Reproducing Adaptive Reranking for Reasoning-Intensive IR

Mandeep Rathee
L3S Research Center
Hannover, Germany
rathee@l3s.de

Venktesh V
Stockholm University
Stockholm, Sweden
venktesh.viswanathan@dsv.su.se

Sean MacAvaney
University of Glasgow
Glasgow, United Kingdom
sean.macavaney@glasgow.ac.uk

Avishek Anand
Delft University of Technology (TU Delft)
Delft, The Netherlands
avishek.anand@tudelft.nl

Abstract

The classical cascading pipeline of retrieve–rerank suffers from a bounded recall problem, stemming from limitations of the first-stage retriever. Most current approaches address the bounded recall problem by improving the first-stage retriever, but this incurs substantial training and inference costs, especially to handle queries that require substantial reasoning. To circumvent the computational costs of reasoning-based retrievers, we replicate the findings of GAR, Graph-based Adaptive Reranking, on the BRIGHT reasoning-intensive retrieval benchmark. GAR addresses the bounded recall problem by modifying the reranking process itself through iterative exploration of a corpus graph, but it was previously only tested on models designed for topical and question-answering-style queries. Hence, reproduce GAR in reasoning-intensive settings with reasoning and non-reasoning reranking models. We observe that the quality of the reranker's signal plays an important role in identifying additional relevant documents within the corpus graph. Overall, we find that GAR boosts the effectiveness of reasoning-intensive retrieval across a variety of models while contributing minimally to computational overheads. Ultimately, this work enables more practical deployment of retrieval systems that can address reasoning-intensive queries.

https://github.com/Mandeep-Rathee/gar_repro

CCS Concepts

• **Information systems** → **Retrieval models and ranking**.

Keywords

Reranking, Reasoning IR, Adaptive Retrieval

ACM Reference Format:
Mandeep Rathee, Venktesh V, Sean MacAvaney, and Avishek Anand. 2026. Reproducing Adaptive Reranking for Reasoning-Intensive IR. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '26), July 20–24, 2026, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3805712.3808568

SIGIR '26, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-2599-9/2026/07
https://doi.org/10.1145/3805712.3808568

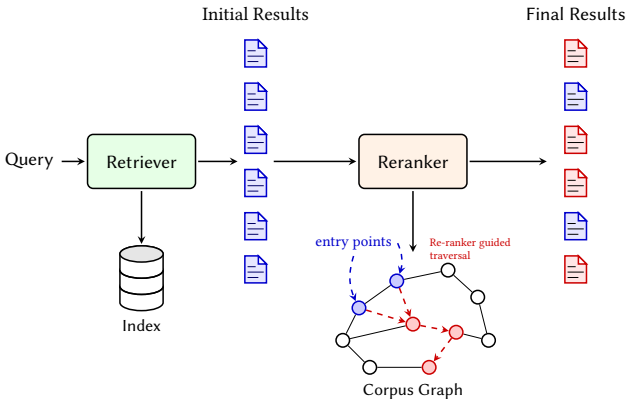

Figure 1: Overview of Graph-based Adaptive Reranking. The blue nodes represent the initial retrieved results, which can also be seen as entry points in the corpus graph. Based on the re-ranker guidance, the corpus graph is traversed. The final results list contains documents from initial results (in blue) as well as new documents from the corpus graph (in red).

1 Introduction

Ad hoc retrieval has historically focused on matching based on topic relevance [32]. The increasing capacity of relevance models (especially those powered by Large Language Models (LLMs) [22]) has pushed the field to explore more complicated retrieval tasks, such as those that require *reasoning*[1] capabilities (e.g., [5, 15, 36, 38, 43]). These benchmarking efforts have spawned a variety of new relevance models that aim address this task. Modeling efforts for reasoning-intensive retrieval have focused primarily on cross-encoders as re-ranking models (e.g., [9, 45, 48]) since they enable fine-grained interaction between the query and document. Still, sufficient recall is required from the first-stage retriever to perform well, so there have also been efforts to improve the reasoning capabilities of bi-encoders either by using a reasoning LLM as backbone (without "thinking") [9, 48] or using test-time scaling for generating reasoning traces explicitly ("thinking") [19, 44, 47, 49] before generating relevance label. In all cases, however, the trend is clear:

[1] In this work, we use BRIGHT's [36] definition of reasoning-intensive retrieval: search tasks that need more than lexical or semantic similarity to successfully address the provided information need. We acknowledge that this definition has limitations, but leave this discussion for other works.

larger models (requiring more compute) generally improve the effectiveness of reasoning-intensive retrieval.

