Plan-and-Rerank Framework for Reasoning-Intensive Retrieval

Anonymous ACL submission

Abstract

Reasoning-intensive retrieval necessitates recovering chains of interdependent evidence, where "bridge" documents that are not directly similar to the query are crucial. Prior approaches often fall short due to their isolated processing of reasoning steps and their lack of a mid-course correction mechanism. To overcome this, we propose RIFT, a plan-and-rerank framework with two key innovations: (1) We explicitly plan reasoning steps and rank documents based on their collective contribution to the entire reasoning plan, (2) We densify the reward signal from the reranker's output to provide mid-course correction by selectively retrieving documents for steps that improve the answer coverage. Experimental results on reasoning-intensive retrieval and complex QA tasks demonstrate that our method outperforms existing baselines by 5.9%pt.[1]

1 Introduction

The central challenge of reasoning-intensive retrieval (Su et al., 2024) lies in the semantic and lexical similarity gap between a 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", returning a top-ranked set $\mathcal{D}$ by surface similarity. However, the gold document is expected to provide the underlying scientific principle required to reason from the premise (collecting runoff water) to the conclusion (it is harmful to reuse). Therefore, the core challenge is identifying such "bridge documents" which may not be similar to the query, and thus are ranked low.

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

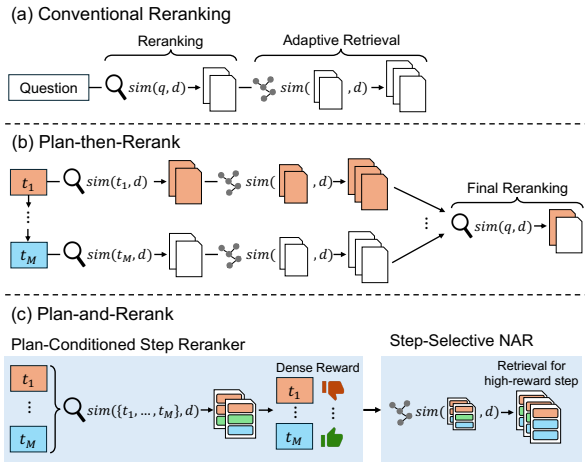

Figure 1: A comparison of adaptive reranking paradigms. (a) Adaptive reranking relying on the query fails to rank bridge documents. (b) Decomposing the query into sub-queries (or reasoning steps) cannot capture their collective coverage and often leaves some sub-queries covered only by unrelated documents. (c) In contrast, our approach maximizes coverage of reasoning steps and selectively performs adaptive retrieval with high-reward steps.

To surface the bridge in the ranking, conventional approaches consist of reranking and adaptive retrieval as a corpus feedback, as shown in Figure 1(a), capturing $\mathrm{sim}(q, d)$ and $\mathrm{sim}(d, d)$ respectively. Listwise rerankers (Sun et al., 2023; Pradeep et al., 2023) rank a more similar document d with q higher within $\mathcal{D}$. However, any relevant document absent from the initial candidate set $\mathcal{D}$ cannot be recovered by post-hoc rerankers, known as "bounded recall" problem. In another side is $\mathrm{sim}(d, d)$ to recover d missing in $\mathcal{D}$: Neighborhood-aware Adaptive Retrieval (NAR) methods (Rathee et al., 2025a,b) address this recall problem by leveraging document-document similarity in a pre-computed corpus graph. It retrieves the neighbors of top-ranked documents, assuming that similar documents tend to be relevant to the same query (Jardine and van Rijsbergen, 1971).

