

NAVIRANK: LLM-Informed Latent Space Navigation Enhances Reasoning-Intensive Information Retrieval

Jinwen Pan, Haolin Jiang, Hang Liu

Rutgers, The State University of New Jersey

{jp2622, hj462, hl1097}@scarletmail.rutgers.edu

Abstract

Retrieve-then-rerank pipelines restrict large language models (LLMs) to a fixed top- k candidate set, making their reasoning ability fundamentally limited by the recall ceiling of the first-stage dense retriever. On reasoning-intensive retrieval, we make two observations: first, relevant documents missed by dense top- k retrieval are frequently reachable within a few hops on the native approximate nearest neighbor (ANN) graph; second, LLM relevance judgments can identify high-yield anchors whose neighborhoods contain these missed documents. Based on these findings, we propose **NAVIRANK**, an LLM-guided navigation framework that performs greedy expansion over the HNSW graph: it uses highly parallel pointwise LLM judgments to steer anchor selection and local expansion, followed by listwise calibration for final ranking. Experiments on BRIGHT show that NAVIRANK achieves 27.0 NDCG@10, while establishing the strongest accuracy–efficiency Pareto frontier in both token usage and latency among all evaluated methods.¹

1 Introduction

Consider a reasoning-intensive query in biological research: *"In a genetic experiment, deleting Gene A or Gene B individually has no effect on cell survival, but deleting both simultaneously results in immediate cell death. What is the term for this genetic interaction?"*

To answer this, a retrieval system must deduce the underlying abstract mechanism (e.g., Synthetic Lethality) from the described logical conditions ($A + B = \text{death}$). However, dense retrieval, reliant on embeddings and vector similarity search, is biased toward lexical or shallow semantic overlap (Karpukhin et al., 2020; Xiong et al., 2021; Reimers and Gurevych, 2019). For this query, a

¹Code is available at <https://github.com/JinwenPan/navirank>.

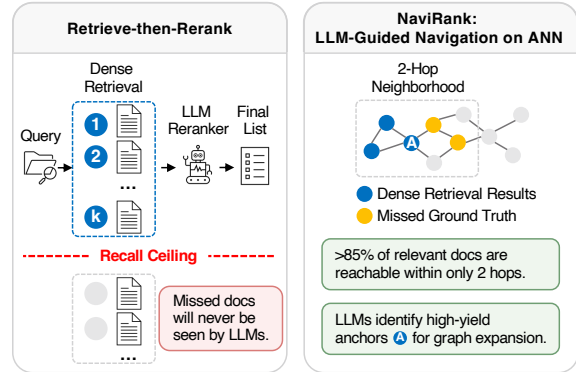


Figure 1: From static retrieve-then-rerank to LLM-guided proximity graph navigation. NAVIRANK uses LLM-selected anchors to explore short-hop neighborhoods beyond the dense top- k recall ceiling.

dense retriever typically surfaces documents narrowly focused on "cell death" or "Gene A/B." It fails to map the specific logical relationship to the theoretical document defining "Synthetic Lethality," which may not even contain the query terms.

This exposes a critical vulnerability in the widely adopted retrieve-then-rerank pipeline: the **recall ceiling** problem (Nogueira and Cho, 2020; Zamani et al., 2018; Arabzadeh et al., 2022; Cuconasu et al., 2024). Because computationally expensive large language models (LLMs) are restricted to reranking only the initial top- k candidates (typically $k = 100$), their reasoning power is fundamentally bottlenecked by the poor semantic alignment of the first-stage dense retriever (Sun et al., 2023; Ma et al., 2023). If the ground-truth evidence is ranked after top- k by the dense index, it becomes unreachable, rendering even the most sophisticated reranker useless.

Recent work has begun to mitigate this limitation by incorporating proximity graphs in dense retrieval into reranking, allowing candidate sets to expand beyond the initially retrieved top- k results (Rathee et al., 2025a; Xu and Chen, 2026;

Kim et al., 2026; V et al., 2025). However, existing graph-guided reranking methods leave two important questions underexplored. First, they largely treat graph expansion as a heuristic add-on, without systematically diagnosing whether reasoning-intensive evidence missed by dense retrieval is actually recoverable from the local topology of the proximity graph index. Second, their navigation policies often rely on expensive LLM usage or rigid expansion schedules, making it unclear how to efficiently decide where the LLM should spend its limited computation budget.

In this work, we make two empirical observations that answer these questions. First, dense retrieval often fails in ranking, but not in topology: relevant documents missed by dense top- k retrieval are frequently reachable within only one or two hops from the retrieved candidates on the proximity graph. Second, LLM relevance scores are predictive of expansion utility: high-scoring documents often serve as anchors whose neighborhoods contain missed evidence. As illustrated in Figure 1, these findings suggest a different role for both components: the approximate nearest neighbor (ANN) graph should be treated as a candidate-acquisition space, and the LLM should be used not only to rerank documents, but also to prioritize where retrieval should expand.

Motivated by this view, we propose NAVIRANK, an LLM-measured priority-queue framework that replaces fixed top- k reranking with budgeted candidate acquisition. NAVIRANK maintains an expanding queue of LLM-evaluated candidates, uses pointwise LLM relevance to prioritize HNSW neighborhood expansion, and applies listwise reranking only to the final top candidates. Experiments on BRIGHT and TREC DL19/20 show that NAVIRANK improves both recall and top-rank precision over dense retrieval, retrieve-then-rerank, and graph-guided reranking baselines, while achieving a stronger token-latency trade-off.

Our contributions are summarized as follows:

- We show that missed reasoning-intensive evidence is often locally recoverable on the ANN graph, and that LLM scores identify high-yield expansion anchors.
- We propose NAVIRANK, an LLM-measured priority-queue framework for budgeted candidate acquisition over HNSW.
- We demonstrate consistent gains in retrieval

quality and token-latency efficiency across benchmarks, embeddings, graphs, and LLM backbones.

2 Related Work

Graph-based ANN search. Graph-based ANN indices such as HNSW, NSG, and Vamana have become standard backbones for large-scale dense retrieval due to their strong recall-latency trade-offs (Malkov and Yashunin, 2020; Fu et al., 2019; Jayaram Subramanya et al., 2019; Kulkarni et al., 2023). These methods construct proximity graphs over dense embeddings and use greedy graph traversal to efficiently approximate nearest-neighbor search under a fixed geometric metric (e.g., cosine similarity). In this paradigm, the graph mainly serves as an acceleration structure for retrieving dense nearest neighbors. In contrast, NAVIRANK repurposes the native ANN graph as a candidate-acquisition space: it navigates local document neighborhoods under an LLM-induced priority measure.

LLM-based reranking. LLMs have been widely used as rerankers for information retrieval, especially through pointwise, pairwise, and listwise prompting strategies (Sun et al., 2023; Ma et al., 2023; Pradeep et al., 2023). Recent reasoning-enhanced rerankers further improve relevance estimation for complex queries by incorporating reasoning traces or task-specific fine-tuning (Yang et al., 2025c; Zhuang et al., 2025; Weller et al., 2025; Liu et al., 2026). However, these methods usually operate after candidate generation and are therefore constrained by the recall ceiling of the first-stage retriever. NAVIRANK addresses this limitation by using LLM judgments during retrieval, not only after retrieval, to maintain an expanding priority frontier over the ANN graph.

Graph-guided reranking. Recent work has explored using proximity graphs to expand candidate sets beyond the initial dense results (Kim et al., 2026; MacAvaney et al., 2022; Rathee et al., 2025b; V et al., 2025; Gupta et al., 2026; Jiang et al., 2025). For instance, SlideGAR (Rathee et al., 2025a) augments listwise reranking windows with graph neighbors, whereas RGS (Xu and Chen, 2026) greedily expands from locally high-scoring candidates. However, these methods use proximity graphs largely as heuristic expansion devices, without systematically profiling whether missed

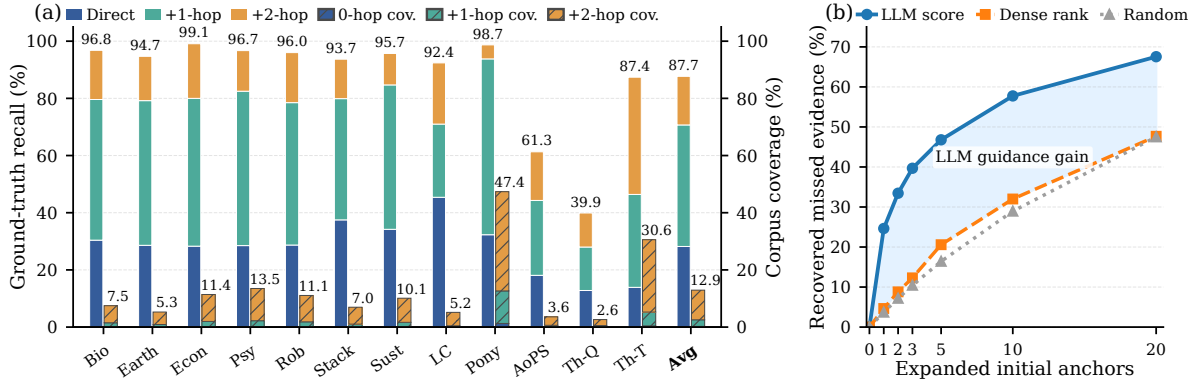


Figure 2: Two structural findings motivating NAVIRANK. (a) Missed evidence is locally recoverable: 1- and 2-hop HNSW expansion recovers much of the evidence absent from dense top-100. (b) LLM scoring identifies high-yield anchors, with the shaded area showing its recovery gain over dense ranking.

evidence is locally recoverable or which anchors provide high-yield neighborhoods. And their LLM usage during traversal is also often costly, leading to high token consumption and latency. In contrast, NAVIRANK organizes retrieval around an LLM-measured priority queue: HNSW proposes locally reachable candidates, pointwise LLM scores define their global expansion priority, and listwise reranking is delayed until the final ordering stage.