Meanwhile, MacAvaney et al. [24] introduced a new perspective on the traditional cascading (retrieve-then-rerank) pipeline. Rather than limiting the re-ranker to only documents identified in the first stage, their approach used a lightweight feedback mechanism from the re-ranker to select which documents to score from a document similarity graph (constructed offline). An overview of this process is shown in Figure 1. This so-called *Graph-based Adaptive Re-ranking* (Gar) method consistently improved retrieval effectiveness on traditional benchmarks with minimal computational overhead. The approach has since been adapted to a variety of settings, including improving bi-encoder performance [17, 18], support for listwise rerankers [29, 46], improved graph construction mechanisms [7, 30], alternative prioritization strategies [31], question answering with retrieval augmented generation [1, 39], and a specific adaptation for reasoning-intensive retrieval [16].

The potential benefits of Gar are appealing in reasoning-intensive retrieval. If the findings of Gar hold in reasoning-intensive retrieval settings, it would buck the trend that more expensive models are needed to improve reasoning capabilities, since effectiveness could be improved with minimal additional cost. Therefore, in this work, we reproduce[2] the findings of Gar [24] to test whether they still hold in reasoning-intensive retrieval settings. We focus on three key findings from the original Gar paper: (1) its overall effectiveness, (2) its robustness across rerankers, and (3) its robustness across its hyperparameters. We confirm all three: (1) Gar is highly-effective in reasoning-intensive retrieval, (2) it works across a variety of reasoning rerankers (Qwen [48], TFRank [9], and Rank1 [45]), and (3) it is largely robust to its batch size and number of neighbor hyperparameters in the reasoning-intensive settings. Beyond these reproduction findings, we observe that Gar can overcome model size trends (e.g., TFRank-4B with Gar performs better than TFRank-8B without Gar by average nDCG@10 performance on BRIGHT). We also observe that the strongest models usually benefit the most from Gar (e.g., Qwen-4B benefits *more* with Gar than Qwen-0.6B). This result is important because it shows that findings are *composable*: improvements in relevance modeling stack upon (and actually accelerate) those from Gar.

Overall, our reproduction study findings have important implications on future directions in reasoning-intensive retrieval. We show that re-ranking "smarter" using Gar has cost advantages over "bigger" re-rankers. For instance, in our results, we observe that TFRank-1.7B augmented with Gar achieves better performance than Qwen-4B (e.g: upto **27.72%** gains in nDCG@10 on biology subset) and Rank1-7B (e.g: upto **26.51%** gains in nDCG@10 on earth sciences subset). Furthermore, the Gar and larger re-ranker strategies are composable.

2 Background and Preliminaries

Complex transformer based ranking models and, more recently instruction tuned Large Language Models (LLMs) have advanced document ranking. LLM-based rankers are highly effective in providing precise relevance estimates, but computationally expensive. More recently, reasoning-based LLMs have been adopted for reasoning-intensive IR tasks, which are further expensive and lead to high latency during inference. We contextualize our work into non-reasoning (traditional) and reasoning-based retrieval and reranking methods.

[2]Technically, *replicate* (using the ACM v1.1 definitions), since we change the experimental setup from testing traditional ad hoc retrieval to reasoning-intensive retrieval.

Query

Claim in article about why insects are attracted to light... Heat radiation as an attractive component is refuted by the effect of LED lighting... Could they for example be evolutionarily programmed to associate light with heat? So that even though they don't encounter heat near/on the LEDs they still "expect" to?

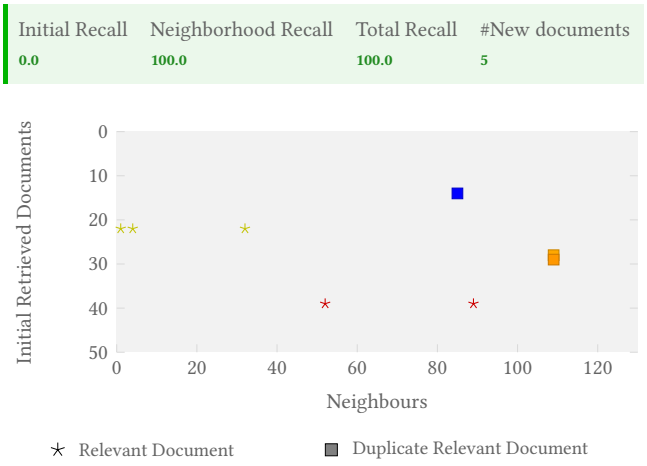

Figure 2: Qualitative example for showing the position of relevant documents in the neighborhood for a query from the Biology subset of BRIGHT. The initial retrieval BM25 fails to find any relevant document (hence Recall is 0); however, the 1-hop neighbors contain all relevant documents (5 new and 3 duplicates) and demonstrate Recall of 100%.