A naive extension of this pipeline is plan-*then*-rerank (V et al., 2025). As shown in Figure 1(b), it decomposes the query into sub-queries $\{t_{(\cdot)}\}$, and performs reranking and retrieving neighbors for each independently, based on $\mathrm{sim}(t_{(\cdot)}, d)$. However, this method has two limitations stemming from treating the sub-queries in isolation. (1) Two-phase Plan-then-Rerank: It fails to retrieve documents that *collectively* cover the complete reasoning chain, since each sub-query is processed independently. For example, during processing t_1 in Figure 1(b), it overlooks documents that could jointly answer multiple sub-queries. (2) Lack of mid-course correction: After reranking and NAR are performed independently for each sub-query, high-ranked results from each are forwarded to the final reranker.

To overcome these limitations, we propose **RIFT** (**R**easoning-**I**ntensive **F**eedback for re**T**rieval and reranking), a novel plan-*and*-rerank framework that combines Plan-Conditioned Step Reranker (PSR) and Step-Selective NAR (SAR). As shown in the left box of Figure 1(c), **PSR** first plans high-level reasoning steps required to answer the query and jointly reranks documents to maximize their collective coverage across all reasoning steps. Unlike plan-then-rerank, which treats each step in isolation, PSR prioritizes documents that jointly match multiple steps, denoted as multi-colored documents in Figure 1(c).

However, due to the bounded recall, some steps may still fail to rank relevant documents. To address this, we introduce **SAR**, which uses a reward-based mechanism for mid-course correction. As shown in the right panel of Figure 1(c), it leverages the reward computed from PSR's ranking results during the process. The reward for each reasoning step is to identify which one, if targeted for adaptive retrieval, would most effectively improve the overall answer coverage.

To contrast, see plan-then-rerank in Figure 1(b). If a sub-query t_M lacks sufficiently similar documents in the reranked result, NAR may also return unrelated documents, which are then carried into the final ranking without correction. In direct contrast, in Figure 1(c), we can identify that t_M is a key step but lacks supporting documents, assigning a higher reward than other reasoning steps (e.g., t_1) and promoting adaptive retrieval selectively for t_M during the process. The newly retrieved documents are then provided back into PSR, allowing the reranker to update its ranking candidates to cover all reasoning steps.

Our contributions are threefold:

- We identify fundamental limitations of the plan-then-rerank, where a naive integration of planning and NAR is often detrimental, underperforming standard rerankers without planning or even BM25.
- We introduce RIFT that actualizes the plan-and-rerank paradigm, featuring: (1) PSR that optimizes for collective reasoning, and (2) SAR that selectively perform adaptive retrieval with high-reward steps.
- We demonstrate the effectiveness of RIFT on reasoning-intensive retrieval and complex QA datasets, showing improvements over the existing baselines by 5.9%pt.

2 Preliminaries

This section formalizes reasoning-intensive retrieval (§ 2.1) and reviews existing approaches that leverage corpus feedback. We first discuss reranking (§ 2.2) and adaptive retrieval (§ 2.3), which aim to improve precision and recall, respectively. We then describe more recent work that integrates the two for reasoning-intensive retrieval (§ 2.4).

2.1 Reasoning-Intensive Retrieval in RAG

Let q be a complex question. A first-stage retriever (e.g., BM25) returns an initial candidate set $\mathcal{D}$. In reasoning-intensive retrieval, identifying bridge documents is essential, as they provide information needed in the reasoning process towards q. The challenge is that many bridge documents are only indirectly similar to q. As a result, a retrieval system must retrieve not just obviously similar documents, but also the hidden links.

2.2 Improving Precision: Listwise Reranker

A Listwise reranker ϕ takes the query q and a list of documents, and then outputs a ranked list based on similarity. It is done by instructing an LLM with the prompt p_{sim}. Since the input often contains a large number of documents (e.g., 100), reranking is applied in sliding window $\mathcal{W}$ (e.g., size 20 with overlap 10), iteratively covering the full set.

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

where $\{r_1, \ldots, r_{|\mathcal{W}|}\}$ denotes a permutation of $\{1, \ldots, |\mathcal{W}|\}$, representing the estimated similarity ranking. Recent reasoning-enhanced rerankers, e.g., Rank-R1 (Zhuang et al., 2025) and REARANK (Zhang et al., 2025), leverage reasoning abilities to estimate the similarity between q and d.