3 Preliminary Study

This section presents an empirical analysis of the reasoning-intensive BRIGHT benchmark (Su et al., 2025) to investigate the structural and semantic properties of dense retrieval. We summarize our findings into the following two key observations:

Observation #1: Dense retrieval fails in ranking, not in topology. Dense retrieval is typically judged by whether ground-truth documents appear in the top- k ranked list, but this view ignores the proximity graph already maintained by ANN indices. Motivated by the Cluster Hypothesis (Jardine and van Rijsbergen, 1971), we ask whether evidence missed by dense top- k retrieval remains close to the retrieved candidates on the native HNSW graph.

To answer this question, we conduct a local recoverability analysis on BRIGHT. For each query, we use the dense top-100 results as seed nodes and cumulatively expand their neighborhoods on the base layer of the HNSW graph. Figure 2(a) shows where ground-truth evidence lies relative to the dense top-100 seeds, distinguishing direct hits from evidence recovered through 1- and 2-hop HNSW expansion. Dense retrieval directly cap-

tures only 28.2% of the ground-truth documents on average. However, a one-hop expansion increases the average recall to 70.7%, and a two-hop expansion further raises it to 87.7%.

This observation refines the failure mode of dense retrieval on reasoning-intensive tasks. The embedding model often fails to rank the exact evidence into the fixed top- k list, but the underlying ANN topology still preserves short-range paths to much of the missed evidence. Therefore, the recall ceiling of retrieve-then-rerank is not an inherent absence of evidence from the index; it is a consequence of prematurely truncating search before local graph navigation can occur. To verify that this effect is not merely caused by candidate-pool growth, we provide recall-coverage controls against budget-matched random expansion and degree-matched random graph expansion in Appendix Figure 6.

Observation #2: LLM scoring identifies high-yield graph entry points. Observation #1 shows that missed evidence is often locally recoverable, but it also reveals that naive expansion is not a viable policy. Hard reasoning datasets such as AoPS and Th-Q require deeper traversal to recover more evidence, while in smaller corpora such as Pony, even two-hop expansion already covers nearly half of the corpus. Thus, the key challenge is not simply whether to expand the graph, but where to spend the expansion budget. This motivates the anchor-selection problem: among the dense top-100 candidates, which nodes are most likely to lead to useful unseen evidence?

To answer this question, we leverage the bi-metric framework (Xu et al., 2026), which suggests that proximity graphs built under one metric (vec-

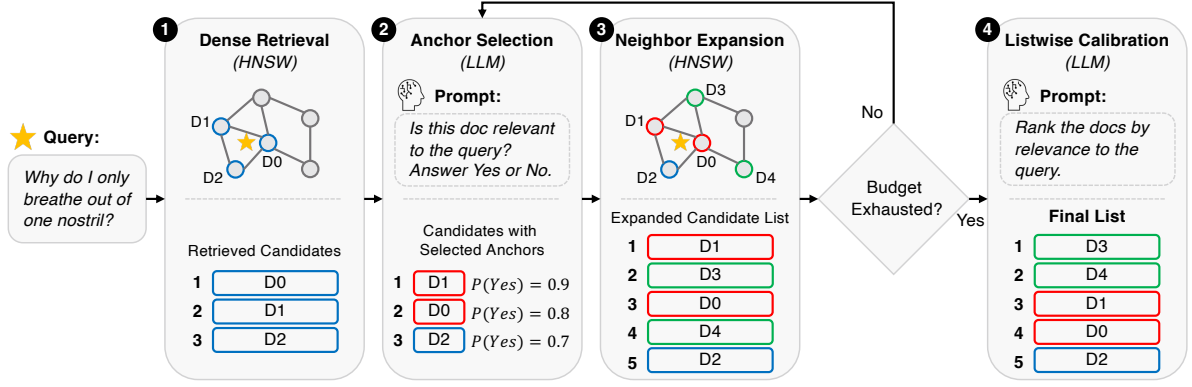


Figure 3: Design overview of NAVIRANK. The framework initializes from dense HNSW retrieval, uses pointwise LLM scores to maintain a global priority queue, select anchors for best-fit 1-hop graph expansion, and applies a final listwise reranker to calibrate the top candidates in the priority queue.

tor similarity) can be efficiently traversed to find neighbors according to a different metric (LLM relevance). We analyze whether LLM relevance scores predict graph expansion utility. We call a seed candidate an h -hop entry point if at least one ground-truth document can be reached from it within h hops on the HNSW graph. An effective navigator should therefore prioritize not only directly relevant documents, but also near-miss candidates whose neighborhoods contain hidden evidence.

Figure 2(b) presents a one-step anchor-selection diagnostic. For each query, we select the top- b anchors from the dense top-100 candidates according to pointwise LLM relevance score (prompt in Appendix C.2), dense rank, or random selection, and expand their one-hop HNSW neighborhoods. LLM-selected anchors consistently recover more missed evidence than dense-ranked or random anchors under the same anchor budget. For example, with only one expanded anchor, LLM scoring recovers 24.6% of recoverable missed evidence, compared with 4.6% for dense rank and 3.7% for random selection. This indicates that LLM scoring is predictive of neighborhood utility, rather than merely improving the order of a fixed candidate list.

We further provide a rank-promotion analysis in Appendix Table 5, showing that LLM scoring promotes direct hits as well as 1-hop and 2-hop entry points. Together, these observations motivate using LLM relevance as the priority measure for maintaining an expanding search frontier over the ANN graph.

4 Methodology

Motivated by the findings in Section 3, we propose NAVIRANK, which reframes reasoning-intensive retrieval as maintaining an LLM-measured priority queue over the native ANN graph. Unlike retrieve-then-rerank pipelines, NAVIRANK keeps the candidate set open: HNSW provides local graph transitions, while a pointwise LLM-induced measure $\mu_q(d) = S(q, d) \in [0, 1]$ determines which evaluated documents should be prioritized for expansion. In practice, $S(q, d)$ is instantiated as the probability of a “Yes” answer under a pointwise relevance judgment prompt.

LLM-measured priority queue. Given a query q , a corpus $\mathcal{D}$, and an HNSW graph $G = (\mathcal{D}, E)$ built from dense document embeddings, NAVIRANK maintains a global priority queue $\mathcal{Q}$ whose elements are pairs $(d, \mu_q(d))$. For a queue $\mathcal{Q}$, we use $\text{dom}(\mathcal{Q})$ to denote the set of stored documents. The queue is ordered by the LLM-induced measure μ_q . We also maintain an expanded-anchor set $\mathcal{A}$, which records which queued documents have already been used for neighborhood expansion. Thus, the active queue is $\text{dom}(\mathcal{Q}) \setminus \mathcal{A}$, i.e., evaluated documents whose neighborhoods remain unexplored.

The queue follows two invariants: each document is measured at most once, and expansion is non-destructive. Repeated encounters reuse the cached priority, and selected anchors remain in $\mathcal{Q}$ for final ranking. The navigation budget B limits $|\text{dom}(\mathcal{Q})|$, equivalently the number of unique pointwise LLM evaluations. Figure 3 presents the overall architecture: NAVIRANK initializes the queue from dense retrieval, expands high-priority anchors on HNSW, and applies listwise calibration

to the top queued candidates.

