2025) use explicit reasoning, often with RL or chain-of-thought style supervision, to promote latent supporting evidence within a fixed candidate pool. Another line of work focuses on query rewriting such as TongSearch-QR (Qin et al., 2025), where the system rewrites the query with intermediate reasoning to increase retrieval recall upstream. Complementarily, reasoning-aware retrievers like ReasonIR (Shao et al., 2025) train the retriever itself to cover in-depth information needs. REPAIR is orthogonal to those approaches, by repurposing the reranker's reasoning plan as a dense feedback signal to drive adaptive retrieval.

A.2 Neighborhood-Adaptive Retrieval

The effectiveness of reranking systems is fundamentally constrained by the recall of the initial retrieval stage (MacAvaney et al., 2022; Rathee et al., 2025a). Relevant documents that are missing from the candidate set cannot be recovered later, which leads to a significant decline in overall performance. To address this, a line of research adopts Neighborhood Aware Retrieval (NAR) based on the Clustering Hypothesis (Jardine and van Rijsbergen, 1971). Conventional adaptive algorithms alternate between scoring first-stage results and exploring the corpus graph with reranked documents, either by pointwise (MacAvaney et al., 2022) or slidegar manner (Rathee et al., 2025a). A recent extension, SUNAR (V et al., 2025), incorporates query decomposition to support multi-hop retrieval. Our framework is distinguished by selecting the effective step for the adaptive retrieval.

B Prompt for PSR

System Prompt for PSR

You are an AI assistant that analyzes complex questions and identifies which documents best support answering them. Given a user's query and a set of documents, your task is to:
1. Generate a reasoning trace, thinking step by step about what knowledge or types of information are necessary to answer the query. These should be abstract but specific enough to guide document selection.
2. Select and rank at least 10 documents that best support the reasoning steps. Consider how each document contributes to the reasoning process. Order them from most to least useful using > between document IDs (e.g., `[3] > [7]`).

Use the following format:

[Reasoning Trace]
Step 1: <First reasoning step>
Step 2: <Second reasoning step>
...
Step N: <Final reasoning step>

[Document Ranking]
[9] > [5] > [6] > ... > [12]

Only produce the output in the format shown above.

User Prompt for PSR

[Query]
{query}

[Documents]
[1]: {document 1}
[2]: {document 2}
...
[num]: {document {num}}

C Implementation Details

To train PSR, we use Deepseek-r1 (Guo et al., 2025) for train data generation and train Qwen-2.5-7B-Instruct (Qwen et al., 2025). We used 72k of randomly sampled MS MARCO training set following Zhuang et al. (2025). We train the model using DeepSpeed ZeRO-3 with offloading (Rasley et al., 2020) and LoRA (Hu et al., 2022) adapter, on a cluster of 8 NVIDIA A6000 GPUs. We set the batch size to 32, the learning rate to 1e-5, and lora rank to 16. For comparison with ReasonRank, we additionally train REPAIR using the ReasonRank dataset following its training pipeline [5].

We used VLLM (Kwon et al., 2023) during inference. For inference, we follow the convention of previous adaptive rerankers (Rathee et al., 2025a; V et al., 2025) that use a left-to-right window, with the window size $b = 20$. In NAR, we set $k = 10$, meaning that neighbors are retrieved for the top-10 documents in each window, with each document retrieving 16 neighbor documents. Contriever (Izacard et al., 2022) [6] is used to build the corpus graph, measure $\mathrm{sim}(\cdot, \cdot)$ score, and approximate each step's rank in influence score. Single-run performance is reported.

Model	nDCG@10
7B-scale	
RankZephyr-7B	13.1
Rank-R1-7B	17.9
REARANK-7B	19.1
(Ours) REPAIR-7B	**24.7**
3B-scale	
Rank-R1-3B	14.0
(Ours) REPAIR-3B	**17.7**

Table 8: nDCG@10 performance on BRIGHT when scale down REPAIR.

D Further Analysis

D.1 Generalization to small-scale model

In addition to 7B model, we trained a smaller 3B parameter version of REPAIR and compared it against baselines. The results in Table 8 highlight that our approach is generalizable, with REPAIR-3B achieving an nDCG@10 of 17.7. This outperforms the same-sized baseline, Rank-R1-3B, and even the larger model, RankZephyr-7B, while remaining competitive with Rank-R1-7B and REARANK-7B.