While this helps promote documents directly similar to q, these methods do not explicitly optimize for reasoning coverage across multiple steps. In addition, any relevant document absent from the initial set $\mathcal{D}$ cannot be recovered by post-hoc rerankers. Such a "bounded-recall" problem indicates that missing evidence in the first stage leads to irreversible failures in subsequent reranking.

2.3 Improving Recall: NAR

NAR addresses bounded recall by retrieving neighbors of the top-k documents from the current window $\mathcal{W}'$ using a pre-computed corpus graph G (Rathee et al., 2025a,b):

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

where b is the window size, and $\oplus$ denotes concatenation. Following prior work (Rathee et al., 2025a), Neighbors retrieves and sorts 16 neighbors for each document based on their $\text{sim}(d, d)$ scores obtained from G.[2] By the Clustering Hypothesis (Jardine and van Rijsbergen, 1971), documents similar to highly ranked ones tend to be relevant to the same query. Thus, iteratively fetching these candidates improves recall by pulling in related documents that the first-stage retriever missed.

2.4 Plan-then-Rerank

Recent work (V et al., 2025) decomposes q into sub-queries $(sq_1, sq_2, \ldots)$ and applies NAR-based reranking (Eq.2) to each sub-query. Formally, this can be formulated by replacing q with sq_i in Eq.2:

$$\begin{aligned} \mathcal{W}' &= \phi\left(p_{\text{sim}}, sq_i, \mathcal{W}\right) \\ \mathcal{N} &= \text{Neighbors}(G, \mathcal{W}'[:k]) \\ \mathcal{W}_{\text{next}} &\leftarrow \mathcal{W}'[:k] \oplus \mathcal{N}[:b-k], \end{aligned} \tag{3}$$

where sq_i denotes i-th sub-query. After reranking and NAR to all sub-queries using Eq.3, the top-ranked documents from each sub-query's results are aggregated to cover the reasoning steps. A final ranked list is then obtained by reranking the aggregated documents $\mathcal{D}_{\text{agg}}$ with respect to the original query:

$$\mathcal{D}'_{\text{agg}} = \phi\left(p_{\text{sim}}, q, \mathcal{D}_{\text{agg}}\right) \tag{4}$$

However, treating each sub-query in isolation exacerbates the inherent limitations of the two modules. (1) They often assign high ranks to documents similar only to individual sub-queries, overlooking the holistic reasoning steps. (2) High-ranked documents from each are indiscriminately forwarded to the final reranker without correction. Consequently, plan-then-rerank often fails to capture documents that address the reasoning process as a whole.

[2] Standard NAR methods typically perform this neighbor expansion on every second (even-numbered) window.

Algorithm 1 RIFT

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.

for $i \in [\lceil I/2 \rceil, I)$ **do**
  ▷ Plan-Conditioned Step Reranker:
  ▷ Planning
  $\mathcal{R}_i = \{t_{i,1}, \ldots, t_{i,M}\}, \mathcal{R}_i \leftarrow \phi(p_{\text{plan}}, q, \mathcal{W}_i)$
  ▷ Reranking
  $\mathcal{W}'_i \leftarrow \phi(p_{\text{plan}}, q, \mathcal{R}, \mathcal{W}_i)$
  ▷ Compute reward for each reasoning step $t_{i,\ell} \in \mathcal{R}_i$
  $r_{i,\ell} \leftarrow f(t_{i,\ell}, \mathcal{W}'_i)$
  ▷ Step-Selective NAR:
  ▷ Take the step with the maximum reward
  $t_{i,\star} \leftarrow \arg\max_{t_{i,\ell} \in \mathcal{R}_i} r_{i,\ell}$
  ▷ Retrieve neighbors for the step $t_{i,\star}$
  $\mathcal{N}_i \leftarrow \text{Neighbors}(G, \mathcal{W}'_i[:k])$
  $\mathcal{W}_{i+1} \leftarrow \mathcal{W}'_i[:k] \oplus \{d \in \mathcal{N} \cup \mathcal{W}_{i+1} \mid \text{rank}_{\text{sim}}(d) \leq b-k\}\}$
end for
return $\mathcal{W}'_I$