2.1 Traditional Retrieve-then-Rerank and Adaptive Retrieval

Traditional IR pipelines usually adopt a telescoping setting, which entails a fast but imprecise estimation of relevance using first-stage retrieval, followed by heuristic filtering of top-k documents which are then sent to a reranker for a slow but precise estimation of relevance.

Traditional lexical retrieval methods, such as BM25 [10, 33, 40], are efficient for first-stage retrieval, but the resulting relevance scores are imprecise. In modern retrieval systems, dense retrievers [14, 21], learned sparse [10, 23], and hybrid sparse-dense ensembles are used for first-stage retrieval [3, 4, 6, 41]. Meanwhile, complex re-rankers such as cross-encoders, which employ transformers, provide better relevance estimates. However, due to their computational requirements and overhead in latency, they are leveraged in the final stage of the telescoping setting [31] to rank a limited set of top-k documents filtered based on retrieval scores. However, as observed in prior works [30], this leads to a "bounded recall"

problem, where documents that are ignored based on imprecise estimation of relevance using first-stage retrieval scores are never considered for re-ranking. To recover such relevant documents, methods like Gar [24] were proposed, which adaptively retrieve documents guided by a corpus graph. Gar builds upon the *clustering hypothesis* [13], which states that documents that are close to each other help address the same query. Gar constructs a neighborhood graph comprising of top-k neighbors for each document and employs a cross-encoder to rank neighbors of already ranked documents adaptively. Kulkarni et. al. [18] explore employing a bi-encoder for efficiency instead of a cross-encoder for adaptive ranking. There are other types of further fine-grained graphs, e.g., knowledge graphs, [8] creation methods. In our study, following the original paper Gar, we stick to corpus graphs where nodes are the documents.

With advances in Large Language Models (LLMs) on a wide range of language tasks, they were also extended to document ranking tasks to obtain better relevance estimates. Several pointwise rankers like RankLLama [22] and listwise rankers like RankZephyr [28], RankGPT [37] have been proposed.

However, reasoning-intensive tasks like BRIGHT [36] require complex query and relevance understanding. Hence, they require intensive Chain-of-Thought [42] based reasoning to accurately model the semantics of the query and the documents to arrive at the right relevance estimate. We detail on some of the approaches for reasoning-intensive rankers in Section 2.2.

2.2 Reasoning-based Retrieval and Reranking

Apart from traditional retrieval tasks, more recently, much interest has been garnered around reasoning-intensive tasks [36, 38]. Rather than simple determination of similarity through semantic relatedness and surface form matching, these tasks require deliberate reasoning. For instance, a coding task may require the ranker to match based on the underlying logic required to solve the question, rather than just keyword and semantic matching. Hence, such tasks require retrievers and rankers that are capable of deeper reasoning to find accurate matches.

Hence, in recent years, with advances in LLMs and instruction tuning, retrieval systems have advanced beyond simple phrase-level or surface form matching [2, 26, 43, 45]. These models, like TART and FollowIR, usually use instruction-based data in their training data, so that they learn to adapt to new user instructions. These models understand the meaning/intent underlying the user query better than just phrase-level matching and are also capable of following instructions. Advances in decoder LLMs led to their increased adoption for document ranking tasks through Low Rank Adaptation (LoRA) for efficient fine-tuning. Hence, LLM based rankers have been adopted for pointwise [22] (RankLLaMA), pairwise [35], and listwise settings [28] (RankZephyr). While they demonstrate better query understanding and strong performance compared to existing rankers, they still fall short in reasoning-intensive tasks. Hence, more recently, reasoning-based LLMs have been adapted for retrieval and ranking tasks that require deeper reasoning.

For instance, Rank1 [45] introduces test-time compute in IR for allotting more compute for enhancing re-ranking of documents. Rank1 [45] first obtains 635,000 reasoning traces from a large reasoning model like Deepseek-R1 [11] on MS-MARCO. These reasoning traces are then used for distillation to smaller reranker LLMs. Thinking free ranker TFRank [9] further enhances the training paradigm by proposing an efficient reasoning-based ranker leveraging reasoning models. Their training recipe entails a multi-task learning strategy including fine-grained relevance scores for pointwise, listwise, and pairwise paradigms distilled from a Deepseek-R1 reasoning model. REARank [47] extends reasoning language models to listwise ranking, with explicit reasoning that is incentivized using reinforcement learning, under data-scarce scenarios with a data augmentation strategy, outperforming larger scale LLMs on reasoning-intensive document ranking tasks. Rank-R1 [49] trains an LLM as a setwise reranker using RL. It simplifies the task of finding the most relevant passage index, relying on a sparse matching-based binary reward signal. While these models have significantly improved ranking capabilities, they are still limited by the "bounded recall" problem. Hence, in this work, we analyze the effect of reproducing Gar in a reasoning-intensive setup.