Queue initialization. NAVIRANK first retrieves an initial seed set $D_{\text{seed}} = \text{Top}_{k_{\text{seed}}}^{\text{dense}}(q)$ from the HNSW index based on vector similarity (❶), where $|D_{\text{seed}}| = k_{\text{seed}} = 50$ by default. These seeds are not treated as a closed candidate set, but as entry points into the graph. Each seed document is measured by a pointwise LLM evaluator $\mathcal{M}_{pt}$ (❷), yielding the initial queue $\mathcal{Q}_0 = \{(d, \mu_q(d)) \mid d \in D_{\text{seed}}\}$ and $\mathcal{A}_0 = \emptyset$. Since each query–document pair is scored independently, this step is naturally parallelizable through batching.

Queue-maintained graph navigation. At iteration t , NAVIRANK extracts the highest-priority documents that have not yet been expanded:

$$\Delta_t = \text{Top}_m\{d \in \text{dom}(\mathcal{Q}_t) \setminus \mathcal{A}_t\},$$

where candidates are ordered by $\mu_q(d)$ and $m = 1$ by default (❸). The selected documents Δ_t serve as anchors for local graph exploration (❹). For each anchor $a \in \Delta_t$, NAVIRANK reads its 1-hop HNSW neighborhood $\mathcal{N}_G(a)$ and obtains the newly discovered candidates

$$U_t = \left(\bigcup_{a \in \Delta_t} \mathcal{N}_G(a) \right) \setminus \text{dom}(\mathcal{Q}_t).$$

The documents in U_t are measured in parallel and inserted into the queue:

$$\begin{aligned} \mathcal{Q}_{t+1} &= \mathcal{Q}_t \cup \{(d, \mu_q(d)) : d \in U_t\}, \\ \mathcal{A}_{t+1} &= \mathcal{A}_t \cup \Delta_t. \end{aligned}$$

Navigation stops when the budget is exhausted or no unexpanded document remains in the queue.

Delayed listwise ordering. The pointwise stage is used for candidate acquisition rather than final ranking. Although $\mu_q(d)$ is efficient and parallelizable, it scores documents independently and lacks the comparative context needed for fine-grained ordering. After the final iteration T , NAVIRANK selects the top- L queued documents, denoted as $C_L = \text{Top}_L \text{dom}(\mathcal{Q}_T)$ under $\mu_q(d)$, where $L = 20$ by default. A listwise reranker $\mathcal{M}_{list}$ is then applied to C_L to produce the final ranked output (❺). This delayed ordering strategy reserves expensive comparative reasoning for a small set of high-priority candidates, while scalable pointwise measurements guide graph navigation. A concrete execution trace of NAVIRANK under the default configuration and an illustrative case study are provided in Appendix B.

5 Experiments

5.1 Experimental setting

Datasets. We evaluate NAVIRANK on both reasoning-intensive and standard ad-hoc retrieval benchmarks. Our primary benchmark is BRIGHT (Su et al., 2025), which is designed to test retrieval under complex information needs where relevance often depends on reasoning beyond surface lexical or semantic matching. BRIGHT consists of 12 datasets spanning three domain groups: general knowledge, code, and math. To further assess whether our method generalizes beyond reasoning-intensive settings, we also evaluate on TREC DL19 and DL20, two widely used ad-hoc web search datasets built on the MS MARCO passage corpus with graded relevance judgments (Craswell et al., 2020, 2021). Dataset statistics are provided in Appendix C.1.

Baselines. We compare against three groups of baselines. First, **dense retrieval** uses HNSW (Malkov and Yashunin, 2020) with vector cosine similarity, which also serves as the initial candidate generator for all subsequent methods. Second, **retrieve-then-rerank** methods rerank a fixed HNSW top- k list, including RankGPT (Sun et al., 2023) and reasoning-oriented rerankers Rank1/Rank-R1 (Weller et al., 2025; Zhuang et al., 2025); these methods preserve the same Recall@100 as dense retrieval. Third, **graph-guided reranking** methods expand candidates on proximity graphs, including SlideGAR (Rathee et al., 2025a) and RGS (Xu and Chen, 2026). For a consistent comparison, SlideGAR, RGS, and NAVIRANK all use the HNSW base layer as the navigation substrate.

Evaluation metrics. We evaluate retrieval quality using NDCG@10 for top-ranking precision and Recall@100 for candidate coverage. For algorithm-level alignment, we evaluate all LLM-based methods under a unified candidate-exposure cap, defined as the number of distinct documents exposed to the LLM per query. This cap matches the search scope across LLM-based methods, and is set to 100 by default.

Implementation details. We use BGE-M3 (Chen et al., 2024) to generate embeddings and build the HNSW index with Faiss (Johnson et al., 2021). RankGPT, SlideGAR, RGS, and NAVIRANK use Qwen3-30B-A3B-Instruct-2507-

Method	General Knowledge							Code		Math			Avg
	Bio	Earth	Econ	Psy	Rob	Stack	Sust	LC	Pony	AoPS	Th-Q	Th-T	
NDCG@10													
HNSW	9.1	12.3	11.9	13.0	12.0	9.9	10.4	24.7	14.9	4.8	6.4	4.3	11.1
RankGPT	24.6	33.2	26.1	31.6	22.6	26.0	28.2	22.7	21.8	4.8	6.7	9.8	21.5
Rank1	28.4	34.8	27.3	36.7	22.2	26.4	34.5	10.9	20.2	5.0	9.4	9.7	22.1
Rank-R1	22.7	36.7	28.3	36.6	18.9	27.9	35.7	20.6	22.5	4.6	7.2	14.2	23.0
SlideGAR	34.3	41.1	30.5	36.4	25.0	26.2	33.5	8.6	16.8	1.9	3.0	14.4	22.6
RGS	34.5	38.9	27.7	36.7	22.4	28.4	35.8	21.0	20.7	4.5	6.2	16.9	24.5
NAVIRANK	36.9	42.3	30.5	37.1	25.5	29.3	36.1	25.5	23.8	7.0	12.7	16.9	27.0
Recall@100													
HNSW	28.9	38.1	39.2	43.8	33.0	41.1	44.6	41.6	33.3	11.3	13.6	15.2	32.0
SlideGAR	46.1	52.3	47.9	54.1	39.3	47.8	52.8	40.6	34.9	12.7	16.1	19.4	38.7
RGS	45.7	50.3	45.9	54.9	38.7	47.6	54.0	41.4	34.7	11.8	13.3	19.9	38.2
NAVIRANK	46.8	53.0	48.7	55.8	39.3	47.9	54.9	46.2	40.8	19.0	19.2	21.3	41.0

Table 1: NDCG@10 and Recall@100 results on BRIGHT. Datasets are grouped into general knowledge, code, and math domains. Retrieve-then-rerank methods do not expand the initial candidate set from HNSW and therefore share HNSW’s Recall@100, so their Recall@100 rows are omitted.

Method	DL19	DL20	Avg	DL19	DL20	Avg
	NDCG@10			Recall@100		
HNSW	61.0	64.2	62.6	48.6	53.3	51.0
RankGPT	66.4	66.3	66.3	48.6	53.3	51.0
Rank1	64.8	67.1	66.0	48.6	53.3	51.0
Rank-R1	67.0	66.4	66.7	48.6	53.3	51.0
SlideGAR	66.2	64.5	65.3	53.2	55.5	54.3
RGS	67.4	66.0	66.7	51.8	56.3	54.0
NAVIRANK	68.7	67.8	68.2	53.4	56.4	54.9

Table 2: NDCG@10 and Recall@100 results on TREC DL19 and DL20. Retrieve-then-rerank methods preserve HNSW’s Recall@100 because they do not expand the initial candidate set.

FP8 (Yang et al., 2025a); Rank1 and Rank-R1 use their released 32B checkpoints. All LLMs are served with vLLM on an NVIDIA A100 80GB GPU. Prompts and hyperparameter analysis are provided in Appendix C.2 and Appendix C.3.

5.2 Main results

Reasoning-intensive retrieval. Table 1 presents the main results on BRIGHT. Overall, NAVIRANK achieves the best average performance on both metrics, reaching 27.0 NDCG@10 and 41.0 Recall@100. Compared with retrieve-then-rerank methods, NAVIRANK improves average NDCG@10 by 4.0 points over the strongest variant, Rank-R1, while also overcoming their fixed-candidate recall ceiling. Compared with graph-guided reranking methods, NAVIRANK improves average NDCG@10 from 24.5 to 27.0 and Re-

call@100 from 38.7 to 41.0, showing that its gains are not simply due to using graph neighbors, but to selecting and exploiting them more effectively.

The gains are consistent across domains. On general knowledge, NAVIRANK achieves the best or tied-best NDCG@10 on all seven datasets. On code and math, where dense retrieval and graph-guided baselines are less stable, NAVIRANK also obtains the best average NDCG@10 and Recall@100. These results indicate that LLM-measured priority-queue navigation improves both top-rank precision and candidate coverage across diverse reasoning-intensive settings.