3 Proposed: Dense Plan-and-Rerank

We propose RIFT combining PSR and SAR that plan, rerank, and adaptively retrieve for each window. RIFT introduce two key innovations. First, to address the limitations of prior works, PSR generates the reasoning plan and jointly reranks the documents based on their collective contribution to the entire reasoning plan. Our second innovation lies in our *dense* use of the sliding window, which deviates from conventional listwise rerankers that use a sliding window just to deliver a single top-ranked document order to the next window. In our framework, the results from each window are leveraged as a dense reward signal for the next window. Based on the densified reward, RIFT 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.

3.1 Plan-Conditioned 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 $\mathcal{W}'$.

Specifically, at each window, we obtain a sequence of reasoning steps $\mathcal{R}_i$ using an LLM prompt p_{plan} (presented in Appendix A):

$$\begin{aligned}\mathcal{R}_i &= \phi(p_{\text{plan}}, q, \mathcal{W}) \\ &= \{t_1, \ldots, t_\ell, \ldots, t_M\},\end{aligned} \tag{5}$$

where $\mathcal{W}$ denotes the window. For reranking $\mathcal{W}$, $\mathcal{R}$ serves as the multi-criterion pivot, replacing a single query q with a set of intermediate information requirements that collectively define the relevance criteria.

Given $\mathcal{R}$, the core objective of PSR is to improve the answer coverage for q by ranking documents based on their collective contribution to the holistic reasoning chain $\mathcal{R}$. This distinguishes PSR from baseline rerankers that merely capture similarity to the given q or individual sub-queries, missing bridge documents not directly similar to q or only partially relevant to any individual sub-queries (Jedidi et al., 2025).

To this end, we train our LLM-based reranker to take $(p_{\text{plan}}, q, \mathcal{R}, \mathcal{W})$ as input and output a reordered list of documents $\mathcal{W}'$ that contains the essential information needed to answer the given query.

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 that only emerge after each step is processed, we evaluate each step in $\mathcal{R}$ within the current window $\mathcal{W}'$ to guide selective expansion during the process. We use three dense reward functions, elaborated on below.

Baseline Reward Similar to prior work (V et al., 2025) that improve 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:

$$r_\ell^{\text{base}} = -\frac{1}{|\mathcal{W}'|} \sum_{d \in \mathcal{W}'} \sigma\left(\text{sim}(t_\ell, d)\right), \tag{6}$$

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 coverage gaps in isolation, over-prioritizing less important steps, while overlooking critical ones that substantially affect final answer coverage.

To overcome this limitation, our contribution is introducing two complementary rewards—**influence** and **consistency**—that leverage feedback signals from the reranker to infer each step's influence based on dependency within-window and consistency across-windows.

Influence Reward While r^{base} captures whether a step is covered, it does not reveal how much that step actually influenced the reranker's decision within the current window. To quantify a step's contribution to the ongoing ranking process, we borrow the inspiration from previous findings (Hammoudeh and Lowd, 2024), which suggests that a subset can be considered influential if it is representative of the whole. We define the influence reward r^{inf}, which we compare rankings with all reasoning steps and with one step isolated, where the influence of the step is high if two are similar. Conversely, a low alignment implies that the step is less influential or potentially distracting.