D.2 Generalization to ReasonIR first-stage setting

To test REPAIR's generalizability, we replaced the BM25 first-stage retriever with ReasonIR (Shao

[5] https://github.com/8421BCD/reasonrank
[6] We used contriever-msmarco (Izacard et al., 2022).

et al., 2025), a retriever that is trained for reasoning tasks. Table 9 shows that REPAIR continues to deliver significant improvements on top of ReasonIR. Moreover, Table 10 highlights that this trend remains consistent when using an even more powerful first-stage retriever, ReasonIR with GPT-4 queries. These results confirm that REPAIR's effectiveness is not limited to improving weak initial retrievals but also generalizes to enhance the already powerful output.

D.3 Case study

Table 12 presents two representative case studies illustrating how step selection affects adaptive retrieval. For the query on mental fatigue ("What's the mechanism for being mentally tired?"), the Baseline (w/o rewards) selects a step focusing on categorizing types of fatigue, which shares surface-level similarity with the query but fails to retrieve the gold document. In contrast, REPAIR selects a step targeting the physiological mechanisms underlying mental fatigue, enabling retrieval of the correct document at rank 1. A similar pattern appears in the butterfly cocoon example. The Baseline prioritizes a step listing specific pupal structures (e.g., chrysalis or cocoon), whereas REPAIR focuses on explaining why the pupal stage is biologically necessary. By selecting this causal bridge step, REPAIR successfully retrieves the gold document at the top rank. In both cases, the selected steps are from the last window.

D.4 End-to-End Efficiency Comparison with SUNAR$_R$

In this section, we compare the end-to-end FLOPs required to process a single query for REPAIR and a representative NAR baseline, SUNAR$_R$. Following the SUNAR paper (V et al., 2025), we reproduce SUNAR$_R$ using Llama-3.1-8B-Instruct for query decomposition. For a fair comparison, both methods employ the same size of listwise reranker (RankZephyr-7B and PSR). Unlike Table 7, which reports FLOPs for a single reranking window, this analysis measures the total FLOPs accumulated over all windows required to complete one query. Under this setup, REPAIR processes 9 windows, resulting in a total cost of $191.6 \times 9 = 1724.4$ TFLOPs. SUNAR$_R$ incurs a total cost of 1893.6 TFLOPs per query under the same conditions, in part because it performs reranking separately for all decomposed sub-queries. These results show that REPAIR is slightly more efficient than SUNAR$_R$ in end-to-end computation, while delivering substantially stronger retrieval performance.

D.5 Analysis of potential data contamination

To assess the possibility of dataset overlap, we perform a two-sided contamination analysis between the training data used for ReasonRank and the BRIGHT test set, as shown in Table 13.

Query-level overlap: We examine semantic overlap between training queries and BRIGHT test queries. All queries are encoded using `princeton-nlp/sup-simcse-roberta-large`, and pairwise cosine similarities are computed. Following prior work on near-duplicate detection (Ko et al., 2024), we filter out BRIGHT test queries whose similarity to any ReasonRank training query exceeds 0.99.

Document-level overlap: The ReasonRank training corpus contains 148,107 unique documents. Among these, we find that 276 documents overlap with the BRIGHT gold documents. We then removed every BRIGHT test query whose gold document appears in this overlap set.

Table 13 reports the number of remaining queries relative to the total, suggesting that models trained on such data may benefit from prior exposure to evaluation documents. Based on this analysis, we focus on other baselines rather than ReasonRank to avoid confounding effects from potential data overlap and to ensure fair evaluation across methods trained on knowledge-intensive data.

D.6 Planning behavior in reasoning-based rerankers

Table 14 and Table 15 present representative outputs from reasoning-based rerankers. Across models, we observe a common pattern: before assessing the relevance of individual passages, the reranker first produces a planning-like reasoning trace that outlines intermediate considerations or subgoals for answering the query. This behavior indicates that planning is already an integral part of reasoning-based reranking.

However, these plans are typically generated in free-form text and serve only as intermediate scaffolding for ranking decisions. They are not exposed in a structured or interpretable form, nor are they designed to support downstream components such as adaptive retrieval. This observation motivates our design choice to reproduce this planning behavior through a simplified and explicit interface that can be interpreted and leveraged beyond reranking.

