Group-Aware Adaptive Retrieval for Evidence Navigation

June Park*, **Jun Kwon***, **Jonghyo Kim**, **Jongwuk Lee**†
Sungkyunkwan University, Republic of Korea
{pj00515, kwon04210, naye971012, jongwuklee}@skku.edu

Abstract

Reasoning-intensive retrieval addresses queries whose relevance cannot be identified by surface-level matching, thereby requiring multi-step reasoning. Because relevant documents rarely appear in the initial candidate set, retrieval systems suffer from the *bounded recall problem*. Existing methods iteratively expand a candidate pool at the document level over a corpus graph, examining each neighbor in isolation and drifting toward a narrow region of the corpus. To address this problem, we propose ***G**roup-Aware **A**daptive **R**etrieval for **E**vidence **N**avigation (**GAREN**)*, which explores the corpus graph through group-level expansion. GAREN organizes documents into semantically coherent and distinguishable groups based on their connections in the corpus graph. The information in each group indicates what can be accessed by expanding through it, providing guidance beyond individual document-level signals. At each iteration, GAREN uses a group-level navigator to identify promising expansion directions, retrieves documents from the selected groups, and applies a document-level reranker to evaluate the updated candidate set. Extensive experiments show that GAREN achieves up to 8.0% improvement over the strongest baseline on BRIGHT. The source code is available at https://github.com/KJ12124/GAREN

1 Introduction

Reasoning-intensive retrieval (Su et al., 2025) targets queries where relevance requires reasoning beyond lexical or semantic similarity. This indirect relevance makes relevant documents difficult to retrieve initially. Although stronger retrievers (Shao et al., 2025) or query reformulation (Zhong et al., 2025) may improve recall, a fixed initial candidate set still struggles to capture documents whose relevance emerges only through multi-step reasoning.

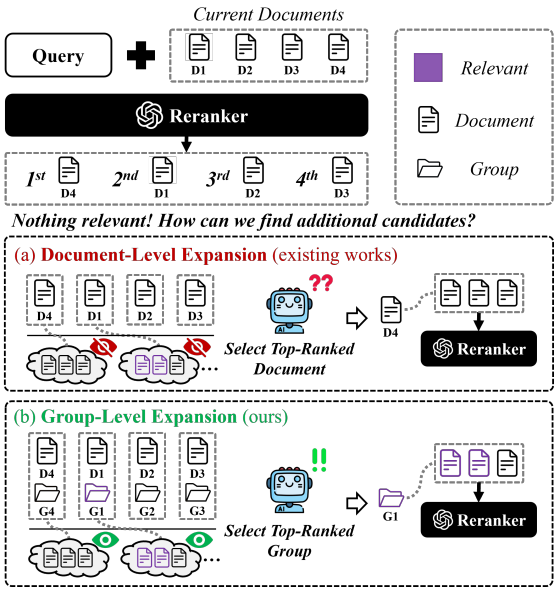

Figure 1: Illustration of our motivation. While (a) existing methods expand based on individual documents, (b) GAREN evaluates groups that reveal the documents accessible through each expansion direction.

As a result, relevant documents missing from this initial candidate set are difficult to recover downstream, leaving the *bounded recall problem* as a persistent challenge (Rathee et al., 2025a).

Recently, Adaptive Retrieval (MacAvaney et al., 2022) has emerged as a promising solution by adaptively expanding the candidate set. Existing methods (Rathee et al., 2025a; Xu and Chen, 2025) leverage a corpus graph that connects similar documents, following the *clustering hypothesis* (Jardine and van Rijsbergen, 1971), and expand the neighbors of documents assessed as relevant. At each step, the relevance of documents in the candidate set serves as evidence for deciding what to expand, enabling the expansion of additional relevant documents.

However, a document may not appear directly relevant to the query, even if expanding from it can lead to relevant documents (Kim et al., 2026). As shown in Figure 1(a), when none of the current documents appear clearly relevant, document-

* Equal contribution.
† Corresponding author

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 48728–48747
October 24-29, 2026 ©2026 Association for Computational Linguistics

level relevance signals provide little indication of which expansion direction is likely to lead toward relevant documents. This limitation is especially critical in early retrieval stages, where the current candidate set provides only weak or unreliable evidence. In such cases, early retrieval errors can propagate through the iterative expansion process, as documents retrieved along an incorrect direction introduce misleading signals for subsequent expansion decisions (Kim et al., 2026).

This motivates our central research question: *How can expansion directions toward relevant documents be selected from weak and indirect signals?* To address this question, we propose ***G**roup-Aware **A**daptive **R**etrieval for **E**vidence **N**avigation* (**GAREN**), a framework that evaluates expansion directions beyond individual document-level relevance. For each expansion direction, it treats the documents that would be introduced by expanding in that direction as a *group*. As illustrated in Figure 1(b), the group-level context indicates what additional information can be obtained by expanding in that direction, allowing the framework to assess whether the direction is worth exploring. As a result, GAREN can more reliably identify expansion directions that are likely to lead toward relevant documents, even when the current retrieval state provides only limited evidence.

Specifically, GAREN consists of three steps. (i) *Group Construction* organizes densely connected documents into semantically coherent and distinguishable groups, enabling more informed comparison of potential expansion directions. It then summarizes the shared semantic context of each group into a representation that provides a coarse preview of the information reachable through that direction. (ii) *Group-Aware Adaptive Retrieval* then combines a document-level reranker and a group-level navigator: the reranker assesses the relevance of current candidate documents, while the navigator selects groups from which to expand next. Documents from the selected groups are added to the candidate set, which becomes the retrieval state for the next iteration. Within this process, we adopt an *explore-then-exploit* strategy: the framework initially explores multiple groups to reduce uncertainty under limited evidence, and gradually focuses on more promising groups as reliable signals accumulate, reducing error propagation. (iii) Following iterative retrieval, *Group-Driven Evidence Propagation* refines candidate scores by allowing high-confidence documents to provide supporting evidence for other documents within the same group.

We summarize our key contributions as follows.

- We introduce *group* as a coarse-grained unit for expansion decisions in the corpus graph, allowing retrieval to assess where an expansion may lead.
- We propose GAREN, a group-aware adaptive retrieval framework that couples a coarse-grained navigator with a fine-grained reranker.
- We show that GAREN achieves strong performance on BRIGHT and remains effective on additional retrieval datasets.

2 Related Work

2.1 Reasoning-Intensive Retrieval

Recent reasoning-intensive retrieval benchmarks have highlighted scenarios in which relevant documents cannot be identified through lexical overlap or semantic similarity (Su et al., 2025). To address this challenge, prior work has incorporated reasoning into different stages of the retrieval pipeline. One line of work enhances query interpretation using LLMs to decompose complex queries, generate reasoning chains, or rewrite queries with corpus feedback (Zhong et al., 2025; Lei et al., 2025). Another line trains retrievers with reasoning-oriented supervision, encouraging retrieval models to capture indirect relevance (Shao et al., 2025; Das et al., 2025). Others enrich documents offline with hypothetical information needs or query scenarios they can address, making implicit relevance explicit at indexing time (Lee et al., 2025; Chen et al., 2025). Complementary to these approaches, reranking methods perform reasoning over candidate documents to assess their relevance to the query (Liu et al., 2025). These approaches are orthogonal to ours: query formulation, document enrichment, and stronger retrievers improve the initial retrieval state, while adaptive expansion recovers documents beyond the initial set.

2.2 Adaptive Retrieval

Adaptive retrieval mitigates the *bounded recall problem* by dynamically introducing additional candidates beyond the initial retrieval results. Grounded in the clustering hypothesis, early graph-based methods construct document graphs and expand candidates through reranker-guided exploration (MacAvaney et al., 2022; Kulkarni et al., 2023). Subsequent methods further refine graph traversal by incorporating additional signals such

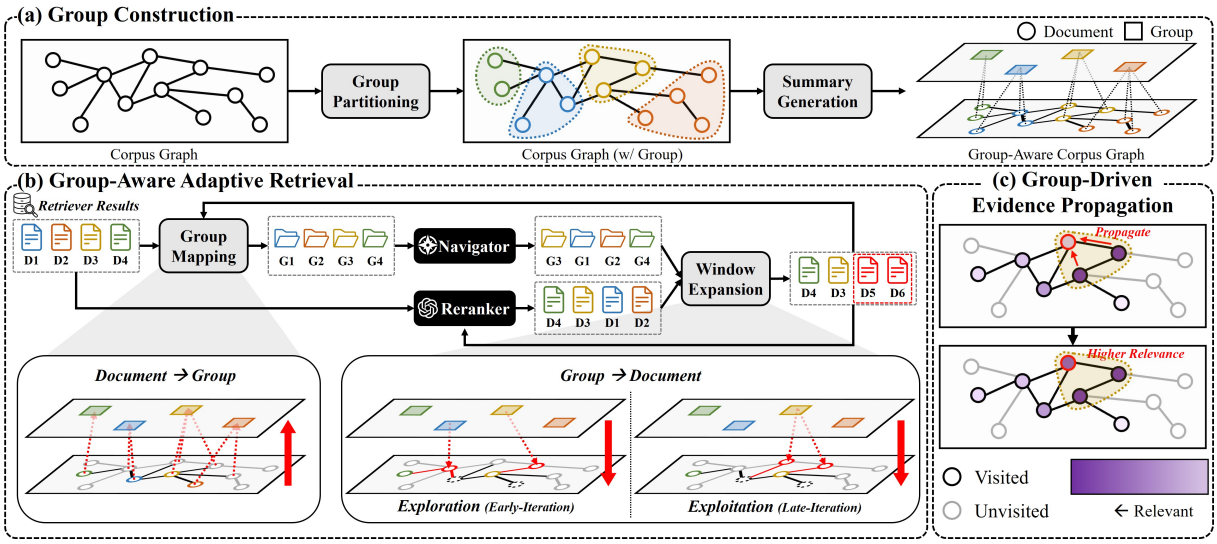

Figure 2: The overall framework of GAREN. (a) The corpus graph is partitioned into semantic groups, each summarized by an LLM. (b) During retrieval, GAREN iteratively reranks the current window and expands it with documents from groups identified by the navigator under an *explore-then-exploit* strategy. (c) After retrieval, observed documents are re-scored using rank-weighted evidence from selected documents within the same group.

as document-document relationships (Rathee et al., 2025b,c). More recent LLM-based methods use LLM-based scoring to guide graph traversal with richer semantic interpretation (Rathee et al., 2025a; Xu and Chen, 2025). Other work incorporates intermediate reasoning steps as feedback for adaptive expansion or uses LLM preferences to construct corpus graphs (Kim et al., 2026; Yoon et al., 2026). Despite these advances, relying primarily on signals from local candidate documents can make it difficult to assess broader expansion directions.