Formally, we compute the normalized ranking correlation (nDCG) between $\mathcal{W}'_{t_\ell}$ (ranking by t_ℓ alone) and $\mathcal{W}'$ (ranking using all steps):

$$r_\ell^{\text{inf}} = \text{nDCG}(\mathcal{W}'_{t_\ell}, \mathcal{W}'), \tag{7}$$

where $\mathcal{W}'_{t_\ell}$ denotes the ranked list obtained using only step t_ℓ. In practice, we approximate this rank efficiently using a compact bi-encoder.

Consistency Reward Even if a step appears high contribution in the current window, its contribution may be transient or context-specific for the current window. 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:

$$P(d_p \succ d_q) = \frac{e^{\gamma_{pq}}}{e^{\gamma_{pq}} + e^{\gamma_{qp}}}, \tag{8}$$

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:

$$\mathcal{T}_{t_{i,\ell}} = \{(d_p, d_q) \mid d_p > d_q \text{ in } \mathcal{W}'_{t_{i,\ell}}\} \tag{9}$$

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:

$$r^{\text{con}}_{i,\ell} = \frac{1}{|\mathcal{T}_{t_{i,\ell}}|} \sum_{(d_p,d_q)\in\mathcal{T}_{t_{i,\ell}}} P(d_p \succ d_q), \tag{10}$$

A low $r^{\text{con}}_{i,\ell}$ 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 three rewards[3]:

$$r_{i,\ell} = f(r^{\text{base}}_{i,\ell}, r^{\text{inf}}_{i,\ell}, r^{\text{con}}_{i,\ell}) \tag{11}$$

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

3.3 Step-Selective NAR (SAR)

In contrast to standard NAR (Eq.2), which indiscriminately retrieves neighbors based on the top-k 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.

$$t_{i,\star} = \underset{t_{i,\ell}\in\mathcal{R}_i}{\arg\max}\ r_{i,\ell} \tag{12}$$

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 $t_{i,\star}$:

$$\begin{aligned} \mathcal{N} =& \text{Neighbors}(G, \mathcal{W}'_i[:k]) \\ \mathcal{W}_{i+1} \leftarrow& \mathcal{W}'_i[:k] \oplus \\ & \{d \in \mathcal{N} \cup \mathcal{W}_{i+1} \mid \text{rank}_{\text{sim}}(d) \le b-k\} \end{aligned} \tag{13}$$

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 schedule 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 IR 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

Model	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Pony	LC.	AoPS	TheoT.	TheoQ.	
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
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) RIFT-7B	**41.9**	**37.5**	**22.7**	**34.4**	**23.6**	16.3	**30.2**	**17.8**	**30.2**	**6.3**	**16.9**	**22.4**	**25.0**

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
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) RIFT-7B	**30.6**	**42.7**	**19.0**	**32.9**	**8.4**	**15.6**

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.

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 RIFT 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 plan-then-rerank method that decomposes the query and applies reranker with NAR for each sub-query. Rank-R1 (Zhuang et al., 2025) and REARANK (Zhang et al., 2025) are reasoning-enhanced 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 B.

4.2 Results and Analysis

We address the following research questions:

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

4.2.1 How does the RIFT affect the performance?

Effectiveness on Reasoning-Intensive Retrieval Table 1 compares our method on BRIGHT, against various conventional rerankers (RankZephyr, REARANK, Rank-R1) and a plan-then-rerank baseline (SUNAR$_R$) under a BM25 first-stage retrieval setting. The underperformance of SUNAR$_R$ compared to conventional rerankers highlights that a naive "plan-then-rerank" approach can be detrimental. On the other hand, RIFT consistently outperforms baselines. Notably, RIFT achieves the highest average nDCG@10 gain of 10.5%pt (14.5 vs 25.0) from BM25 and of 5.9%pt (19.1 vs 25.0) from the best-performing baseline, demonstrating its effectiveness in reasoning-intensive retrieval.

Effectiveness on Multi-Hop Question Answering Beyond retrieval metrics, we demonstrate that the improved document sets provided by RIFT also lead to substantial gains in downstream QA performance. As shown in Table 2, RIFT 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 RIFT contribute to performance?