Ad-hoc web retrieval. Table 2 reports the results on TREC DL19 and DL20. Compared with BRIGHT, these standard ad-hoc web search benchmarks start from a much stronger dense retrieval baseline: HNSW already achieves 62.6 average NDCG@10, and retrieve-then-rerank methods further improve it to around 66–67 without changing Recall@100. This leaves a smaller margin for graph-based candidate expansion. Existing graph-guided baselines improve Recall@100 over HNSW, but their ranking gains are less consistent; for example, SlideGAR increases average Recall@100 to 54.3 but obtains lower average NDCG@10 than RankGPT and Rank-R1.

Under this setting, NAVIRANK still achieves the best results on both datasets and both metrics. It reaches 68.2 average NDCG@10, improving over the strongest retrieve-then-rerank and graph-guided

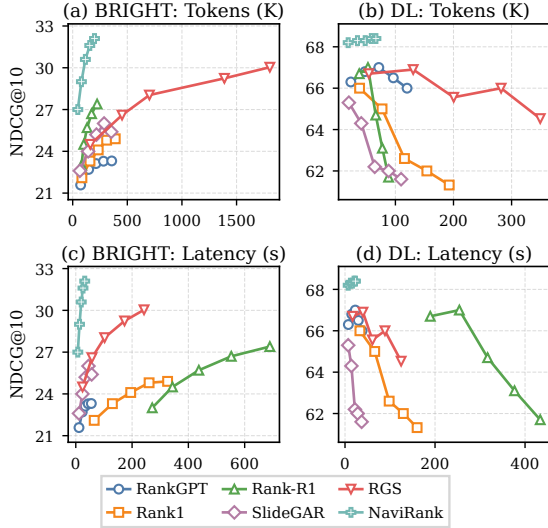


Figure 4: Accuracy–efficiency trade-off on BRIGHT and TREC DL. We plot NDCG@10 against token usage (K) and latency (s) as the number of unique LLM-evaluated documents increases from 100 to 500. Curves closer to the upper-left indicate better trade-offs.

baselines by 1.5 points, and obtains 54.9 average Recall@100, improving over the strongest graph-guided baseline by 0.6 points. The absolute gains are smaller than on BRIGHT, but they are consistent across DL19 and DL20, indicating that NAVIRANK does not rely on reasoning-intensive failure cases alone and can also improve widely used ad-hoc web retrieval benchmarks. Statistical significance tests and run-to-run variation for the main results are reported in Appendix C.4.

6 Analysis

6.1 Accuracy–efficiency trade-off

To better evaluate the accuracy–efficiency trade-off, Figure 4 reports NDCG@10 against two practical cost measures: token usage and latency per query. For each method, the curve moves from left to right as the number of unique documents evaluated by the LLM increases from 100 to 500. Overall, NAVIRANK establishes the strongest Pareto frontier on both BRIGHT and TREC DL: it consistently reaches higher accuracy with fewer tokens and lower latency. The numerical results underlying Figure 4 are reported in Appendix D.

On BRIGHT, NAVIRANK achieves the best trade-off because it improves recall beyond the fixed dense candidate set while keeping navigation lightweight. Retrieve-then-rerank methods, including reasoning-tuned Rank1 and Rank-R1, remain limited by the recall ceiling of the initial dense re-

trieval results; increasing the number of reranked documents only yields gradual gains and still underperforms NAVIRANK. Graph-guided methods can break this ceiling, but they spend substantially more LLM computation during expansion. In contrast, NAVIRANK combines highly parallel pointwise scoring for exploration with a small final listwise calibration step, and each visited document is scored only once and cached globally. As a result, NAVIRANK reaches 27.0 NDCG@10 with only 46.8K tokens and 7.2 seconds, while the strongest graph-guided baseline, RGS, requires much larger token and latency budgets to reach a comparable level. At the largest setting, NAVIRANK further improves to 32.1 NDCG@10 with 199.6K tokens and 31.7 seconds, outperforming RGS while using only a fraction of its cost.

On TREC DL, the pattern highlights scalability rather than large absolute gains. Since dense retrieval is already strong on DL19/20, exposing more top-ranked dense candidates to retrieve-then-rerank methods does not necessarily help: the added documents can introduce reranking noise and even reduce NDCG@10. Graph-guided methods face a different scaling issue. As expansion proceeds, the LLM must repeatedly maintain a limited candidate list under context constraints, increasing the chance that relevant documents are displaced by noisy graph neighbors. NAVIRANK avoids this failure mode by using pointwise LLM scores to maintain a global priority queue over all explored documents, so promising candidates are preserved rather than forgotten across iterations. Consequently, NAVIRANK remains stable around 68.2–68.4 NDCG@10 while using the lowest or among the lowest token and latency costs, whereas several baselines degrade as more documents are processed. A component ablation disentangling the contributions of pointwise scoring, graph-guided candidate acquisition, and final listwise calibration is provided in Appendix E.

6.2 Robustness analysis

We evaluate the robustness of NAVIRANK on BRIGHT by varying the embedding model and the proximity graph in the main text. Additional results across LLM backbones are reported in Appendix F. All experiments use the same candidate-exposure cap of 100 and report NDCG@10.

Impact of embedding models. Table 3 evaluates whether NAVIRANK is sensitive to the underlying

Embedding	HNSW	RankGPT	SlideGAR	RGS	Ours
Noisy Query	0.0	0.0	0.7	3.9	4.1
Contriever-0.1B	8.7	17.4	19.5	21.1	22.7
BGE-Large-0.3B	10.6	24.1	21.8	25.2	26.8
BGE-Base-0.1B	10.4	21.6	22.2	24.3	26.2
BGE-Small-33.4M	9.0	20.4	19.6	22.1	25.2

Table 3: NDCG@10 results on BRIGHT across embedding models.

embedding model (Izacard et al., 2022; Chen et al., 2024). Across Contriever and the BGE family, NAVIRANK consistently achieves the best performance, indicating that its gain does not rely on a particular embedding scale or variant.

We also include a stress-test setting, Noisy Query, where we corrupt only the query embedding by applying an element-wise random shuffle, while keeping the document embeddings and the document graph unchanged. This setting simulates query-side embedding failure: query–document proximity becomes unreliable, but document–document neighborhoods are still preserved. Although all methods degrade substantially under this severe perturbation, NAVIRANK still performs best. This suggests that LLM-guided navigation provides a limited but useful error-correction mechanism: once the search reaches any semantically useful region, the preserved document–document topology can still help redirect exploration toward relevant evidence. Additional results with alternative first-stage retrievers, including BM25 (Robertson and Zaragoza, 2009), Qwen3-Embedding-8B (Zhang et al., 2025), and ReasonIR-8B (Shao et al., 2025), are provided in Appendix G.

Impact of proximity graph structures. Table 4 compares different degree-matched graph structures (Fu et al., 2019; Jayaram Subramanya et al., 2019). On valid proximity graphs, including k NN, NSG, and Vamana, the performance differences are relatively small, and NAVIRANK consistently outperforms the baselines. This indicates that NAVIRANK does not depend on a specific ANN graph construction algorithm, as long as the graph preserves meaningful document–document neighborhoods. In contrast, the random graph destroys such document-level semantic connectivity while keeping a graph structure in place. All methods collapse under this setting, confirming that the gains of graph navigation come from the latent document–document topology rather than from expansion alone.

Graph	BGE-M3	RankGPT	SlideGAR	RGS	Ours
Random	0.1	0.3	0.3	0.1	0.3
k NN	10.8	21.5	22.8	24.6	26.2
NSG	11.9	21.4	23.8	26.3	26.9
Vamana	11.9	22.1	24.0	24.5	26.7

Table 4: NDCG@10 results on BRIGHT across graphs.

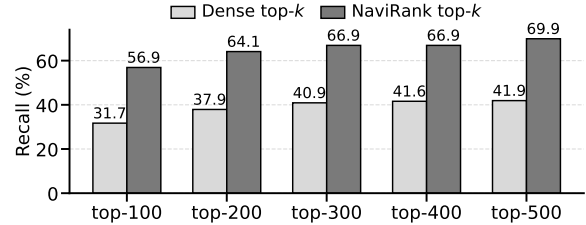


Figure 5: Candidate-pool recall comparison between query–document expansion and document–document expansion on BRIGHT.

6.3 Query–document expansion vs. document–document expansion

We further examine what type of relation is more effective for expanding the candidate pool. Retrieve-then-rerank pipelines expand candidates by increasing the dense top- k list, which relies on query–document similarity. In contrast, NAVIRANK expands candidates through document–document neighborhoods on the HNSW graph. To isolate this distinction, we use a one-seed setting on BRIGHT: for each query, we take only the top-1 dense seed as the entry point and expand from it through document–document navigation. Thus, after the initial seed is selected, all additional candidates come from graph neighborhoods. We compare this navigation pool with dense top- k retrieval under the same candidate-pool sizes, and report Recall@ k over all relevant documents across BRIGHT.