2.3 Reasoning over Multiple Granularities

Prior work has shown that the choice of retrieval unit significantly affects both retrieval and downstream QA performance (Chen et al., 2024). Building on this insight, recent RAG methods organize corpora into hierarchical structures, enabling retrieval over both fine-grained evidence and higher-level abstractions (Edge et al., 2024; Sarthi et al., 2024; Wang et al., 2026). In retrieval, recent work introduces an LLM-guided hierarchical retrieval framework that performs top-down traversal over a pre-constructed semantic tree (Gupta et al., 2025). In contrast, our approach is designed for iterative adaptive retrieval, where coarse-grained group signals and fine-grained document feedback guide expansion toward relevant documents more reliably.

3 Preliminary

Listwise Reranking. Given a query q and a candidate window $\mathcal{W} = (d_1, \ldots, d_w)$ of size w, listwise reranking aims to reorder candidates so that documents relevant to q are placed higher in the ranking. We denote the listwise reranker by Rerank, where

$$\tilde{\mathcal{W}} = \mathrm{Rerank}(q, \mathcal{W}). \tag{1}$$

The output $\tilde{\mathcal{W}}$ is an ordered list of the candidate documents.

Adaptive Retrieval over Corpus Graphs. Reranking can only reorder documents that are already included in the candidate window. To retrieve relevant documents outside this window, adaptive retrieval updates the window using a pre-constructed corpus graph $G = (D, E)$, where nodes are documents and edges capture semantic proximity. Given a reranked window $\tilde{\mathcal{W}}$, adaptive retrieval preserves the top h documents (the keep size) and appends $w - h$ graph-expanded candidates:

$$\mathcal{X} = \mathrm{Expand}(q, \tilde{\mathcal{W}}, G, w - h), \tag{2}$$

$$\mathcal{W}_{\text{next}} = \tilde{\mathcal{W}}[:h] \oplus \mathcal{X}, \tag{3}$$

where $\mathcal{X}$ denotes the graph-expanded candidates and $\oplus$ denotes list concatenation. Existing methods typically realize $\mathrm{Expand}(\cdot)$ by deriving expansion anchors from the evidence in the current window. However, because they lack visibility into the information reachable through each potential expansion direction, they may fail to identify which direction leads toward relevant documents. We instead perform expansion at the group level, using group representations as coarse previews of the information reachable through alternative directions.

4 Proposed Method

We propose GAREN, a novel group-aware adaptive retrieval framework, as illustrated in Figure 2. GAREN first partitions the corpus graph into semantically coherent groups (Section 4.1). During retrieval, GAREN uses a navigator to select expansion directions at the group level (Section 4.2). After retrieval, GAREN additionally applies a lightweight score refinement step that incorporates group evidence into the final ranking (Section 4.3).

4.1 Group Construction

Corpus graph expansion typically exposes only local document neighborhoods at each step, which makes it difficult to decide an expansion direction from a broader semantic context. To provide a larger decision unit, we first construct semantically coherent document groups over the corpus graph. This construction is performed offline on a pre-constructed corpus graph $G = (D, E)$, where nodes correspond to documents and edges capture embedding-based semantic proximity between them.

Group Partitioning. For groups to serve as distinguishable expansion directions, each group should form a semantically coherent unit while remaining distinct from the others. We realize this property through graph *modularity* (Newman, 2006), which favors partitions with dense connections within groups and relatively sparse connections across groups. Specifically, we apply Leiden community detection (Traag et al., 2019) to optimize *modularity* over the document-level semantic-proximity graph G, yielding a partition into densely connected document groups:

$$\mathcal{P} = \{C_1, \ldots, C_M\}. \tag{4}$$

The groups in $\mathcal{P}$ form a partition of the corpus, and we denote the unique group containing document d by $\Omega(d) \in \mathcal{P}$. Detailed partitioning procedures and hyperparameters are described in Appendix A.

Summary Generation. For each group C_m, we use an LLM to generate a summary S_m of its shared semantic context. These summaries serve as coarse previews of information accessible through each expansion direction, without requiring all documents in a group to be examined individually.

4.2 Group-Aware Adaptive Retrieval

During retrieval, we use the constructed groups and summaries to guide expansion toward relevant documents. At each iteration, the navigator selects promising expansion directions from group-level information, and the reranker evaluates the documents exposed along those directions. Algorithm 1 summarizes the overall procedure.

Algorithm 1 Group-Aware Adaptive Retrieval

Input: query q, initial ranked result $\mathcal{R}_0$, corpus graph G, group map Ω, neighborhood N_k, window size w, keep size h, iterations T, switch iteration τ_{switch}

▷ **Window Initialization**
$\mathcal{W}_1 \leftarrow \mathcal{R}_0[:w]$
for $t = 1$ to T **do**
  ▷ **Document Reranking**
  $\tilde{\mathcal{W}}_t \leftarrow \text{Rerank}(q, \mathcal{W}_t)$
  if $t < T$ **then**
    ▷ **Group Mapping**
    $\mathcal{G}_t^{\text{cand}} \leftarrow \{\, \Omega(d') \mid d' \in \bigcup_{d \in \mathcal{W}_t} N_k(G, d) \,\}$
    ▷ **Group Scoring**
    $\mathcal{G}_t^{\text{score}} \leftarrow \text{Navigate}(q, \mathcal{G}_t^{\text{cand}})$
    ▷ **Window Expansion**
    if $t \leq \tau_{\text{switch}}$ **then**
      $\mathcal{X}_t \leftarrow \text{Expand}^{\text{explore}}(\mathcal{G}_t^{\text{score}}, w - h)$
    else
      $\mathcal{X}_t \leftarrow \text{Expand}^{\text{exploit}}(\mathcal{G}_t^{\text{score}}, w - h)$
    end if
    $\mathcal{W}_{t+1} \leftarrow \tilde{\mathcal{W}}_t[:h] \oplus \mathcal{X}_t$
  end if
end for
return $\tilde{\mathcal{W}}_T$

Since document-level reranking follows the retrieval formulation introduced in Section 3, we focus below on the group-level expansion mechanism. We first describe how the navigator scores candidate expansion directions, and then explain how our *explore-then-exploit* strategy retrieves documents from these directions.

4.2.1 Group-level Direction Scoring

Let $\mathcal{W}_t$ be the document window at iteration t, and $N_k(G, d)$ the neighborhood of document d in the pre-constructed k-NN corpus graph G. Rather than scoring the entire group space at every iteration, we construct a set of candidate expansion directions around the current retrieval state:

$$\mathcal{G}_t^{\text{cand}} = \{\, \Omega(d') \mid d' \in \bigcup_{d \in \mathcal{W}_t} N_k(G, d) \,\}. \tag{5}$$

Here, $\mathcal{W}_t$ specifies the current document-level retrieval state, while $\mathcal{G}_t^{\text{cand}}$ comprises the group-level directions reachable from that state. The navigator evaluates these candidate directions by examining the summaries of the groups in $\mathcal{G}_t^{\text{cand}}$, which provide coarse previews of the information accessible through each direction:

$$\mathcal{G}_t^{\text{score}} = \text{Navigate}(q, \mathcal{G}_t^{\text{cand}}). \tag{6}$$

The resulting $\mathcal{G}_t^{\text{score}}$ provides scored group-level directions for the subsequent expansion step. The navigator evaluates at most K_g candidate groups at each iteration to keep the cost bounded. The detailed scoring procedure is provided in Appendix E.

4.2.2 Explore-then-Exploit Expansion

Given the group scores in $\mathcal{G}_t^{\text{score}}$, GAREN selects groups for the next expansion. In early iterations, the limited observed evidence makes it risky to concentrate expansion on a single high-scoring group, as an incorrect early decision can misguide subsequent traversal. We therefore adopt an explore-then-exploit strategy: GAREN initially expands across multiple promising groups, and shifts to concentrating on high-scoring groups at iteration τ_{switch}, once relevant evidence has accumulated.