Model	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Pony	LC.	AoPS	TheoT.	TheoQ.	
ReasonIR†	26.3	31.5	23.3	30.3	18.0	23.5	20.6	10.3	35.0	14.3	31.6	27.2	24.3
SUNAR$_R$	7.8	25.2	6.6	7.4	3.7	7.1	8.0	32.0	12.3	5.8	5.8	6.7	10.7
QwenRerank-7B	31.9	40.7	20.3	31.2	18	23.4	24.6	18.6	27.1	6.5	23.9	33.1	24.2
RankZephyr-7B	25.8	31.5	22.3	30.0	16.4	21.0	20.6	10.1	33.0	13.1	29.9	26.4	23.3
Rank-R1-7B	39.5	25.7	22.8	33.4	18.6	19.6	29.6	9.3	21.4	7.6	26.7	36.6	24.2
REARANK-7B	28.8	25.4	24.7	31.5	19.4	18.8	28.5	13.1	33.8	10.5	29.7	30.5	24.6
REPAIR-7B	44.4	36.7	25.4	41.2	26.7	18.3	28.7	23.9	27.7	7.4	34.1	33.9	29.0

Table 9: nDCG@10 performance on BRIGHT. † indicates initial retrieval. All other models show reranking performance on the top-100 ReasonIR results.

Model	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Pony	LC.	AoPS	TheoT.	TheoQ.	
ReasonIR w/ GPT-4 query†	43.6	42.9	32.7	38.8	20.9	25.8	27.5	31.5	19.6	7.4	33.1	35.7	29.9
QwenRerank-7B	35.9	41.8	20.5	32.2	16.5	23.2	26.1	19.9	21.6	5.6	14.8	35.1	24.4
REPAIR-7B	53.8	45.6	34.4	45.9	28.1	27.9	41.3	24.6	24	7.7	38.6	44.2	34.7

Table 10: nDCG@10 performance on BRIGHT. † indicates initial retrieval. All other models show reranking performance on the top-100 ReasonIR results with GPT-4 query, but are not given the GPT-4 query during reranking. REPAIR is trained with the reasoning-intensive data.

Model	HotpotQA		2WIKI		MuSiQue	
	EM	F1	EM	F1	EM	F1
Qwen3-Embedding-0.6B†	27.6	38.5	18.6	30.2	3.0	9.9
QwenRerank-7B	26.8	36.3	16.0	23.8	3.4	10.0
RankZephyr-7B	29.2	41.0	18.8	30.5	5.2	12.1
Rank-R1-7B	32.2	42.5	18.0	30.3	5.2	12.3
REARANK-7B	32.4	43.2	18.2	29.4	5.6	13.0
(Ours) REPAIR-7B	**33.4**	**44.3**	**19.6**	**31.8**	**7.4**	**14.8**

Table 11: Performance comparison on QA task. Each model returns the top-10 documents used as context for answer generation, with Llama-3.1-8B-Instruct serving as the answer generator. † indicates initial retrieval. All other models rerank the top-100 documents retrieved by Qwen3-Embedding-0.6B. The **bold**-faced indiciates the best results.

E Use Or Create Scientific Artifacts

We used existing scientific artifacts for research purposes, and the use of existing artifacts was consistent with their intended applications.

Used Datasets The datasets we used are in English and are anonymized and protected from offensive content.

- BRIGHT benchmark (Su et al., 2024): CC-BY-4.0 license
- MS MARCO dataset (Izacard et al., 2021): non-commercial research purposes only
- HotpotQA dataset (Yang et al., 2018): CC BY-SA 4.0 license
- 2WIKI (Ho et al., 2020): Apache 2.0 license
- MusiQue (Trivedi et al., 2022): CC BY 4.0 license

Used Models

- Deepseek-r1 (Guo et al., 2025): MIT license
- Qwen-2.5-7B-Instruct (Qwen et al., 2025): Apache license 2.0
- contriever-msmarco (Izacard et al., 2021): CC BY-NC 4.0 license

Created Artifacts We release our model under CC-BY-NC-SA 4.0 license.

F Ai Assistants In Research Or Writing

We used ChatGPT for naming suggestions.