Figure 5 shows that this document–document expansion consistently recovers more relevant documents than increasing the dense top- k list, reaching 56.9 Recall@100 versus 31.7 for dense retrieval. Moreover, dense retrieval quickly saturates as the budget grows, improving only from 40.9 to 41.9 between top-300 and top-500, while graph navigation continues to expose additional relevant evidence and reaches 69.9 Recall@500. These results suggest that document–document graph navigation is more effective for expanding the candidate pool.

7 Conclusion

We propose NAVIRANK, a framework that rethinks retrieval not as a static selection but as a dynamic,

LLM-guided greedy expansion process on the latent proximity graph. By identifying that reasoning-intensive evidence often resides in the immediate latent neighborhood of retrieved candidates, we utilize the LLM as an active navigator to uncover these hidden documents that dense retrievers miss. Empirically, NAVIRANK establishes a new Pareto frontier for the accuracy–efficiency trade-off, suggesting that the latent knowledge manifold encoded by graph-based ANN indices can be effectively navigated to recover evidence beyond the dense top- k boundary.

Limitations

This work focuses on retrieval settings where the underlying proximity graph preserves meaningful document–document neighborhoods. NAVIRANK uses the ANN graph not only as an acceleration structure, but also as a candidate-acquisition space. Therefore, its effectiveness is expected to be most beneficial when local graph neighborhoods provide useful semantic connectivity. Our robustness analysis shows consistent gains across several valid proximity graph structures, but settings with highly uninformative or poorly constructed graph topology are beyond the main scope of this work.

Our evaluation focuses on retrieval quality, using standard metrics such as NDCG@10 and Recall@100 on reasoning-intensive and ad-hoc retrieval benchmarks. This design isolates the effect of LLM-guided graph navigation on candidate acquisition and ranking. We do not further evaluate how the retrieved results affect downstream applications such as retrieval-augmented generation, multi-step question answering, or agentic tool use. Studying how retrieval-side improvements translate into end-to-end task performance is an important direction for future work.

Acknowledgments

We thank the anonymous reviewers and the Area Chairs for their constructive feedback and suggestions. This work was supported in part by the U.S. National Science Foundation (NSF) under CAREER Award No. 2326141, CyberTraining Award No. 2417750, CSSI Award No. 2411294, CICI Award No. 2319880, and ExpandQISE Award No. 2328948, and by the TechAdvance Award from Rutgers University.

References

- Negar Arabzadeh, Alexandra Vtyurina, Xinyi Yan, and Charles L. A. Clarke. 2022. [Shallow pooling for sparse labels](#). *Inf. Retr.*, 25(4):365–385.
- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [M3-embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335, Bangkok, Thailand. Association for Computational Linguistics.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. [Overview of the TREC 2020 deep learning track](#). *Preprint*, arXiv:2102.07662.
- Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020. [Overview of the TREC 2019 deep learning track](#). *Preprint*, arXiv:2003.07820.
- Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. [The power of noise: Redefining retrieval for RAG systems](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24*, page 719–729, New York, NY, USA. Association for Computing Machinery.
- Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. [Fast approximate nearest neighbor search with the navigating spreading-out graph](#). *Proc. VLDB Endow.*, 12(5):461–474.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. [The Llama 3 herd of models](#). *Preprint*, arXiv:2407.21783.
- Nilesh Gupta, Wei-Cheng Chang, Ngot Bui, Cho-Jui Hsieh, and Inderjit S. Dhillon. 2026. [LLM-guided hierarchical search for end-to-end reasoning intensive retrieval](#). *Preprint*, arXiv:2510.13217.
- 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.
- 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 *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.
- Haolin Jiang, Santosh Pandey, and Hang Liu. 2025. [A comprehensive survey of subgraph matching: \[experiments & analysis\]](#). *Proc. ACM Manag. Data*, 3(6).
- Jeff Johnson, Matthijs Douze, and Herve Jegou. 2021. [Billion-scale similarity search with GPUs](#). *IEEE Transactions on Big Data*, 7(03):535–547.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Jongho Kim, Jaeyoung Kim, Jihyuk Kim, Yu Jin Kim, Seung-won Hwang, and Moontae Lee. 2026. [Adaptive retrieval for reasoning](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 37366–37381, San Diego, California, United States. Association for Computational Linguistics.
- Hrishikesh Kulkarni, Sean MacAvaney, Nazli Goharian, and Ophir Frieder. 2023. [Lexically-accelerated dense retrieval](#). In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '23*, pages 152–162, New York, NY, USA. Association for Computing Machinery.
- Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2026. [ReasonRank: Empowering passage ranking with strong reasoning ability](#). In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22007–22031, San Diego, California, United States. Association for Computational Linguistics.
- Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. [Zero-shot listwise document reranking with a large language model](#). *Preprint*, arXiv:2305.02156.
- Sean MacAvaney, Nicola Tonellotto, and Craig Macdonald. 2022. [Adaptive re-ranking with a corpus graph](#). In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, CIKM '22*, pages 1491–1500, New York, NY, USA. Association for Computing Machinery.
- Yu A. Malkov and D. A. Yashunin. 2020. [Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs](#). *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836.
- Rodrigo Nogueira and Kyunghyun Cho. 2020. [Passage re-ranking with BERT](#). *Preprint*, arXiv:1901.04085.
- Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. [RankZephyr: Effective and robust zero-shot listwise reranking is a breeze!](#) *Preprint*, arXiv:2312.02724.
- Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025a. [Guiding retrieval using LLM-based listwise rankers](#). In *Advances in Information Retrieval*, pages 230–246, Cham. Springer Nature Switzerland.
- 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, WSDM '25*, page 954–962, New York, NY, USA. Association for Computing Machinery.
- Nils Reimers and Iryna Gurevych. 2019. [Sentence-BERT: Sentence embeddings using Siamese BERT-networks](#). In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.
- Stephen Robertson and Hugo Zaragoza. 2009. [The probabilistic relevance framework: BM25 and beyond](#). *Found. Trends Inf. Retr.*, 3(4):333–389.
- 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](#). In *Second Conference on Language Modeling*.
- Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Liu Haisu, Quan Shi, Zachary S Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Serkan O Arik, Danqi Chen, and Tao Yu. 2025. [BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval](#). In *The Thirteenth International Conference on Learning Representations*.
- 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 *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14918–14937, Singapore. Association for Computational Linguistics.
- Venktesh V, Mandeep Rathee, and Avishek Anand. 2025. [SUNAR: Semantic uncertainty based neighborhood aware retrieval for complex QA](#). In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5818–5835, Albuquerque, New Mexico. Association for Computational Linguistics.
- Orion Weller, Kathryn Ricci, Eugene Yang, Andrew Yates, Dawn Lawrie, and Benjamin Van Durme. 2025.

- Rank1: Test-time compute for reranking in information retrieval. In *Second Conference on Language Modeling*.
- Shixun Wu, Jinwen Pan, Jinyang Liu, Jiannan Tian, Ziwei Qiu, Jiajun Huang, Kai Zhao, Xin Liang, Sheng Di, Zizhong Chen, and Franck Cappello. 2025. Boosting scientific error-bounded lossy compression through optimized synergistic lossy-lossless orchestration. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, SC '25*, page 2024–2037, New York, NY, USA. Association for Computing Machinery.
- Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.
- Haikexu and Tong Chen. 2026. Beyond sequential reranking: Reranker-guided search improves reasoning intensive retrieval. In *The Fourteenth International Conference on Learning Representations*.
- Haikexu, Sandeep Silwal, and Piotr Indyk. 2026. A bi-metric framework for efficient nearest neighbor search. In *Forty-third International Conference on Machine Learning*.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *Preprint*, arXiv:2505.09388.
- 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, Junyang Lin, Kai Dang, and 23 others. 2025b. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.
- Eugene Yang, Andrew Yates, Kathryn Ricci, Orion Weller, Vivek Chari, Benjamin Van Durme, and Dawn Lawrie. 2025c. Rank-K: Test-time reasoning for listwise reranking. *Preprint*, arXiv:2505.14432.
- Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik Learned-Miller, and Jaap Kamps. 2018. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM '18*, page 497–506, New York, NY, USA. Association for Computing Machinery.
- 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. *Preprint*, arXiv:2506.05176.
- Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-R1: Enhancing reasoning in LLM-based document rerankers via reinforcement learning. *Preprint*, arXiv:2503.06034.

A Additional Preliminary Study Results

This section provides supporting analyses for the two preliminary observations in Section 3. The main paper presents the core evidence in Figure 2; here we report additional controls and full per-dataset diagnostics.