- **Exploration**. Early in the traversal, the limited number of observed documents makes the best expansion direction uncertain. We therefore perform *breadth-first expansion*, retrieving new documents from multiple high-scoring groups rather than concentrating on a single group. Specifically, groups in $\mathcal{G}_t^{\text{score}}$ are ranked by their navigator scores, and unobserved documents are selected from them in a round-robin manner. This mitigates error propagation from incorrect early decisions and preserves opportunities to reach groups containing relevant evidence.
- **Exploitation**. As more documents are observed and reranked, the accumulated evidence provides a more reliable signal for identifying promising expansion directions. We therefore perform *depth-first expansion*, retrieving new documents preferentially from the highest-scoring groups. Specifically, groups are visited in descending order of their navigator scores, and unobserved documents are selected from a higher-scoring group before moving to the next one. This focuses subsequent traversal on groups supported by stronger accumulated evidence.

Finally, we obtain the expansion documents $\mathcal{X}_t$ following the detailed procedure in Appendix E, and construct the next retrieval window as:

$$W_{t+1} = \tilde{W}_t[:h] \oplus \mathcal{X}_t. \tag{7}$$

4.3 Group-Driven Evidence Propagation

After the group-aware adaptive retrieval loop terminates, we apply a lightweight post-processing step called *Group-Driven Evidence Propagation* over the observed document set. This step is motivated by prior work that regularizes retrieval scores based on the clustering hypothesis (Diaz, 2005). Using the final ranking obtained after evidence has accumulated through iterative retrieval, we propagate rank-weighted support from selected documents to other observed documents in the same group.

Let R_{final} denote the final top-10 ranked results. We first assign each selected document an RBP-style base weight (Moffat and Zobel, 2008):

$$b(d) = \begin{cases} (1-p)p^{\text{rank}(d)-1}, & d \in R_{\text{final}}, \\ 0, & \text{otherwise}. \end{cases} \tag{8}$$

Here, $\text{rank}(d)$ denotes the position of d in R_{final}, and p controls how quickly the weight decays with rank. This rank-based decay assigns stronger evidence to higher-ranked documents, allowing them to contribute more strongly to group-level support.

For each observed document, *Group-Driven Evidence Propagation* combines its document-level evidence with the rank-weighted evidence accumulated in its group:

$$\text{score}(d) = b(d) + \alpha \sum_{\Omega(d')=\Omega(d)} b(d'). \tag{9}$$

Here, α controls how strongly evidence from neighboring selected documents is propagated.

Finally, we sort all observed documents by $\text{score}(d)$ to obtain the final ranking. This post-processing step helps promote potentially relevant documents belonging to evidence-rich semantic groups, but not sufficiently promoted by document-level reranking alone.

5 Experimental Setup

Datasets. We evaluate GAREN on three retrieval benchmarks: BRIGHT (Su et al., 2025), R2MED (Li et al., 2025), and BEIR (Thakur et al., 2021). BRIGHT is a reasoning-intensive benchmark across diverse domains, and R2MED evaluates reasoning-driven medical retrieval. BEIR provides diverse retrieval tasks; following prior work (Sun et al., 2023), we evaluate on eight BEIR datasets. For metrics, we report nDCG@10 and Recall@100 (R@100), where R@100 measures recall over the documents observed by the reranker.

Baselines. We compare our method with four baselines. (i) **Retrieve-and-Rerank** performs sliding-window reranking (Sun et al., 2023) within a fixed first-stage retrieval pool, without expanding the pool through the corpus graph. (ii) **SlideGAR**

Method	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.4	4.9	14.5
Non-Reasoning-based Reranking													
Retrieve-and-Rerank	34.0	41.6	21.7	28.7	29.7	20.2	29.9	17.5	14.5	3.6	13.7	14.8	22.5
SlideGAR	40.0	42.1	25.9	32.0	28.2	19.9	32.7	14.4	12.4	**4.1**	11.6	22.6	23.8
RGS	43.3	42.0	28.5	35.4	30.0	22.9	**34.5**	**18.2**	12.5	3.2	15.0	**32.0**	26.4
(Ours) GAREN	**45.3**	**43.6**	**30.3**	**38.5**	**31.6**	**25.2**	34.4	16.7	**20.4**	**4.1**	**17.8**	31.0	**28.3**
Reasoning-based Reranking													
Retrieve-and-Rerank	34.7	42.5	23.8	29.0	31.6	23.2	32.0	**23.4**	24.1	5.7	15.7	13.4	24.9
SlideGAR	42.8	43.1	29.5	31.3	31.1	25.0	36.5	17.3	20.5	3.6	14.3	16.8	26.0
RGS	46.6	46.0	32.4	35.3	33.3	29.4	35.9	21.7	19.9	3.3	17.8	25.7	28.9
REPAIR	44.6	43.7	32.3	33.7	32.0	29.2	36.1	20.3	24.3	3.6	15.3	25.3	28.4
(Ours) GAREN	**47.1**	**46.5**	**34.1**	**40.6**	**37.1**	**31.0**	**37.8**	21.1	**25.0**	**6.2**	**18.6**	**29.9**	**31.2**

Table 1: nDCG@10 performance of various methods and reranking settings on BRIGHT. The best performance is marked in **bold**, and the second-best is underlined within the same reranking prompt setting. Across all BRIGHT queries, GAREN significantly outperforms the best-performing baseline in both reranking settings under a query-level paired t-test ($p < 0.05$).

(Rathee et al., 2025a) alternates between the retrieval pool and the corpus graph by expanding the window with document-level graph neighbors. (iii) **RGS** (Xu and Chen, 2025) performs adaptive retrieval on an ANN corpus graph. (iv) **REPAIR** (Kim et al., 2026) performs graph-based adaptive retrieval guided by intermediate reasoning steps.

Implementation Details. For all methods, we use BM25 (Robertson and Walker, 1994) for first-stage retrieval, construct the corpus graph using Qwen3-Embedding-4B (Zhang et al., 2025), and use Qwen3-Next-80B-A3B-Instruct-FP8 (Qwen Team, 2025) as the reranker with a window size of $w = 20$, a keep size of $h = 10$, and a total reranking budget of 100 documents. All experiments are conducted with the vLLM (Kwon et al., 2023) inference engine, and we report results averaged over 5 runs due to the variance of vLLM inference. Unless otherwise specified, we use the non-reasoning prompt setting. For GAREN, we additionally use Qwen3-Next-80B-A3B-Instruct-FP8 for offline group summary generation and Qwen3-Reranker-4B as the navigator, with a group scoring budget of $K_g = 20$ and $\tau_{\text{switch}} = 4$. Additional hyperparameters are in Appendix A.

6 Results and Analysis

6.1 Main Results

Effectiveness on Reasoning-Intensive Retrieval. Table 1 compares GAREN with baselines on BRIGHT under both non-reasoning and reasoning prompting settings. (i) In the non-reasoning setting, GAREN improves average nDCG@10 over Retrieve-and-Rerank by 25.8%, demonstrating the benefit of expanding beyond the initial retrieval pool, where relevant documents may be missing. GAREN also outperforms adaptive retrieval baselines such as SlideGAR and RGS by 18.9% and 7.2%, indicating that group-level expansion provides more effective expansion directions than document-level traversal based on local implicit signals. (ii) In the reasoning setting, GAREN again achieves the strongest overall performance, improving over Retrieve-and-Rerank by 25.3%. Although REPAIR improves expansion decisions using reasoning-based anchors, GAREN achieves stronger performance by explicitly comparing group-level expansion directions. (iii) Across both settings, reranking-based methods show limited effectiveness on AoPS and LeetCode, and adaptive

Method	R2MED		BEIR	
	nDCG@10	R@100	nDCG@10	R@100
BM25	15.1	44.6	43.4	43.7
Retrieve-and-Rerank	35.0	44.6	51.0	43.7
SlideGAR	42.7	62.9	51.3	**48.6**
RGS	44.5	63.7	52.0	43.7
(Ours) GAREN	**45.1**	**67.6**	**52.4**	47.4

Table 2: Performance comparison on R2MED and BEIR. The best performance is marked in **bold**, and the second-best is underlined.

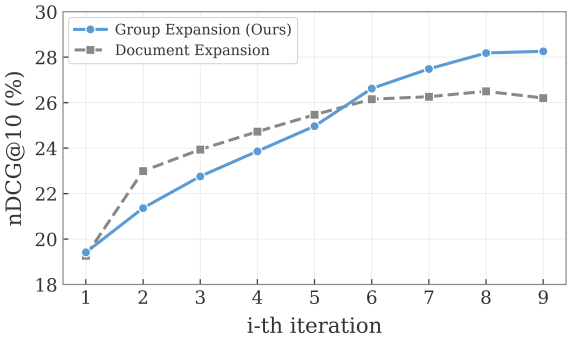

Figure 3: Performance comparison of Document Expansion and Group Expansion. Unlike Document Expansion, Group Expansion consistently improves performance as the number of iterations increases.

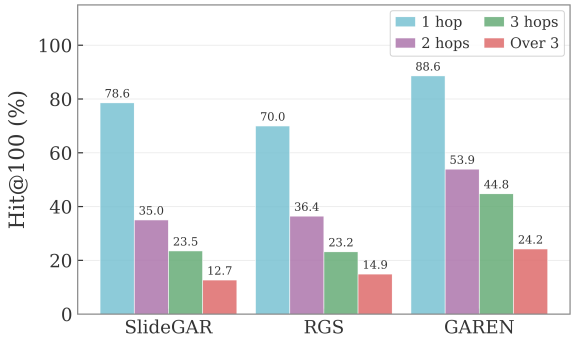

Figure 4: Hit@100 comparison between GAREN and baseline methods by graph-hop distance to the nearest gold document on BRIGHT. The gap between GAREN and the baselines widens as gold documents become farther from the initial retrieval results.

expansion provides little additional gain. In these subsets, the reranker and the embedding model often fail to identify the underlying algorithmic or mathematical techniques that connect relevant documents, favoring surface-level overlap instead. The effectiveness of our method is thus expected to improve as stronger rerankers and embedding models become available.