We validate the individual contributions of PSR and SAR, regarding improving precision of ranked

Reranker	nDCG@10
w/o planning	
REARANK	19.1
w/ planning	
SUNAR$_R$	10.1
SUNAR$_R$ (w/ our $\mathcal{R}$)	11.0
(Ours) PSR	**20.2**

Table 3: Comparison between rerankers on BRIGHT.

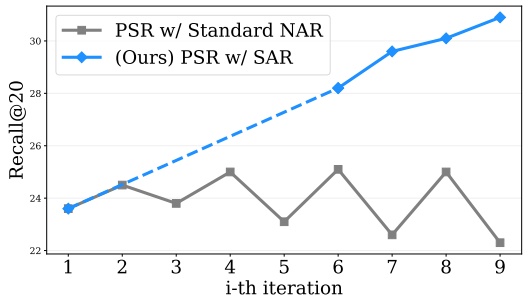

Figure 2: Comparison between retrievers on BRIGHT. $i = 1$ corresponds to the first-stage retrieval (BM25). For (Ours) PSR w/ SAR, we used BM25 ranking results up to $i < 6$ (See Figure 4 for related discussion.).

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) RIFT-7B	20.2	**25.0 (+4.8)**

Table 4: Compatibility between rerankers and NAR on BRIGHT. Each row compares the same reranker with and without NAR. RIFT w/o NAR corresponds to PSR.

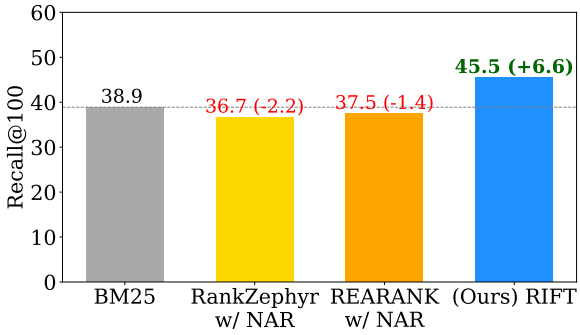

Figure 3: Compatibility comparison between different rerankers and NAR on BRIGHT benchmark.

lists and improving recall via retrieval, respectively, and finally the synergy between the two.

PSR improves precision (Table 3). We validate the effectiveness of our PSR (ablating SAR from RIFT) in reranking by comparing it with a standard reranker (REARANK) and a plan-then-rerank model (SUNAR$_R$). As the official implementation of SUNAR$_R$ uses a different planning model, for a fair comparison, we also compare SUNAR$_R$ (w/ $\mathcal{R}$) using the same plan generated by PSR. We report average nDCG@10 on BRIGHT subsets, which measures precision on the top-10 ranked documents. Results are presented in Table 3.

PSR outperforms REARANK, demonstrating the effectiveness of our plan-and-rerank approach. In contrast, SUNAR$_R$'s underperformance suggests that a naive plan-then-rerank approach without mid-course correction can be detrimental. PSR, however, leverages the plan as a multi-criterion pivot for collective scoring, similar to an ensemble of multiple queries, enhancing robustness to occasional errors in individual reasoning steps.

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.

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 RIFT 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, RIFT shows substantial gains when enhanced with our SAR module.

Figure 3 compares Recall@100 before and after applying NAR. While baselines combined with

Method	nDCG@10	Recall@100
RIFT-7B	**25.0**	**45.5**
w/o r^{inf}	24.7	45.3
w/o r^{con}	24.4	44.8
w/o r^{inf} & r^{con}	23.6	43.6

Table 5: Ablation study of the reward components in S^2NAR on BRIGHT.