Recall-coverage controls for Observation #1.

Observation #1 shows that many ground-truth documents missed by dense top-100 are recovered within one or two HNSW hops. A natural concern is that this gain may simply come from visiting more documents rather than from meaningful graph locality. To test this, we compare native HNSW expansion with two controls under the same visited-corpus budget: random-node expansion and degree-matched random-graph expansion.

As shown in Figure 6, native HNSW expansion achieves substantially higher recall than both controls at the same coverage level. The average visited corpus percentage grows from only 0.24% at 0-hop to 2.55% at 1-hop and 12.92% at 2-hop, yet the recall rises from 28.2% to 70.7% and 87.7%, respectively. This indicates that the recovered evidence is specifically enriched in the local neighborhoods of the ANN graph, rather than being obtained by random candidate-pool growth.

Rank promotion analysis for Observation #2.

Observation #2 evaluates whether LLM scores identify high-yield anchors for graph expansion. As a complementary diagnostic, we measure how LLM scoring changes the positions of different candidate types within the dense top-100 list. We call a candidate an h -hop entry point if a ground-truth document is reachable from it within h hops on the HNSW graph.

Table 5 shows that LLM scoring consistently promotes not only direct hits, but also candidates that lead to ground-truth documents through short graph paths. On average, direct hits are promoted by +25.6 positions, while 1-hop and 2-hop entry points are promoted by +44.1 and +56.0 positions, respectively. This supports the claim that LLM

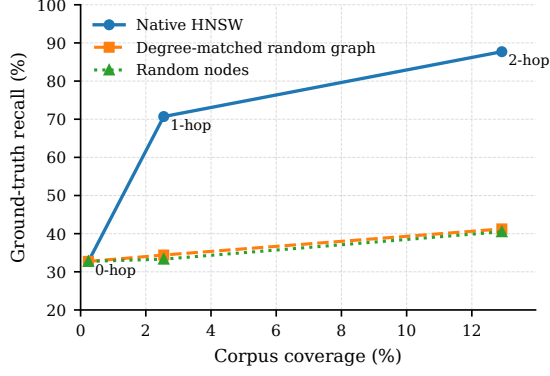


Figure 6: Recall-coverage trade-off of local graph expansion on BRIGHT. Native HNSW expansion achieves higher recall than random-node and degree-matched random-graph controls under the same visited-corpus budget, indicating that relevant evidence is enriched in local ANN neighborhoods.

relevance can surface geometrically under-ranked but semantically promising anchors.

B Concrete Execution Details and Case Study

B.1 Default navigation procedure

We provide a concrete execution trace of NAVIRANK under the default configuration to clarify how the navigation budget is consumed. For each query, NAVIRANK proceeds as follows:

1. Retrieve an initial seed set of $|D_{\text{seed}}| = 50$ documents from the HNSW index using dense similarity.
2. Evaluate each seed independently with the pointwise LLM and insert the resulting document-score pairs into the global priority queue.
3. Select the highest-scoring queued document whose neighborhood has not yet been expanded as the next anchor ($m = 1$).
4. Read the anchor’s 1-hop neighbors from the base layer of the HNSW graph.
5. Evaluate newly discovered neighbors with the pointwise LLM and insert them into the same global priority queue. Previously encountered documents reuse their cached scores and are not evaluated again.
6. Repeat anchor selection and neighborhood expansion until the navigation budget of $B =$

100 unique LLM-evaluated documents is exhausted, or no unexpanded queued document remains.

7. Select the top $L = 20$ documents from the final priority queue and apply the listwise reranker to calibrate their final ordering.

Thus, under the default candidate-exposure budget, the pointwise LLM evaluates at most 100 unique query-document pairs per query, including the initial 50 seeds and all subsequently discovered neighbors. Importantly, pointwise evaluation is performed once per unique document.

For BRIGHT, we derive the pointwise priority score from the first-token probabilities of the Yes/No judgment. Specifically,

$$\mu_q(d) = \frac{\text{mass}(\text{Yes})}{\text{mass}(\text{Yes}) + \text{mass}(\text{No})},$$

where $\text{mass}(\cdot)$ aggregates token variants after whitespace and punctuation normalization. The model is instructed to output exactly Yes or No, with temperature 0 and $\text{max_tokens} = 1$.

For TREC DL, the pointwise prompt instead asks the model to output a relevance grade in $\{0, 1, 2, 3\}$. We use the expected grade under the normalized first-token probability mass as the priority score:

$$\mu_q(d) = \sum_{r=0}^3 r \frac{\text{mass}(r)}{\sum_{j=0}^3 \text{mass}(j)}.$$

The final listwise stage follows the RankGPT-style permutation protocol described in Appendix C.2 and parses the returned document identifiers into the final ranking.

B.2 Case study: Recovering hidden evidence

We illustrate how NAVIRANK can recover evidence beyond the dense retrieval boundary using a query from the BRIGHT Biology dataset: “*Why do I only breathe out of one nostril?*” The query describes the physiological phenomenon known as the nasal cycle, but does not explicitly mention this technical term, creating a semantic mismatch between the colloquial query and relevant scientific documents.

Dense retrieval bottleneck. Table 6 shows the ranks of the six relevant documents for this query. Dense HNSW retrieval places all of them relatively low in the ranking: its best relevant result appears at Rank 24, while two relevant documents, d_1

Hop	Bio	Earth	Econ	Psy	Rob	Stack	Sust	LC	Pony	AoPS	Th-Q	Th-T	Avg
0-hop	+34.9	+37.4	+30.1	+33.5	+27.5	+24.7	+34.5	+11.4	+26.5	+7.8	+10.5	+28.5	+25.6
1-hop	+53.4	+57.3	+48.3	+55.0	+44.3	+45.1	+55.8	+22.3	+62.6	+13.5	+21.9	+50.1	+44.1
2-hop	+65.2	+68.7	+62.4	+63.4	+63.8	+53.0	+63.7	+37.8	+74.4	+27.0	+36.7	+56.3	+56.0

Table 5: Average rank promotion (+) of 0-, 1-, and 2-hop candidates within the dense top-100 list after LLM scoring across BRIGHT datasets. A h -hop candidate indicates that a ground truth document is h hops away on the proximity graph (HNSW).

and d_3 , fall entirely outside the top-100 candidate set. A conventional retrieve-then-rerank pipeline restricted to the dense top-100 therefore cannot recover these two documents.

Graph-guided recovery. NAVIRANK uses pointwise relevance judgments to prioritize graph exploration rather than treating the initial dense candidates as a closed set. During navigation, d_0 , a document defining the nasal cycle, becomes a high-priority expansion anchor. Although dense retrieval ranks d_0 only at 79, its content directly explains the phenomenon described by the query. Expanding its local HNSW neighborhood exposes d_1 and d_3 , which are absent from the dense top-100 but discuss closely related aspects of the same physiological phenomenon. These newly discovered documents are evaluated once by the pointwise LLM and inserted into the global priority queue.

After navigation, the final listwise calibration stage jointly reranks the highest-priority candidates. As shown in Table 6, NAVIRANK places all six relevant documents within the top-16, including d_1 and d_3 at Ranks 2 and 3, respectively. This example illustrates the complementary roles of the two stages: pointwise scoring guides candidate acquisition toward promising graph neighborhoods, while listwise reranking refines the ordering of the recovered evidence.

C Additional Experimental Details

C.1 Datasets

This section shows the statistics of datasets used in our experiments in Tables 7 and 8.

C.2 Prompts

This section lists the prompts used in our experiments. We denote the input query by {query}, a candidate document or passage by {document} or {passage}, and the number of candidates in a listwise reranking window by {N}. In listwise prompts, candidate passages are provided with local identifiers [1], [2], ..., [N].

BRIGHT pointwise prompt

You are a relevance judge for a reasoning-intensive retrieval task.

Given a question and a candidate document, decide whether the document is relevant to answering or solving the question.

A document is relevant if it provides information that would directly help answer or solve the question, including:

- explaining a key concept, mechanism, causal relation, constraint, assumption, definition, or reasoning step involved in the question;
- providing evidence, facts, domain knowledge, examples, or closely related cases that can be adapted to the question;
- describing an algorithm, data structure, programming pattern, API behavior, implementation detail, bug, edge case, or complexity issue needed for a coding-style question;
- providing a theorem, lemma, identity, formula, proof technique, worked example, intermediate result, or prerequisite mathematical fact needed for a mathematical question.

A document is not relevant if it only shares superficial keywords, notation, variable names, broad topics, or background information without helping reason toward the answer or solution.

Do not judge relevance by lexical overlap alone. Do not require the document to contain the final answer verbatim. Focus on whether the document provides a useful step, tool, constraint, or transferable argument for solving the question.

Question:
{query}

Candidate document:
{document}

Is the candidate document relevant to answering or solving the question?