Evaluation on R2MED and BEIR. As shown in Table 2, GAREN achieves competitive performance on both benchmarks. These results indicate that group-level guidance generalizes beyond BRIGHT to both medical reasoning and traditional retrieval settings. The improvement margins are relatively modest compared with BRIGHT, because first-stage retrieval can already recover gold documents on these benchmarks.

6.2 Document-Level vs. Group-Level Expansion

We examine the benefit of group-level over document-level expansion by comparing GAREN against a document-level variant that directly scores neighboring documents $N_k(d)$ connected to the current window with the navigator. [1] As shown in Figure 3, the document-level variant improves rapidly in early iterations but quickly saturates. Direct neighbor scoring is effective near the current window, but has limited ability to discover relevant documents beyond immediate graph neighbors. In contrast, group-level expansion improves more steadily and eventually surpasses the document-level variant. By evaluating semantic groups, GAREN selects expansion directions using broader contextual signals beyond immediate neighbors. We further analyze this gap in Appendix C.1, showing that the advantage of group-level expansion grows as gold documents become more distant, which is structurally explained by the group graph substantially reducing the graph distance to gold documents.

[1] For each step, the document-level variant can observe up to window size $\times$ #edges $= 20 \times 16 = 320$ documents.

Strategy	Propagation	Navigator	BRIGHT Avg.	
			nDCG@10	R@100
Explore-then-Exploit	✓	✓	**28.3**	**53.1**
Exploration	✓	✓	27.0	50.2
Exploitation	✓	✓	27.8	51.1
Explore-then-Exploit	✗	✓	27.9	**53.1**
Explore-then-Exploit	✓	✗	21.2	38.4

Table 3: Ablation study of GAREN on BRIGHT. We analyze the effects of navigator, expansion strategy, and propagation.

6.3 Gold Document Location Analysis

We analyze whether GAREN retrieves gold documents far from the initial documents. On BRIGHT, we group queries by the minimum graph-hop distance from the BM25 top-20 set to a gold document, and report Hit@100, indicating whether at least one gold document is observed during retrieval. We focus on queries whose gold documents lie outside the BM25 top-20 set and must therefore be reached through expansion. As shown in Figure 4, GAREN observes more gold documents than graph-based baselines across all hop distances. SlideGAR performs well only for nearby gold documents, while RGS reaches farther regions through ANN graph traversal but shows a trade-off between nearby and distant cases. GAREN maintains strong performance across distances, supporting our hypothesis that group-level expansion can reach gold documents beyond local document-level traversal.

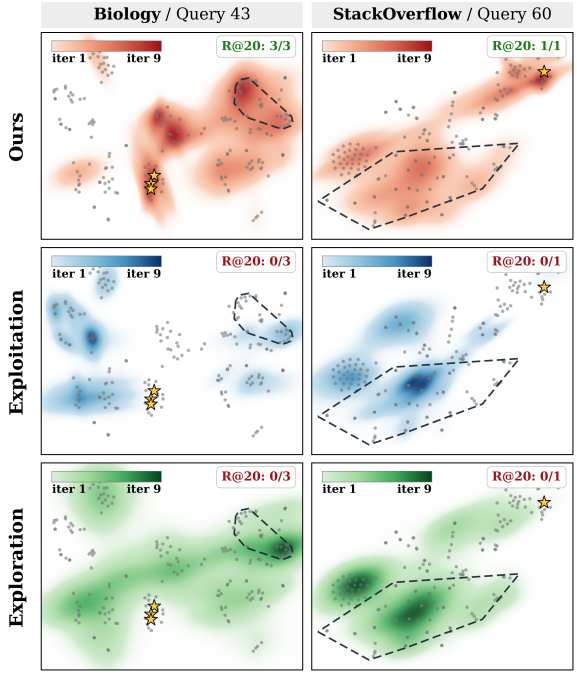

Figure 5: Case study on expansion strategies. GAREN successfully retrieves the gold documents through an explore-then-exploit strategy. The dashed area denotes the initial region at iteration 1.

6.4 Ablation Study

We construct several variants to assess the impact of each component in GAREN and evaluate them on BRIGHT, as shown in Table 3.

Explore-then-Exploit Strategy. Using a single expansion strategy degrades performance: *Exploration* fails to concentrate on promising directions in later iterations, while *Exploitation* commits too early before sufficient evidence emerges to identify directions leading to relevant documents. These results confirm that our *explore-then-exploit* strategy is effective for reaching relevant documents throughout the retrieval process.

Group-Driven Evidence Propagation. Removing this component degrades the final ranking quality, showing that sharing evidence from high-confidence documents with other members of the same group helps prioritize relevant documents.

Navigator Guidance. We replace navigator-based group scoring with a connectivity-based heuristic while maintaining all other components. Specifically, candidate groups are scored based on their connectivity to the neighboring documents in the current window. This replacement reduces nDCG@10 and R@100 by 25.1% and 27.7%, respectively, highlighting the importance of semantic group evaluation for expansion.

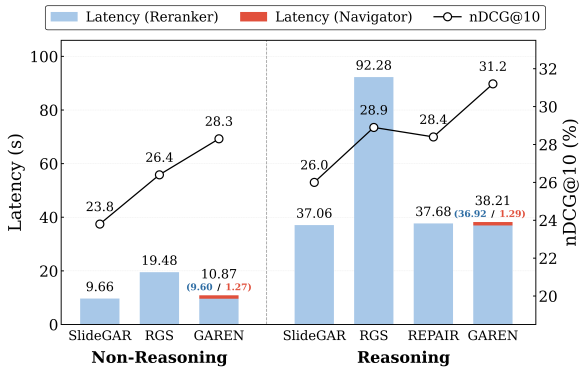

Figure 6: Retrieval effectiveness and latency comparison on BRIGHT, measured on 30 queries sampled from each subset. GAREN achieves strong retrieval performance with competitive latency under both prompting settings. Values in parentheses indicate the latency breakdown of the reranker and navigator, respectively.

6.5 Case Study for Expansion Strategy

We conduct a case study to visualize how different expansion strategies traverse the corpus, as shown in Figure 5. *Exploitation* focuses only on the highest-scoring groups, remaining near the seed set and failing to reach the gold documents. *Exploration* spreads expansion across many groups, exploring diverse directions and approaching the gold documents but failing to concentrate enough to retrieve them. In contrast, GAREN first explores diverse directions broadly, and then concentrates subsequent expansion on the most promising direction, successfully retrieving the gold documents. This confirms that the *explore-then-exploit* strategy is necessary to both *explore* promising directions and *exploit* them to retrieve gold documents.

6.6 Efficiency Analysis

To evaluate the efficiency of GAREN, we measure latency on BRIGHT. We sample 30 queries from each subset, resulting in 360 queries in total. The reported nDCG scores are computed over the full set of queries. Experiments are conducted on two A100 GPUs. All queries are processed sequentially, and we report the average latency. As shown in Figure 6, GAREN achieves high accuracy at competitive latency under both prompting settings. These results indicate that the navigator, being substantially smaller than the reranker, introduces only limited overhead while enabling group-level navigation. This design choice is further supported in Appendix B.1, where scaling the reranker yields substantially larger gains than scaling the navigator, indicating that a lightweight navigator

is sufficient for group-level direction assessment. Moreover, since the reranker and navigator operate independently on the current window, they can be executed in parallel in practical deployments to further reduce latency.

7 Conclusion

In this paper, we propose **GAREN**, a group-aware adaptive retrieval framework for addressing the *bounded recall problem* in reasoning-intensive retrieval. Existing adaptive retrieval methods rely on local signals from the current candidates to determine expansion directions, which can be ineffective when the initial retrieval state provides limited or unreliable clues. To mitigate this, GAREN organizes the corpus graph into coarse-grained *groups* and coordinates a group-level navigator with a document-level reranker through an *explore-then-exploit* strategy. A *group-driven evidence propagation* step further consolidates evidence around high-confidence candidates. Experiments demonstrate that GAREN outperforms strong adaptive retrieval baselines, particularly when relevant documents are distant from the initial retrieval pool.

8 Limitations

We discuss the limitations of GAREN as follows:

Group Construction Strategy. We construct document groups using community-detection-based partitioning over the corpus graph, as our primary focus is on leveraging group-level structure for adaptive retrieval. While this design performs effectively in our experiments, more advanced construction strategies may produce groups that provide more informative expansion directions. For example, learned partitioning, alternative community detection algorithms, or corpus graphs built from stronger embeddings could further improve retrieval performance.

Fixed Explore-then-Exploit Transition. We transition from exploration to exploitation using a fixed parameter τ_{switch}. This schedule performs effectively across our experiments by encouraging broad exploration in early iterations and more focused expansion in later iterations. Incorporating query-specific characteristics or intermediate retrieval signals could enable a more adaptive transition that better identifies when sufficient evidence has accumulated for focused expansion.