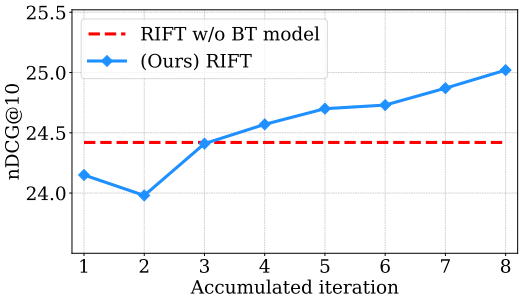

Figure 4: nDCG@10 vs. accumulated iterations used for the Bradley-Terry (BT) model on BRIGHT.

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, RIFT 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 influence reward (r^{inf}) and consistency reward (r^{con}). Table 5 shows each reward is crucial for achieving optimal performance. Notably, removing both rewards (w/o r^{inf} & 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.

Model	nDCG@10	Avg. # Tokens (↓)
Rank-R1-7B	17.9	2622.0
REARANK-7B	19.1	7088.2
RIFT-7B	**25.0**	<u>1804.7</u>
RIFT-7B w/ reuse	<u>24.6</u>	**555.0**

Table 6: nDCG@10 and the average number of generated tokens for reranking on BRIGHT. 'w/ reuse' denotes the variant of RIFT 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.

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

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

4.2.3 How efficient is RIFT 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 70% with only a marginal performance drop.

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 sparse plan-then-rerank methods in reasoning-intensive retrieval by introducing RIFT. PSR ensures holistic evidence coverage, while the SAR uses a densified reward for targeted, mid-course correction. Our experiments show that RIFT is an effective and efficient solution, validating the plan-and-rerank approach and the value of dense feedback in reasoning-intensive retrieval.

6 Limitation

Our framework's performance, like other planning-based approaches, is influenced by the quality of the initial reasoning plan generated by the LLM. While our selectively adaptive retrieval is designed to avoid selecting low-quality steps, a promising direction for future work is to enable dynamic plan revision.

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*.

Zayd Hammoudeh and Daniel Lowd. 2024. Training data influence analysis and estimation: A survey. *Machine Learning*, 113(5):2351–2403.

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.

Nour Jedidi, Yung-Sung Chuang, James Glass, and Jimmy Lin. 2025. Don't" overthink" passage reranking: Is reasoning truly necessary? *arXiv preprint arXiv:2505.16886*.

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.

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*.

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*.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. *arXiv preprint arXiv:2304.09542*.

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 re-ranking 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 Prompt for PSR

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.
```

B 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,

Model	nDCG@10
7B-scale	
RankZephyr-7B	13.1
Rank-R1-7B	17.9
REARANK-7B	19.1
(Ours) RIFT-7B	**25.0**
3B-scale	
Rank-R1-3B	14.0
(Ours) RIFT-3B	**17.7**

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

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.

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) [4] 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.

C Further Analysis

C.1 Generalization to small-scale model

In addition to 7B model, we trained a smaller 3B parameter version of RIFT and compared it against baselines. The results in Table 8 highlight that our approach is generalizable, with RIFT-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.

C.2 Generalization to ReasonIR first-stage setting

To test RIFT's generalizability, we replaced the BM25 first-stage retriever with ReasonIR (Shao et al., 2025), a retriever that is trained for reasoning tasks. Table 9 shows that RIFT continues to deliver significant improvements on top of ReasonIR. This result confirms that RIFT's effectiveness is not limited to improving weak initial retrievals but also generalizes to enhance the already powerful output.

[4] We used contriever-msmarco (Izacard et al., 2022).

D 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.

E Ai Assistants In Research Or Writing

We used ChatGPT for naming suggestions.

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
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
RIFT-7B	**45.2**	**38.1**	**25.3**	**37.8**	**25.7**	15.1	**29.9**	24.9	25.6	7.0	**34.9**	**38.6**	**29.0**

Table 9: nDCG@10 performance on BRIGHT. † indicates initial retrieval. All other models show reranking performance on the top-100 ReasonIR results. The best performing results are highlighted in bold.