2.3 A Recap of Gar

Figure 1 shows the Gar visual representation. Gar solves *bounded-recall* problem by adaptively reranking documents. For a given reranking budget c, it starts with selecting b documents from the initially retrieved documents list, denoted by R_0. These first b documents can be seen as entry points in the corpus graph, as shown in Figure 1. Then, a pointwise reranker (a cross encoder such as MonoT5) is applied to rerank these b documents, and then these documents are added to the final rank list (denoted as R_1). Then, a graph frontier (F) is prepared using the neighbors of already reranked documents. For this, the *offline* built corpus graph is traversed. In the graph frontier, each neighbor is assigned the score of the source document given by the ranker. By this, the neighbors of highly reranked documents are prioritized based on the *clustering hypothesis*, and therefore the feedback from the reranker (quantified by relevance score) becomes crucial. For the next batch, the top b documents from the graph frontier F are selected and ranked. Next, these documents are added to the final rank list R_1. Then, the graph frontier F is updated with the neighbors of these newly ranked documents. Gar keeps selecting batches from R_0 and graph frontier F alternatively until the reranking criteria is met, i.e, $|R_1| = c$. The final rank list R_1 contains documents from the initial results and the graph (as shown in Figure 1). Gar shows high effectiveness on MSMARCO-passage collection and test sets such as TREC DL19 and 20. However, these test sets consist of web queries that are short and ambiguous. In our work, we reproduce Gar on the BRIGHT benchmark, where queries are verbose, and relevance between a query and a document is defined by reasoning between them.

3 Experimental Setup

While reproducing Gar, we address the following research questions through extensive experiments:

RQ1: Is Gar's effectiveness reproducible in reasoning-intensive information retrieval task?

RQ2: What is the effect of feedback signals from different rerankers in Gar?

Table 1: BRIGHT benchmark statistics.

	Total Number			Avg. Length	
Dataset	Q	$\mathcal{D}$	$\mathcal{D}^+$	Q	$\mathcal{D}$
StackExchange					
Biology	103	57,359	3.6	115.2	83.6
Earth Science	116	121,249	5.3	109.5	132.6
Economics	103	50,220	8.0	181.5	120.2
Psychology	101	52,835	7.3	149.6	118.2
Robotics	101	61,961	5.5	818.9	121.0
Stack Overflow	117	107,081	7.0	478.3	704.7
Sustainable Living	108	60,792	5.6	148.5	107.9
Coding					
LeetCode	142	413,932	1.8	497.5	482.6
Pony	112	7,894	22.5	102.6	98.3
Theorems					
AoPS	111	188,002	4.7	117.1	250.5
TheoremQA-Q	194	188,002	3.2	93.4	250.5
TheoremQA-T	76	23,839	2.0	91.7	354.8

RQ3: Does Gar remain robust to selection of hyperparameters?

3.1 Datasets

We use the recently proposed reasoning-intensive retrieval benchmark, BRIGHT [36], for our evaluation. The benchmark consists of multiple subsets or subtaks from domains such as Stack Exchange posts, Coding, and Mathematical Theorems. For example, in Stack Exchange, each query is a post from the user, and accepted answers or webpages whose link is provided in the answer are relevant documents. Table 1 provides the statistics on these subsets or subtasks. We build a sparse PISA [25] index for each subset of the BRIGHT. To build the corpus graph, we use the sparse index and encode lexical similarities between documents, such as the BM25 score. Similarly to the original paper Gar, we treat the document as a query and add *top-k* retrieved results (given by BM25) as neighbors in the graph. This corpus graph is built *offline*, and the cost of building such graphs is described in the original paper. The original paper Gar built graphs using lexical (BM25) and semantic (TCT-ColBERT [21]) similarities. Due to limited resources, we reproduce Gar only with the BM25-based corpus graph, given its robust performance on BRIGHT. We leave other types of corpus graph formulation methods, such as semantic similarity based using dense encoders, for future work. We release all details on building indexes and corpus graphs along with our code repository.