Ethics Statement

This work adheres to the ACL's ethical guidelines. All scientific resources were obtained under permissive licenses and used for their intended research purposes.

Acknowledgments

This work was supported by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. NRF-RS-2025-00564083, IITP-RS-2022-II220680, IITP-2025-RS-2020-II201821, IITP-2026-RS-2024-00437633) and the SEMES-SKKU collaboration funded by SEMES, each contributing 20% to this research.

References

Peter Baile Chen, Tomer Wolfson, Michael J. Cafarella, and Dan Roth. 2025. Enrichindex: Using llms to enrich retrieval indices offline. *CoRR*.

Tong Chen, Hongwei Wang, Sihao Chen, Wenhao Yu, Kaixin Ma, Xinran Zhao, Hongming Zhang, and Dong Yu. 2024. Dense X retrieval: What retrieval granularity should we use? In *EMNLP*.

Debrup Das, Seán Ó Nualláin, and Razieh Rahimi. 2025. RaDeR: Reasoning-aware dense retrieval models. In *EMNLP*.

Fernando Diaz. 2005. Regularizing ad hoc retrieval scores. In *CIKM*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, and Jonathan Larson. 2024. From local to global: A graph RAG approach to query-focused summarization. *CoRR*.

Gemma Team. 2026. Gemma 4 technical report.

Nilesh Gupta, Wei-Cheng Chang, Ngot Bui, Cho-Jui Hsieh, and Inderjit S. Dhillon. 2025. Llm-guided hierarchical retrieval. *CoRR*.

N. Jardine and Cornelis Joost van Rijsbergen. 1971. The use of hierarchic clustering in information retrieval. *Inf. Storage Retr.*

Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In *NeurIPS*.

Jongho Kim, Jaeyoung Kim, Jihyuk Kim, Yu Jin Kim, Seungwon Hwang, and Moontae Lee. 2026. Adaptive retrieval for reasoning. In *ACL*.

Hrishikesh Kulkarni, Sean MacAvaney, Nazli Goharian, and Ophir Frieder. 2023. Lexically-accelerated dense retrieval. In *SIGIR*.

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 *SOSP*.

Sangam Lee, Ryang Heo, SeongKu Kang, and Dongha Lee. 2025. Imagine all the relevance: Scenario-profiled indexing with knowledge expansion for dense retrieval. *CoRR*.

Yibin Lei, Tao Shen, and Andrew Yates. 2025. Thinkqe: Query expansion via an evolving thinking process. In *Findings of the Association for Computational Linguistics: EMNLP*.

Lei Li, Xiao Zhou, and Zheng Liu. 2025. R2MED: A benchmark for reasoning-driven medical retrieval. *CoRR*.

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

Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. Adaptive re-ranking with a corpus graph. In *CIKM*.

Alistair Moffat and Justin Zobel. 2008. Rank-biased precision for measurement of retrieval effectiveness. *ACM Trans. Inf. Syst.*

M. E. J. Newman. 2006. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*.

Qwen Team. 2025. Qwen3 technical report.

Qwen Team. 2026. Qwen3.5: Towards native multimodal agents.

Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025a. Guiding retrieval using llm-based listwise rankers. In *ECIR*.

Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025b. Quam: Adaptive retrieval through query affinity modelling. In *WSDM*.

Mandeep Rathee, Venktesh V, Sean MacAvaney, and Avishek Anand. 2025c. Breaking the lens of the telescope: Online relevance estimation over large retrieval sets. In *SIGIR*.

Stephen E. Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: recursive abstractive processing for tree-organized retrieval. In *ICLR*.

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. *CoRR*.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han-yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan Ö. Arik, Danqi Chen, and Tao Yu. 2025. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. In *ICLR*.

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. In *EMNLP*.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*.

Vincent A. Traag, Ludo Waltman, and Nees Jan van Eck. 2019. From louvain to leiden: guaranteeing well-connected communities. *Scientific reports*.

Shu Wang, Yixiang Fang, Yingli Zhou, Xilin Liu, and Yuchi Ma. 2026. Archrag: Attributed community-based hierarchical retrieval-augmented generation. In *AAAI*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *SIGIR*.

Haike Xu and Tong Chen. 2025. Beyond sequential reranking: Reranker-guided search improves reasoning intensive retrieval. *CoRR*.

Soyoung Yoon, Jongho Kim, Daeyong Kwon, Avishek Anand, and Seungwon Hwang. 2026. On listwise reranking for corpus feedback. In *WSDM*.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*.

Yunfei Zhong, Jun Yang, Yixing Fan, Jiafeng Guo, Lixin Su, Maarten de Rijke, Ruqing Zhang, Dawei Yin, and Xueqi Cheng. 2025. Reasoning-enhanced query understanding through decomposition and interpretation. *CoRR*.

A Additional Setup

Corpus Graph. We construct the corpus graph by connecting each document to its top 16 most similar documents based on Qwen3-Embedding-4B representations. For RGS, we follow the original implementation and use DiskANN (Jayaram Subramanya et al., 2019) for ANN graph construction.

Group Construction. We partition the corpus graph using the Leiden community detection algorithm. We set the resolution parameter to $\gamma = 1.0$ and the randomness parameter to $\theta = 0.001$. To encourage fine-grained groups, we use a re-partitioning threshold of $\bar{s} = 20$: groups exceeding this threshold are recursively re-partitioned whenever Leiden identifies a non-trivial subdivision. Consequently, larger groups may remain when no further meaningful partition is identified. After partitioning, each group is summarized once offline using the summary prompt shown in Appendix G.

Group-Driven Evidence Propagation. We set the RBP decay parameter to $p = 0.6$ and the neighbor propagation weight to $\alpha = 0.01$ in the non-reasoning reranking setting, and $p = 0.5$ and $\alpha = 0.01$ in the reasoning setting. These values were selected via greedy search on the development splits of NFCorpus (324 queries) and FiQA (500 queries) in BEIR.

B Extended Experiments

B.1 Pipeline Robustness

We examine whether GAREN remains effective across various configurations of the retrieval pipeline by varying the first-stage retrieval, the corpus graph embedding, and the reranker and navigator models.

First-Stage Retrieval. We evaluate whether GAREN remains effective when the initial retrieval pool is strengthened. To this end, we replace the BM25 first-stage retriever with ReasonIR (Shao et al., 2025), a retriever trained for reasoning-intensive retrieval, and further strengthen the initial pool with GPT-4 queries from BRIGHT (Su et al., 2025). All methods are evaluated under the reasoning prompt setting. As shown in Table 4, since the initial pool already contains many relevant documents, fewer documents remain to be recovered through expansion and the margins between methods are narrower than in the BM25 setting. Nevertheless, GAREN still achieves the best performance under the stronger pool, showing that group-level guidance reliably identifies useful expansion directions even when the room for expansion is limited.

Corpus Graph Embedding. We also evaluate whether GAREN is robust to the embedding model used for corpus graph construction by replacing Qwen3-Embedding-4B with BGE-Large (Xiao et al., 2024). As shown in Table 5, GAREN continues to outperform all baselines on BRIGHT under both prompting settings, indicating that the gains of group-aware adaptive retrieval do not rely on a particular embedding model.

Scaling the Reranker and the Navigator. We analyze how the capacity of each component affects performance. To this end, we vary the size of the reranker and the navigator across three scales within each of the Qwen3.5 (Qwen Team, 2026) and Gemma4 (Gemma Team, 2026) model families, and measure nDCG@10 and latency for all combinations. Since these models are not trained for pointwise relevance judgment, we obtain navigator scores using the same prompt as Qwen3-Reranker. As in Section 6.6, we measure latency on 30 queries sampled from each of the 12 BRIGHT subsets. As shown in Figure 7, scaling the reranker leads to substantially larger performance gains than scaling the navigator. This reflects the division of roles between the two components: while the navigator determines which directions are explored, promoting the exposed documents into the top ranks remains the reranker's responsibility. The latency comparison points the same way. Pairing a small navigator with a larger reranker is both faster and more accurate than the reverse. Direction assessment at the group level is thus a less demanding task than fine-grained relevance assessment at the document level, supporting our design choice of a lightweight navigator.

Reranker Backbones. To verify that GAREN is not tied to a specific reranker, we replace the reranker with GPT-4o-mini and Gemma4-26B-A4B-it, and evaluate all methods under the same backbone for a controlled comparison. As shown in Table 6, GAREN achieves the strongest performance in most domain categories across both backbones and under both prompting settings, indicating that the benefit of group-level expansion is not tied to a particular reranker backbone.

First-stage retrieval: *BM25 → ReasonIR with GPT-4 queries*													
Method	**StackExchange**							**Coding**		**Theorem-based**			**Avg.**
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	
ReasonIR	43.0	43.2	33.1	39.7	20.9	30.4	27.4	31.5	19.7	7.4	34.0	37.0	30.6
Reasoning-based Reranking													
Retrieve-and-Rerank	51.5	45.7	34.2	43.0	30.6	30.5	38.9	18.6	**29.5**	4.9	30.2	36.2	32.8
SlideGAR	53.2	46.3	34.0	43.8	30.0	29.4	37.9	17.2	26.7	4.3	29.3	36.9	32.4
RGS	**54.6**	**49.4**	**37.5**	46.3	31.0	**33.1**	39.0	19.4	23.6	5.4	32.2	38.1	34.1
REPAIR	51.9	46.8	33.9	43.5	30.4	29.3	**39.1**	16.3	24.1	4.6	30.9	37.1	32.3
(Ours) GAREN	53.7	48.1	37.1	**47.9**	**35.8**	32.9	39.0	**19.7**	24.2	**6.3**	**33.6**	**40.0**	**34.9**

Table 4: nDCG@10 performance of various methods on BRIGHT under the reasoning reranking setting, using the top-100 ReasonIR results with GPT-4 queries. The best performance is marked in **bold**, and the second-best is underlined within the reasoning-based reranking setting.