Answer with exactly one word: Yes or No.

Answer:

Doc.	Content Summary	Graph Role	HNSW	NAVIRANK
d_0	Definition of alternating nasal congestion	Anchor	79	1
d_1	Historical description of nasal turgescence	Neighbor	> 100	2
d_3	Effect on detecting a broader range of smells	Neighbor	> 100	3
d_4	Distinction from pathological nasal congestion	Neighbor	57	5
d_2	Role in humidification and moisture	Neighbor	24	6
d_5	Reported effects associated with brain hemispheres	Neighbor	35	16

Table 6: Rank comparison for the BRIGHT Biology query “Why do I only breathe out of one nostril?” NAVIRANK recovers relevant documents that lie outside the dense top-100 by navigating the local HNSW neighborhood.

Metric	Bio	Earth	Econ	Psy	Rob	Stack	Sust	LC	Pony	AoPS	Th-Q	Th-T	Total
# Queries	104	117	104	102	102	118	109	143	113	112	195	77	1.4K
# Documents	57.3K	121.2K	50.2K	52.8K	61.9K	107.0K	60.7K	413.9K	7.8K	188.0K	188.K	23.8K	1.3M
Avg. # ground truth / query	3.5	5.0	7.6	6.7	5.1	4.0	5.2	1.8	19.6	4.6	3.1	1.9	5.5
Avg. query length	518.0	471.2	732.5	686.4	2.1K	1.2K	676.6	1.4K	385.5	316.8	423.4	410.2	788.4
Avg. document length	329.6	338.0	394.4	384.3	291.0	1.7K	344.2	1.1K	259.8	753.9	753.9	873.5	800.4

Table 7: Statistics of the BRIGHT datasets used in our experiments. Dataset abbreviations: Bio = Biology, Earth = Earth Science, Econ = Economics, Psy = Psychology, Rob = Robotics, Stack = StackOverflow, Sust = Sustainable Living, LC = LeetCode, AoPS = Art of Problem Solving, Th-Q = TheoremQA Questions, and Th-T = TheoremQA Theorems. The length is measured by the number of characters. Total averages are weighted by the corresponding number of queries or documents.

Metric	DL19	DL20	Total
# Queries	43	54	97
# Documents	8.8M	8.8M	8.8M
Avg. # ground truth / query	95.4	66.7	79.4
Avg. query length	32.7	33.8	33.3
Avg. document length	334.7	334.7	334.7

Table 8: Statistics of the MS MARCO TREC Deep Learning passage retrieval datasets. DL19 and DL20 share the same MS MARCO passage corpus, so the Total column reports the de-duplicated document count. The length is measured by the number of characters. Total averages are weighted by the corresponding number of queries or documents.

BRIGHT listwise reranking prompt

System:

You are an expert reranker for reasoning-intensive retrieval tasks. These queries often require multi-step reasoning, theorem/application matching, scientific or technical understanding, and careful distinction between superficially similar passages and passages that can actually help solve the problem.

User:

I will provide {N} candidate passages, each marked with an identifier like [1].

Query:

{query}

Rank the passages by how useful they are for answering or solving the query. Prefer

passages that contain directly applicable facts, definitions, equations, algorithms, proofs, examples, or domain constraints. Penalize passages that only share keywords, are too generic, discuss a different setting, or cannot support the reasoning needed for the query.

Read all passages before deciding. Do not explain yet; I will provide the passages next.

Assistant:

Understood. Please provide the passages.

User:

[1] {passage_1}

Assistant:

Received passage [1].

...

User:

[N] {passage_N}

Assistant:

Received passage [N].

User:

Query:

{query}

Now rank the {N} passages above by their usefulness for answering this reasoning-heavy query.

A passage should be ranked higher if it would help derive, justify, compute, or verify the answer, even if it does not contain the final answer verbatim. A passage

should be ranked lower if it is only topically related or lacks the specific information needed for the reasoning chain.

Return only the identifiers in descending relevance order. Use this exact format: [1] > [2] > [3]

Do not include explanations, labels, scores, or any extra text.

TREC DL pointwise prompt

You are an expert relevance judge for ad-hoc web passage retrieval.

Given a search query and a candidate passage, assign a relevance grade from 0 to 3.

Use the following criteria:

- 0 = Not relevant. The passage does not address the query.
- 1 = Related but not useful. The passage is on the same topic or shares keywords, but it does not answer the query.
- 2 = Relevant. The passage contains information that answers the query or provides strong evidence for the answer, but may be incomplete or indirect.
- 3 = Perfectly relevant. The passage directly and completely answers the query with clear supporting information.

Important:

- Judge whether the passage would satisfy the user's information need.
- Do not reward mere topical similarity.
- Do not use outside knowledge.
- Prefer 1 over 2 when the passage is only about the same topic but does not answer the query.
- Output only one digit: 0, 1, 2, or 3.

Query:
{query}

Passage:
{passage}

Relevance grade:

TREC DL listwise reranking prompt

System:

You are RankGPT, an intelligent assistant that can rank passages based on their relevancy to the query.

User:

I will provide you with {N} passages, each indicated by number identifier []. Rank the passages based on their relevance to query: {query}.

Assistant:

Okay, please provide the passages.

User:

[1] {passage_1}

Assistant:

Received passage [1].

...

User:

[N] {passage_N}

Assistant:

Received passage [N].

User:

Search Query: {query}.

Rank the {N} 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 [] > [], e.g ., [1] > [2]. Only response the ranking results, do not say any word or explain.

C.3 Hyperparameters

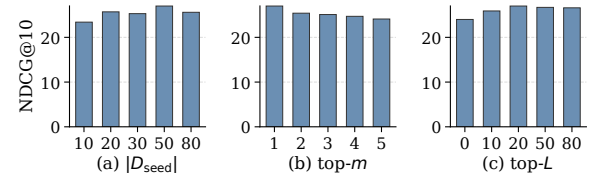


Figure 7: Hyperparameter sensitivity of NAVIRANK. We vary one hyperparameter at a time while keeping the others fixed to the default setting, and report average NDCG@10 on BRIGHT.

We analyze three key hyperparameters of NAVIRANK: the seed set size $|D_{seed}|$, the expansion breadth m , and the listwise calibration depth L . All main experiments use a single default configuration, rather than tuning these hyperparameters for individual datasets.

Seed set size $|D_{seed}|$. Figure 7(a) shows that performance improves as the seed set grows from 10 to 50, increasing NDCG@10 from 23.4 to 27.0. This suggests that a sufficiently broad initial frontier is important for exposing useful graph entry points. However, further increasing the seed set to 80 slightly degrades performance, likely because more of the candidate-exposure cap is spent on initial dense candidates, leaving less room for iterative graph navigation. We therefore set $|D_{seed}| = 50$ by default.

Expansion breadth m . Figure 7(b) shows that expanding only the top-ranked anchor per iteration performs best. As m increases from 1 to 5, NDCG@10 gradually decreases from 27.0 to 24.1. This indicates that overly broad expansion can introduce noisy neighborhoods and dilute the search budget, while a narrower expansion policy better preserves the precision of LLM-guided navigation. We therefore use $m = 1$ in the main experiments.

Listwise calibration depth L . Figure 7(c) confirms the importance of final listwise calibration. Without this step ($L = 0$), performance drops to 24.0 NDCG@10. Increasing L to 20 improves NDCG@10 to 27.0, showing that comparative reranking helps refine the top results after graph navigation. Larger values provide no further benefit and slightly reduce performance, suggesting that excessive listwise context may introduce noise. We set $L = 20$ as the default calibration depth.

C.4 Statistical significance and run-to-run variation

We further assess whether the improvements in Tables 1 and 2 are robust to query-level variation and repeated LLM execution. For BRIGHT, we use paired bootstrap tests, resampling queries within each domain and then macro-averaging the resulting domain-level scores. For TREC DL19/20, we use two-sided paired randomization tests over per-query NDCG@10 and Recall@100. For each metric, we compare NAVIRANK against the strongest baseline in the corresponding main result tables.

Table 9 summarizes the results. All reported comparisons have $p < 0.05$, indicating that the improvements of NAVIRANK are statistically significant under the corresponding paired tests. We additionally repeat all LLM-based evaluations three times. The pointwise stage is effectively deterministic under a fixed model and serving configuration because it uses first-token log probabilities with temperature 0 and max_tokens=1. The small observed run-to-run variation therefore mainly arises from listwise decoding, tie-breaking, and serving-level nondeterminism.