Method	Selected Step for Expansion	Gold Doc Rank
Query: *"What's the mechanism for being mentally tired? ... Is it neurotransmitters or lack of concentration?"*		
Baseline (w/o Rewards)	Step 5: *"Differentiate between acute mental fatigue... and chronic fatigue..."*	> 100 (Failed)
REPAIR	Step 2: *"Identify physiological mechanisms... such as neural activity..."*	**1**
Query: *"Why does the butterfly have a cocoon stage? ..."*		
Baseline (w/o Rewards)	Step 5: *"Highlight specific examples of pupal structures (e.g., chrysalis, cocoon)..."*	10
REPAIR	Step 2: *"Explain why the pupal stage is necessary in holometabolous insects..."*	**1**

Table 12: Case studies comparing the reasoning steps selected for adaptive retrieval. The Baseline (w/o rewards) selects steps with high surface-level similarity to the query, while REPAIR identifies under-supported bridge steps that capture the underlying causal or explanatory mechanism, leading to substantially improved document ranks.

Method	StackExchange							Coding		Theorem-based		
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Pony	LC.	AoPS	TheoT.	TheoQ.
query-level	41/103	100/116	70/103	101/101	87/101	116/117	82/108	112/112	142/142	101/111	194/194	76/76
document-level	103/103	116/116	103/103	101/101	101/101	117/117	108/108	112/112	11/142	17/111	171/194	41/76

Table 13: Analysis of data contamination. The numbers indicate the remaining queries over the total number of queries.

Query 1
Let r, s, and t be the three roots of the equation, $8x^3+1001x+2008=0$. Find $(r+s)^3+(s+t)^3+(t+r)^3$.
Rank-R1 Output 1
The query involves finding the value of $(r+s)^3+(s+t)^3+(t+r)^3$ given that r, s, and t are the roots of the equation $8x^3+1001x+2008=0$. This problem requires knowledge of polynomial roots and symmetric functions. None of the provided documents directly address this specific problem. However, document [10] involves a similar ...
Query 2
Claim in article about why insects are attracted to light In this article they are addressing the reason insects are attracted to light when they say (...) So that even though they donť encounter heat near/on the LEDs they still "expect" to?
Rank-R1 Output 2
The query is about why insects are attracted to light and not heat, specifically addressing the role of LEDs in this attraction. The most relevant document should discuss the relationship between light, heat, and insect behavior. Document 5 and 6-9 are not ...

Table 14: Examples of planning behavior in reasoning-based outputs (Rank-R1).

Query 1
Triangle ABC is inscribed in circle ω. Points P and Q are on side AB with $AP < AQ$. Rays CP and CQ meet ω again at S and T (other than C), respectively. If $AP = 4, PQ = 3, QB = 6, BT = 5$, and $AS = 7$, then $ST = \frac{m}{n}$, where m and n are relatively prime positive integers. Find $m + n$.
ReasonRank Output 1
Okay, let me try to figure out how to rank these passages for the given problem. The query is about a triangle inscribed in a circle with some points and intersections. The problem involves calculating ST given various lengths. First, I need to understand what the problem is asking. It's a geometry problem involving a circle, triangle ABC, points P and Q on AB, rays CP and CQ intersecting the circle again at S and T, and given specific segment lengths like AP=4, PQ=3, QB=6, BT=5, AS=7. The goal is to find ST as a fraction m/n and then m+n. Now, I need to rank the 20 passages based on relevance. The most relevant ones would be those that deal with similar geometry problems, especially involving circles, triangles, intersecting lines, power of a point, chords, or Ceva's theorem, Menelaus, etc. Looking at the passages: [1] Talks about a circle, chord lengths, power of a point. Might be relevant since the problem involves chords and circle properties. ...
Query 2
Why does hair turn grey or white, and why does it happen later for some? The question is pretty simple: what is happening molecularly when hair turns grey or white? I would imagine that it is due to the lack of a particular compound. I'm also interested in why some people's onset of grey hair happens much later than others.
ReasonRank Output 2
Okay, let me try to figure out how to rank these passages. The user wants to know why hair turns gray or white molecularly and why some people gray later. So I need to focus on passages that explain the biological mechanisms, like lack of a compound, melanocyte stem cells, oxidative stress, telomeres, etc. Passage [1] talks about melanocyte stem cells and Bcl2, which affects survival. That's relevant. Passage [2] discusses ...

Table 15: Examples of planning behavior in reasoning-based outputs (ReasonRank).