Corpus graph embeddings: *Qwen3-Embedding-4B → BGE-Large*													
Method	**StackExchange**							**Coding**		**Theorem-based**			**Avg.**
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	
BM25	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.4	4.9	14.5
Non-Reasoning-based Reranking													
Retrieve-and-Rerank	34.0	41.6	21.7	28.7	29.7	20.2	29.9	17.5	14.5	3.6	13.7	14.8	22.5
SlideGAR	42.4	41.4	26.5	31.8	27.9	20.9	33.5	15.2	14.6	**4.4**	12.1	17.6	24.0
RGS	43.3	**42.8**	27.1	36.3	30.1	21.9	34.3	**21.3**	14.2	3.7	14.5	23.8	26.1
(Ours) GAREN	**44.0**	41.1	**29.4**	**37.2**	**32.5**	**25.1**	**34.3**	15.6	**27.0**	4.2	**16.1**	**33.7**	**28.3**
Reasoning-based Reranking													
Retrieve-and-Rerank	34.7	42.5	23.8	29.0	31.6	23.2	32.0	**23.4**	24.1	5.7	15.7	13.4	24.9
SlideGAR	40.2	42.3	28.3	31.0	30.2	27.6	35.2	21.3	23.2	**5.8**	14.8	16.2	26.3
RGS	**45.2**	**44.9**	32.3	38.1	32.2	26.9	36.7	20.2	21.4	4.7	16.5	27.0	28.8
REPAIR	41.8	44.3	31.1	33.9	31.4	29.1	37.0	23.0	21.6	5.1	16.0	23.7	28.2
(Ours) GAREN	43.1	44.0	**33.1**	**38.9**	**33.0**	**30.1**	**37.4**	20.3	**35.6**	5.7	**17.4**	**30.3**	**30.7**

Table 5: nDCG@10 on BRIGHT when constructing the corpus graph with BGE-Large embeddings. The best performance is marked in **bold**, and the second-best is underlined within the same reranking prompt setting.

B.2 Comparison with Hierarchical Retrieval

LATTICE (Gupta et al., 2025) performs retrieval by organizing the entire corpus into a semantic tree and having an LLM traverse the hierarchy top-down, from high-level abstractions to leaf documents. In contrast, GAREN starts from an initial document retrieval state and expands it iteratively: the group-level navigator evaluates which surrounding semantic directions to explore, while the document-level reranker assesses the concrete documents reached along those directions. Thus, LATTICE uses coarse-to-fine hierarchical traversal as the retrieval process, whereas GAREN couples coarse-grained directional guidance with fine-grained document feedback for adaptive expansion.

For LATTICE, we use the official pre-built semantic trees released by the authors. Since both methods construct their corpus structures offline before retrieval, we compare only the LLM usage incurred during iterative retrieval. We estimate token cost using the input and output token counts of each LLM call during retrieval and the corresponding per-token prices reported by OpenRouter[2]. For a controlled comparison, we stop LATTICE after it has examined 100 documents, matching the document observation budget used in GAREN. Table 7 reports retrieval effectiveness and LLM usage and cost incurred during iterative retrieval on BRIGHT. LATTICE achieves strong retrieval effectiveness, but its LLM-driven traversal incurs substantially higher generation cost. With BM25 as the first-

[2] https://openrouter.ai/

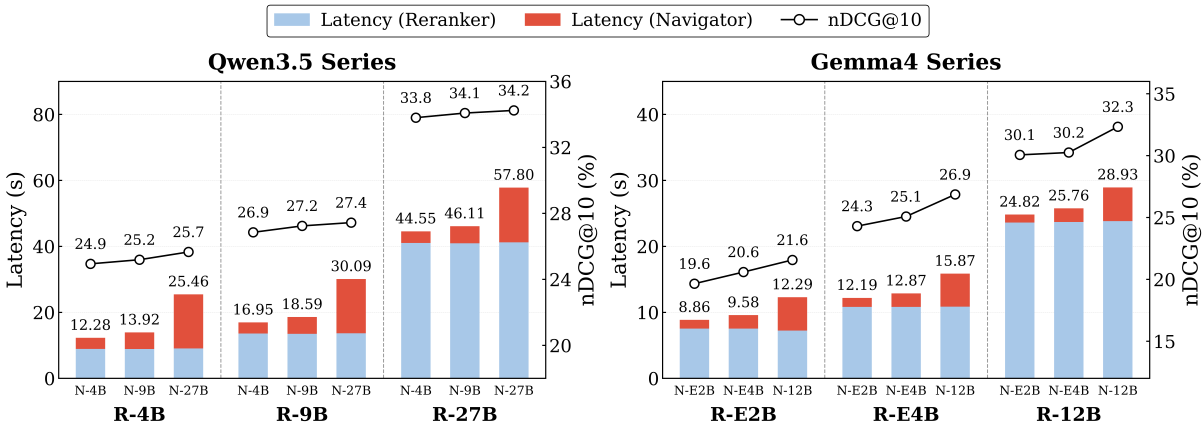

Figure 7: nDCG@10 and latency across reranker (R) and navigator (N) sizes on BRIGHT. Bars decompose latency into the reranker and the navigator components.

Method	GPT			Gemma4		
	SE	CD	TH	SE	CD	TH
BM25	17.2	16.1	7.1	17.2	16.1	7.1
Non-Reasoning Reranking						
Retrieve-and-Rerank	25.4	21.4	11.9	32.9	29.1	14.4
SlideGAR	27.0	19.3	14.2	37.2	26.0	15.3
RGS	27.0	21.6	15.7	37.3	26.1	20.6
(Ours) GAREN	**30.6**	**24.8**	**19.3**	**39.3**	**30.6**	**23.1**
Reasoning Reranking						
Retrieve-and-Rerank	27.4	26.0	12.4	33.5	**34.4**	13.7
SlideGAR	29.0	23.0	14.6	38.6	29.6	16.1
RGS	29.2	23.9	18.8	40.5	26.2	19.4
REPAIR	28.7	25.3	15.8	38.8	29.3	17.6
(Ours) GAREN	**32.2**	**27.6**	**20.5**	**42.3**	30.5	**23.9**

Table 6: nDCG@10 on BRIGHT grouped by domain categories. SE, CD, and TH denote StackExchange, Coding, and Theorem-based, respectively. The best and second-best performance within each reranking setting are marked in **bold** and underlined, respectively.

stage retriever, GAREN achieves slightly lower ranking performance while requiring substantially lower iterative-retrieval cost. When using an enhanced setup with GPT-4 queries and ReasonIR, GAREN achieves stronger retrieval effectiveness while maintaining lower iterative-retrieval cost than LATTICE. These results suggest that group-level adaptive expansion with document-level feedback can provide an effective and efficient alternative to top-down hierarchical traversal.

C Analysis of Group-Level Expansion

C.1 Structural Analysis of the Group Unit

Section 6.2 showed that group-level expansion surpasses document-level expansion as iterations proceed. In this section, we analyze how the two expansion strategies differ in their ability to reach gold documents at varying distances, and provide a structural explanation for this gap by comparing the distance to gold documents on the document graph and the group graph.

Method	nDCG@10	R@100	# Proc.	# Gen.	Cost ($)
LATTICE	31.9	49.2	85,419	13,859	0.0229
GAREN (BM25)	31.2	52.9	125,031	4,683	0.0109
Reranker	–	–	48,938	4,523	0.0094
Navigator	–	–	76,093	160	0.0015
GAREN (ReasonIR)	34.9	62.9	121,994	4,684	0.0106
Reranker	–	–	45,880	4,524	0.0091
Navigator	–	–	76,114	160	0.0015

Table 7: Overall performance and online iterative retrieval LLM cost comparison of GAREN with LATTICE on BRIGHT.

Reachability by distance. We measure how the reachability of the two expansion strategies varies with the distance to the gold document. Following the setting in Section 6.3, we group queries by the minimum document-hop distance from the initial retrieval set to a gold document and report Hit@100. The comparison target is the same document-level expansion variant as in Section 6.2, keeping all components identical except for the expansion unit. As shown in Figure 8, the two strategies show little difference at short distances, but the gap widens as the distance increases. Document-level expansion fails to reach any gold document at 7 or more hops, whereas group-level expansion maintains meaningful reachability even at long distances. This indicates that the performance gap between the two strategies stems from their ability to recover gold documents far from the initial set.

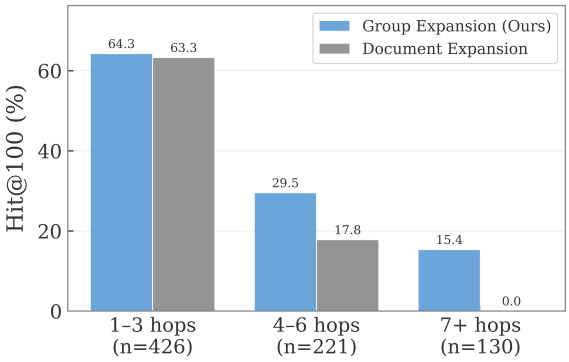

Figure 8: Hit@100 comparison between group-level and document-level expansion by graph-hop distance to the nearest gold document on BRIGHT. The gap between the two widens as gold documents become farther from the initial retrieval results.