D Numerical Efficiency Results

Table 10 reports numerical efficiency results on BRIGHT. As commonly emphasized in GPU systems evaluation, practical effectiveness should be considered jointly with realized computational efficiency (Wu et al., 2025). We therefore report

Benchmark	Metric	Best Baseline	Result	p	Std.
BRIGHT	N@10	RGS	24.5 → 27.0	.004	.2
BRIGHT	R@100	SlideGAR	38.7 → 41.0	.011	.3
TREC DL	N@10	RGS	66.7 → 68.2	.018	.3
TREC DL	R@100	SlideGAR	54.3 → 54.9	.029	.2

Table 9: Statistical significance and run-to-run variation. Results show baseline → NAVIRANK; Std. is computed over three runs.

Method	Exp.	N@10	R@100	Tokens/q	Latency/q
RankGPT	100	21.5	32.0	72.71K	11.39s
RankGPT	300	23.1	40.8	213.20K	33.53s
RankGPT	500	23.3	41.9	357.20K	55.21s
Rank1	100	22.1	32.0	84.36K	65.44s
Rank1	300	24.1	40.8	233.68K	195.12s
Rank1	500	24.9	41.9	389.62K	326.20s
Rank-R1	100	23.0	32.0	80.73K	271.60s
Rank-R1	300	25.7	40.8	132.12K	437.15s
Rank-R1	500	27.4	41.9	223.40K	689.44s
SlideGAR	100	22.6	38.7	65.48K	11.40s
SlideGAR	300	25.2	58.1	217.33K	35.21s
SlideGAR	500	25.4	67.2	353.17K	56.40s
RGS	100	24.5	38.2	161.06K	24.06s
RGS	300	28.0	61.0	702.28K	102.43s
RGS	500	30.0	69.4	1804.60K	242.51s
NaviRank	100	27.0	41.0	46.79K	7.16s
NaviRank	300	30.6	63.9	116.36K	19.42s
NaviRank	500	32.1	71.7	199.64K	31.73s

Table 10: Numerical results for the accuracy–efficiency trade-off on BRIGHT in Figure 4. Exposure denotes the maximum number of unique documents exposed to the LLM. Token usage includes request and response tokens, and latency is measured per query under the same vLLM/A100 setup.

both retrieval quality and measured end-to-end cost under the same vLLM/A100 setup. Token usage includes both request and response tokens. For NAVIRANK, pointwise navigation uses max_tokens=1, while the final listwise calibration is applied only to the top $L = 20$ candidates. With prefix caching, the prefix is computed only once per query and reused across subsequent pointwise navigation calls.

The exposure-100 setting corresponds to the default configuration reported in Table 1, while exposure caps of 300 and 500 correspond to the higher-exposure operating points in Figure 4. Across all three exposure caps, NAVIRANK achieves the highest NDCG@10 and Recall@100 while also using the fewest tokens and incurring the lowest latency among the evaluated LLM-based methods.

In our experiments, exposure is used only to control algorithmic flow and search scope: it spec-

Variant	GEPWLWN@10R@100					Cost
A0: Dense	–	–	–	11.1	32.0	0K / 0.1s
A1: + PW	–	Y	–	19.7	32.0	43.4K / 5.9s
A2: + PW + LW	–	Y	Y	21.9	32.0	46.6K / 6.7s
A3: + Navigation	Y	Y	–	25.8	41.0	44.2K / 6.3s
A4: Full NAVIRANK	Y	Y	Y	27.0	41.0	46.8K / 7.1s

Table 11: Component ablation on BRIGHT. GE denotes graph expansion, PW pointwise scoring, and LW final listwise calibration. **Y** indicates that the corresponding component is enabled. Cost reports tokens and latency per query.

ifies the maximum number of unique documents that each method may expose to the LLM. Efficiency is evaluated independently using measured token consumption and end-to-end latency. Figure 4 therefore visualizes the resulting Pareto trade-off between retrieval effectiveness and computational cost.

E Component Ablation

We conduct a component ablation on the full BRIGHT benchmark to isolate the contributions of pointwise scoring, graph-guided candidate acquisition, and final listwise calibration. All variants use the same BGE-M3 HNSW graph, Qwen3-30B-A3B-Instruct-2507-FP8 backbone, and candidate-exposure budget $B = 100$. When enabled, pointwise scoring uses the same relevance prompt, and final listwise calibration is applied to the top $L = 20$ candidates.

The ablation reveals complementary contributions from the three components. First, pointwise scoring substantially improves the ordering of a fixed dense candidate set ($A0 \rightarrow A1$), increasing NDCG@10 from 11.1 to 19.7, but cannot improve recall because the candidate set remains unchanged. Second, graph-guided candidate acquisition provides the largest additional gain. Comparing $A1$ with $A3$, navigation increases NDCG@10 from 19.7 to 25.8 and Recall@100 from 32.0 to 41.0. The more controlled comparison $A2 \rightarrow A4$ leads to the same conclusion: both variants use identical pointwise scoring and final listwise calibration, while only NAVIRANK can acquire candidates beyond the dense top-100. Enabling graph navigation improves NDCG@10 by 5.1 points and Recall@100 by 9.0 points. Finally, $A3 \rightarrow A4$ isolates the contribution of final listwise calibration, which provides a further 1.2-point improvement in NDCG@10 without affecting candidate recall.

LLM	RankGPT	SlideGAR	RGS	Ours
Qwen2.5-1.5B-Inst.	9.1	5.7	9.4	9.8
Qwen3-4B-Inst.	18.8	14.9	19.5	21.9
Llama-3.1-8B-Inst.	14.4	13.2	14.4	20.0

Table 12: NDCG@10 results on BRIGHT across LLMs.

F Impact of LLM Backbones

Table 12 evaluates the effect of different LLM backbones from the Qwen and Llama families (Yang et al., 2025a; Grattafiori et al., 2024; Yang et al., 2025b). This analysis complements the robustness experiments in the main text by varying the LLM used for relevance estimation and reranking, while keeping the retrieval setting and candidate-exposure budget unchanged. Since NAVIRANK uses LLM scores to prioritize graph expansion, this experiment examines whether the proposed navigation strategy depends on a specific model family or model scale.

Across all tested backbones, NAVIRANK achieves the best NDCG@10 among the compared methods. Although smaller models lead to lower absolute performance for all LLM-based methods, the relative advantage of NAVIRANK remains consistent. For example, with Qwen3-4B-Instruct, NAVIRANK improves over the strongest baseline from 19.5 to 21.9, and with Llama-3.1-8B-Instruct, it improves from 14.4 to 20.0. These results suggest that the benefit of LLM-guided graph navigation comes primarily from the priority-queue search strategy rather than from a particular backbone choice. In other words, once an LLM can provide a useful relevance signal, NAVIRANK can leverage that signal to guide candidate acquisition beyond the fixed dense top- k list.

G Additional First-Stage Retrievers

We further evaluate NAVIRANK with alternative first-stage retrievers on the full BRIGHT benchmark. The main experiments use BGE-M3 to provide a consistent dense retrieval and HNSW configuration across methods. Here, we test whether the benefit of graph-guided navigation persists as the initial retrieval frontier becomes stronger.

We consider three additional settings: BM25 retrieval with the original BGE-M3 HNSW graph for subsequent navigation, Qwen3-Embedding-8B with its corresponding HNSW graph, and ReasonIR-8B with its corresponding HNSW graph. All NAVIRANK variants use the same Qwen3-

First-Stage Setting	NDCG@10	Recall@100
BGE-M3	11.1 → 27.0	32.0 → 41.0
BM25	13.5 → 27.9	34.5 → 43.8
Qwen3-Embedding-8B	19.6 → 31.6	49.3 → 52.4
ReasonIR-8B	21.2 → 33.4	52.2 → 57.1

Table 13: Results on full BRIGHT with alternative first-stage retrievers. Each metric reports fixed top-100 retrieval → retrieval with NAVIRANK. Dense settings use an HNSW graph built from the corresponding embedding model; the BM25 setting uses BM25 seeds with the BGE-M3 HNSW graph.

30B-A3B-Instruct-2507-FP8 backbone, candidate-exposure budget $B = 100$, pointwise scoring procedure, and final listwise calibration with $L = 20$ as in the main experiments.

The results show that stronger first-stage retrieval and graph-guided navigation provide complementary gains. Improving the first-stage retriever substantially raises the quality and recall of the initial top-100 frontier, while NAVIRANK consistently provides further improvements by acquiring candidates beyond this fixed truncation. For example, with ReasonIR-8B, the strongest initial retrieval setting evaluated here, NAVIRANK further improves NDCG@10 from 21.2 to 33.4 and Recall@100 from 52.2 to 57.1. The recall gain becomes smaller as the initial retriever becomes stronger, as expected from the reduced amount of relevant evidence remaining outside the top-100, but remains positive across all evaluated settings. These results indicate that NAVIRANK is not tied to a weak first-stage retriever; rather, its navigation policy complements improvements in the initial retrieval frontier.