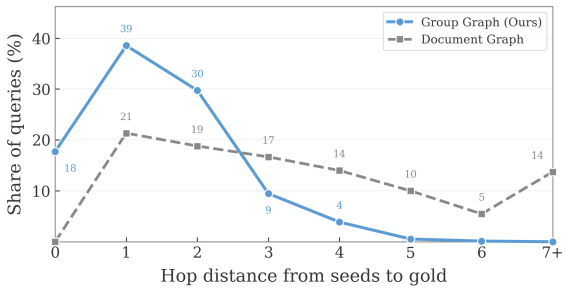

Figure 9: Distribution of the graph-hop distance to gold documents on the document graph and the group graph on BRIGHT. Distances are measured to the nearest gold document missed by first-stage retrieval, which must be reached through expansion.

Distance to gold documents. To analyze the structural cause of this gap, we focus on gold documents that are missed by first-stage retrieval and must therefore be reached through expansion. For each query, we measure the minimum hop distance from the initial retrieval set to such a gold document on the document graph and the group graph, respectively. We connect two groups with a directed edge when a document in one group points to a document in the other. As shown in Figure 9, gold documents are spread across a wide range of distances on the document graph, whereas most of them are concentrated within 3 hops on the group graph. As a result, group-level expansion can skip multiple document hops with a single group selection, reaching distant gold documents within a limited iteration budget.

C.2 Group Construction Strategies

We further compare strategies for constructing document groups. To control for the number of groups, we set the number of K-means clusters equal to the number of groups produced by our community detection construction, and randomly partition documents into the same number of groups for random grouping. As shown in Table 8, community detection performs best, suggesting that graph connectivity provides useful structure for defining expansion directions. Moreover, random grouping falls far behind both methods, showing that coarsening the graph alone brings no benefit. These results confirm that the effectiveness of the group unit comes from the meaningful signals that semantically coherent groups provide.

Group Construction	BRIGHT Avg.	
	nDCG@10	R@100
Random Group	23.1	31.2
K-means Group	26.7	50.2
Community Detection (Ours)	**28.3**	**53.1**

Table 8: Overall performance of different group construction methods in GAREN.

C.3 Failure Case Analysis

To understand why GAREN fails to expand gold documents, we analyze the retrieval process with respect to the *gold group*, the group containing the gold documents for a query. We first examine whether the gold group was scored by the navigator during retrieval. With BM25 first-stage retrieval, the gold group was never scored in 28.5% of the pairs. Stronger first-stage retrieval with ReasonIR and GPT-4 queries reduces this rate to 20.4% but does not eliminate such cases. Because these failures accumulate over sequential expansion decisions, they are difficult to attribute to specific causes. We therefore focus on the cases where the gold group was scored but not selected, qualitatively analyzing 10 sampled cases from each of the 12 BRIGHT subsets (120 in total) and reporting the proportion of each failure type. Representative examples are presented in Table 13.

1. **Failures in group construction (53.3%).** The gold document is not properly represented during group construction or summarization, in two ways. First, even when the document is assigned to a semantically appropriate group, the summary is generated around the content shared across the group and omits aspects specific to individual documents. Second, the corpus graph

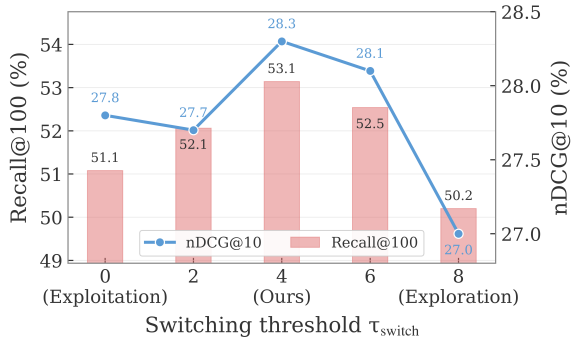

Figure 10: Performance under different switching thresholds τ_{switch} on BRIGHT.

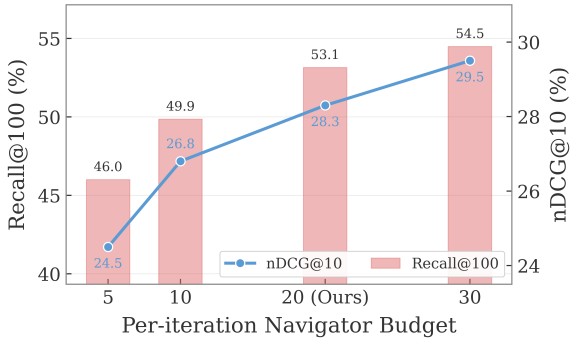

Figure 11: Performance under different per-iteration navigator budgets K_g on BRIGHT.

connects documents through surface-level term overlap rather than topical similarity, assigning the document to a group unrelated to its topic.

2. **Failures in the navigator's judgment (46.7%).** The group assignment is appropriate and the summary contains relevant information, yet the navigator assigns a low score. This is pronounced when the relevance between the query and the summary requires multi-step reasoning.

Improvement directions. The two categories suggest different directions for improvement. Failures in group construction call for better grouping methods, such as constructing groups from LLM-extracted topics of each document, or soft clustering that represents the multiple aspects of a document across different groups. For failures in the navigator's judgment, Figure 7 shows that a larger navigator improves group selection, but the latency cost outweighs the gains, making scale an inefficient solution. Training the navigator to infer the topics required by a query could be a mitigation. We leave these directions as future work.

D Hyperparameter Sensitivity

In this section, we analyze the sensitivity of GAREN to the switching threshold and the navigator budget.

Explore-then-Exploit Strategy. Figure 10 shows the effect of varying τ_{switch}, the iteration at which GAREN switches from exploration to exploitation. Both extremes underperform: $\tau_{\text{switch}} = 0$ (exploitation) commits too early to a narrow set of groups, while $\tau_{\text{switch}} = 8$ (exploration) spreads expansion across too many groups without sufficient focus. Performance peaks at $\tau_{\text{switch}} = 4$, which we use as our default, confirming that the explore-then-exploit strategy benefits from a balanced transition.

Navigator Budget. Figure 11 shows the effect of varying K_g, the maximum number of groups scored by the navigator per iteration. Performance improves with larger K_g, since scoring more groups expands the pool of expansion candidates available to GAREN. We adopt $K_g = 20$ as a balance between accuracy and computational cost; while further increases yield additional gains, they incur proportionally higher navigator cost.

E Details of the Online Retrieval Procedures

This section provides the detailed definitions used in candidate group selection. At iteration t, let W_t be the current document window, let $N_k(d)$ denote the k-NN neighbors of document d on the corpus graph, and let $\Omega(d) \in \mathcal{P}$ denote the unique group containing document d.

Navigate. This section details how the navigator scores the candidate groups $\mathcal{G}_t^{\text{cand}}$ (Section 4.2) under a per-iteration budget K_g using three rules.

1. **Reuse.** A navigator score depends only on the query and the group, not on the iteration, so we cache it in Π_q. A candidate group already scored in an earlier iteration reuses its cached score $\pi_q(C_m)$ without recomputation.

2. **Score.** At each iteration, the navigator newly scores at most K_g previously unscored candidate groups. For each candidate group C_m, a pointwise relevance model evaluates the query q against the group summary S_m. Although the model is prompted to answer with `yes` or `no`, we define $\Phi(q, S_m)$ as the softmax-normalized probability assigned to `yes` over the two response logits. The group score is computed as:

$$\pi_q(C_m) = \Phi(q, S_m). \tag{10}$$

Candidate groups are ranked in descending order of $\pi_q(C_m)$, and each computed score is cached for reuse in subsequent iterations. The navigator prompt is provided in Table 10.

3. **Budget allocation.** When more than K_g unscored candidates are available, we score the K_g groups with the highest *vote*, the number of window documents whose neighborhood intersects the group:

$$\texttt{vote}_t(C_m) = \sum_{d \in \mathcal{W}_t} \mathbb{1}[\, N_k(d) \cap C_m \neq \emptyset \,]. \tag{11}$$

When fewer than K_g are available, we fill the remaining budget by scanning the first-stage pool in rank order and adding the groups of the top-ranked uncovered documents. Groups that remain unscored under the budget are excluded from expansion in the current iteration, and may be scored in a later one.

Only groups with a cached or newly computed score are eligible for expansion in the current iteration, forming $\mathcal{G}_t^{\text{score}}$.

Window Expansion. Given the scored candidate groups $\mathcal{G}_t^{\text{score}}$, we construct the expansion set $\mathcal{X}_t$ according to the explore-then-exploit strategy. Within each visited group, we select unobserved documents in descending order of their rank in the first-stage retrieval result until the expansion budget is filled. The resulting expansion set is combined with the retained reranked documents to construct the next retrieval window.

F Group Statistics

Group construction is performed offline once for each corpus before retrieval. Table 9 reports the number and size distribution of the resulting groups, along with the time required for graph partitioning and group summary generation on each BRIGHT dataset. Graph partitioning completes within one minute for each dataset, while the reported summary generation times are measured using two NVIDIA A100 GPUs.

Analysis of Large Groups. Although we set the re-partitioning target threshold to $\bar{s} = 20$, a few groups exceed this limit. This occurs because some documents in the corpus are effectively identical (or near-identical), making them inseparable during graph partitioning. As a result, these groups remain larger than the target threshold.

Dataset	# Docs	# Groups	Group Size			Time (mins)
			Min	Max	Mean	
StackExchange						
Biology	57,359	5,183	2	58	11.1	81
Earth Science	121,249	11,485	2	22	10.6	196
Economics	50,220	4,561	2	194	11.0	64
Psychology	52,835	4,789	2	240	11.0	65
Robotics	61,961	4,861	2	671	12.7	70
Stack Overflow	107,081	8,136	2	672	13.2	149
Sustainable Living	60,792	5,540	2	208	11.0	76
Coding						
LeetCode	413,932	40,861	2	19	10.1	611
Pony	7,894	621	2	94	12.7	11
Theorem-based						
AoPS	188,002	16,558	2	22	11.4	364
TheoremQA-Q	188,002	16,558	2	22	11.4	364
TheoremQA-T	23,839	2,196	2	19	10.9	40

Table 9: Group construction statistics for GAREN on BRIGHT. Time is reported in minutes.

Scalability. We also confirmed that GAREN operates on corpora of up to 4.6M documents (DBpedia in BEIR) without modification, where the offline group construction cost grows roughly linearly with corpus size. The online cost also remains bounded regardless of corpus size, since the navigator scores at most K_g groups per iteration.

G Prompts

Navigator and Reranker Prompts. Table 10 presents the prompts used for the group-level navigator and the document-level reranker in GAREN. We adopt two reranking prompts, non-reasoning and reasoning, for two reasons. First, REPAIR guides expansion using intermediate reasoning steps and is thus applicable only under the reasoning prompt setting, which is required for a fair comparison with it. Second, the two prompts differ substantially in cost, as the reasoning prompt generates a reasoning trace at every iteration (Figure 6). Reporting both settings makes this effectiveness-efficiency trade-off explicit.

Group Summary Generation. Table 11 presents the prompt used for summary generation. Although the prompt generates multiple descriptive fields for a group, we use only the generated *concept* and *summary* as its representation for subsequent group-level navigation. Table 12 provides an illustrative example of a generated group representation.

Reranker (Reasoning)	
System prompt	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	[Query] {query} [Documents] {docs}
Reranker (Non-reasoning)	
System prompt	You are RankLLM, an intelligent assistant that can rank documents based on their relevancy to the query.
User prompt	I will provide you with passages, each indicated by number identifier []. Rank the passages based on their relevance to the search query. {docs} Search Query: {query}. Rank the {n_docs} passages above based on their relevance to the search query. The passages should be listed in descending order using identifiers. The most relevant passages should be listed first. The output format should be <answer> [] > [] </answer>, e.g., <answer> [1] > [2] </answer>.
Navigator	
System prompt	Judge whether the Document meets the requirements based on the Query and the Instruct provided. Note that the answer can only be "yes" or "no".
User prompt	<Instruct>: Given a query, retrieve relevant passages that answer the query. <Query>: {query} <Document>: {doc}

Table 10: Prompt templates used for the reranker and navigator.

Summarizer	
System prompt	You are an expert AI analyst and summarizer. Output valid JSON only — no markdown, no prose outside the JSON.

User prompt	You are an expert AI analyst and summarizer. Your mission is to create a highly informative and "discriminative signpost" for a navigating search agent. This signpost (a summary) must guide the agent to the correct cluster of nodes to answer a user's query.
	
	You will follow a strict, step-by-step cognitive process. You must analyze the children nodes in a target parent node (the "Positive Set").
	
	## INPUTS
	
	### POSITIVE SET: Information about the target parent node to be summarized
	The following {n} documents belong to the same community detected from a corpus graph.
	
	{docs}
	
	## YOUR TASK & OUTPUT FORMAT
	
	Your entire output must be a single, valid JSON object. Inside this JSON you will follow the 3-step thinking process outlined below, populating each field as instructed.
	
	### JSON Structure and Instructions:
	
	{
	  "detailed_fingerprints": [
	  // For EACH document in the POSITIVE SET (target parent node),
	  // extract a structured object of its key, queryable facts.
	  {
	  "one_line_summary": "...", // very information-dense, concise one-line summary
	  "key_entities": ["..."], // a few key entities central to this document
	  "genre_or_category": ["..."], // a few key genres / categories
	  "name": "..." // name the document
	  }
	  ],
	  "common_theme": "...",
	  // Reason deeply about the common themes between the documents
	  // in the POSITIVE SET. Focus on the majority theme; do not force
	  // outlier items into it.
	  "concept": "...",
	  // A 1-5 word label naming this community
	  // (e.g., "Blue-eyed cats", "Structural coloration",
	  // "HF radio communication").
	  "summary": "..."
	  // Based on step 1 and step 2, write a very information-dense
	  // description of this community. Make sure to include all key
	  // entities from the majority theme. This summary must be
	  // discriminative enough for a search agent to decide whether
	  // to explore this community. Aim for 4-8 sentences covering
	  // the core topic plus all notable entities.
	}
	
	## IMPORTANT GUIDELINES
	- The summary MUST include the specific entities extracted in step 1 (key_entities). Avoid vague labels like "Various topics" or "Miscellaneous".
	- Focus on the majority theme. If some documents do not align with the main theme, down-weight them rather than forcing them into the summary.
	- The "concept" is a short label; the "summary" is the detailed description.
	- Output JSON only — no markdown code fences, no explanation text.
	
	Your Response (JSON only):

Table 11: Prompt template used for the summarizer module.

Document	
d_1 `Hand_washing_0_3`	The most commonly missed areas are the thumb, wrist, areas between the fingers, and under fingernails. Artificial nails and chipped nail polish may harbor microorganisms. There are five critical times during the day where washing hands with soap is important to reduce fecal–oral transmission of disease. . . .
d_2 `Hand_washing_0_1`	The World Health Organization recommends washing hands for **at least 20 seconds**. These include five critical times where washing hands with soap is important to reduce fecal–oral transmission. . . .
d_3 `Hand_washing_5_3`	The World Health Organization defines **"Five Moments"** for hand hygiene: before patient care, after environmental contact, after exposure to blood or body fluids, before an aseptic task, and after patient care. . . .
d_4 `Hand_washing_0_30`	Antimicrobial soaps may be desirable before surgery or in environments where antibiotic-resistant organisms are prevalent. Surgical scrubbing requires a **tap that can be operated without touching it with the hands**. . . .
d_5 `Hand_washing_1_2`	Moist hands are more easily recontaminated. Commonly missed areas include the thumb, wrist, spaces between the fingers, and areas under fingernails. . . .
Community Summary	
Concept	**WHO Hand Hygiene Protocols**
Summary	This community consolidates WHO guidance on community and clinical hand hygiene. It covers the five critical times for community handwashing, frequently missed anatomical areas, contamination risks from artificial nails and chipped polish, the clinical "Five Moments," antimicrobial soaps in surgical and high-risk settings, and touch-free faucets for surgical scrubbing.

Table 12: Example of group summarization. Five overlapping chunks d_1–d_5 from the *Hand washing* Wikipedia article are consolidated into a single community-level summary.

[1] Group construction — missing facet in summary *(biology, "why stars disappear")*	
Query	Why do stars disappear when I **look at them**? . . .
Query intent	needs that foveal cones perform poorly in **dim light**
Gold document	Cones . . . respond differently to light of different wavelengths, and the combination of their responses is responsible for colour vision. Cones function best in relatively bright light . . . as opposed to rod cells, which work better in **dim light**. . . . *(two facets: (i) colour vision, (ii) dim-light performance)*
Group summary	. . . three types of cone photoreceptors—S/M/L-cones—each expressing distinct opsins that determine their **spectral sensitivity** . . .
Why it fails	The group's topic (photoreceptors) matches the query, but the summary describes only facet (i); facet (ii), the one the query needs, is **missing from the summary**.

[2] Group construction — topical mismatch in grouping *(leetcode, "max points on a line")*	
Query	. . . return the maximum number of points that lie on the **same straight line**.
Query intent	needs a **collinearity** counting technique
Gold document	`def count_good_triplets(nums1, nums2): ...` Return the **minimum number of lines** needed to represent the line chart. . . .
Group summary	. . . **triplet-counting** problems in arrays, where the goal is to count sets of three elements satisfying combinatorial, geometric, or algorithmic constraints. . . .
Why it fails	The document is grouped with triplet problems by a **surface cue**—its function name—so **no summary of this group** could surface collinearity.

[3] Navigator judgment *(earth science, "humidity above 100%")*	
Query	. . . what is the limit of humidity and why can it **exceed 100%**? . . .
Query intent	needs the **Kelvin equation**, which explains why vapour stays uncondensed above saturation
Gold document	The **Kelvin equation** describes the change in vapour pressure due to a curved liquid–vapor interface, such as the surface of a droplet. . . .
Group summary	The **Kelvin equation** describes how vapor pressure increases over curved liquid–vapor interfaces due to surface tension . . .
Why it fails	Document and summary both name the **Kelvin equation**, the exact knowledge the query needs, yet the group is still rejected.

Table 13: Case study on the failure modes of GAREN. Failures in group construction appear in two forms: (i) the group topic matches the query but the facet the query needs is not reflected in the summary, and (ii) the gold document is grouped along a surface cue rather than its topic. In failures of navigator judgment (iii), the summary states the required information yet the navigator does not select the group.