3.2 Retrieval and Rerankers

We use BM25 [33] as initial retrieval (with k_1=0.9, b=0.4)[3] for each subset, and retrieve the top c documents exhaustively. For reranking, we choose a wide range of methods, including reasoning and non-reasoning models. We use the definition of reasoning in the Introduction section for the classification of these models. In non-reasoning, we use MonoT5 [27] and RankLLaMA [22] models. Further, we divide the reasoning models into two categories. First, those that do not "think" at test-time (e.g., no reasoning traces are generated before giving the relevance label), such as TFRank [9] (`/no think` variants) and Qwen3 [48] family rerankers. These models also follow instructions given the prompt; however, do not generate reasoning traces explicitly. For Qwen, we follow the prompts [4] used in previous works [20, 36] and models released by Qwen [48]. We use Qwen-0.6B, 4B, and 8B variants. For TFRank, we use fusion of binary judgments (*yes/no*) and fine-grained score (0-4) given by a pointwise ranking setting. We use the models based on GRPO fine-tuning on Qwen3 models and the *huggingface* weights released by the authors of TFRank. We use TFRank-0.6B, 1.7B, 4B, and 8B variants. Second, those reasoning models that "think" at test time and generate reasoning traces before generating relevance labels (`true/false`). In this category, we use Rank1 [45] family rerankers. Rank1 also follows the instruction in the prompt, where the definition of relevancy can be provided to the reasoning model. We follow the prompt for each subtask from the official repository of Rank1 [5] and use Rank1-0.5B, 1.7B, and 7B variants. Although the authors of Rank1 released 14B and 32B variants, we discard those because of limited resources and their diminishing returns on BRIGHT.

We select these rerankers on the basis of their effectiveness on web-style TREC test sets and reasoning-intensive benchmarks. We believe that these rerankers represent most of the existing rerankers.

3.3 Baselines

We compare Gar with the BM25 retriever, which is a strong baseline on the BRIGHT benchmark. We also compare the performance of Gar with the standard reranking pipeline, where BM25 initially retrieves c documents and then a neural ranker (pointwise) reranks. We measure the ranking performance by nDCG@10 and retrieval performance by Recall@c, where c is the retrieval depth or reranking budget. We also report the average performance on the BRIGHT benchmark, in line with other works.

3.4 Other Hyperparameters

We select the retrieval depth or reranking budget c to 50 and 100. As reported in the original paper, Gar is robust to the selection of hyperparameters used in the algorithm, mainly the batch size b and number of neighbors k in the graph. For our main reproducibility experiments, we set $b = 16$ and $k = 16$. We also do ablation on the variation of b and k in Section 4.2. For ablation experiments, we set the ranking budget $c = 50$ and only 3 subtasks of BRIGHT due to limited resources and computationally heavy rerankers.

4 Results and Analysis

4.1 Effectiveness of Gar on BRIGHT

To answer **RQ1** and **RQ2** (partially), we compare different reasoning-based and non-reasoning rankers in settings augmented with Gar and without Gar. The results of ranking performance are shown in

[3]We follow the BRIGHT implementation for these parameters.

[4]https://github.com/augustinLib/SPIKE/tree/main/data/embedding_instruction/gte-Qwen1.5-7B-instruct

[5]https://github.com/orionw/rank1/blob/main/prompts.py

Table 2: Ranking performance in nDCG@10 of different type of rerankers. All models rerank the top-100 retrieved documents from BM25 using the original query. w/ GAR represents the performance of the ranker when reranking is done using GAR. Results in bold represent where the GAR variant outperforms the standard reranking pipeline.

	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	
BM25	17.2	25.6	14.0	12.7	14.9	17.4	12.6	13.2	8.8	3.0	6.7	4.9	12.6
Non-Reasoning Rerankers													
MonoT5-base	9.9	16.6	7.9	7.4	4.5	6.2	10.2	6.9	7.0	2.9	8.0	3.3	7.6
w/ GAR	9.2	16.3	**8.0**	**7.8**	4.4	5.9	9.7	**7.0**	6.6	2.4	**9.8**	**4.1**	7.6
MonoT5-3B	9.4	17.2	10.3	7.4	7.5	8.8	10.6	7.7	6.3	2.7	7.8	4.9	8.4
w/ GAR	**9.5**	16.7	**11.3**	**7.5**	**8.9**	8.8	10.5	7.6	3.9	**2.9**	**9.5**	**5.2**	**8.5**
RankLLaMA-7B	17.4	29.0	9.9	14.2	15.7	7.2	16.6	8.7	23.5	1.6	4.7	3.6	12.7
w/ GAR	10.0	17.5	8.1	9.1	15.5	6.2	**16.8**	6.5	**29.8**	1.6	3.2	3.0	10.6
Reasoning but Non-Thinking Rerankers													
Qwen-0.6B	19.8	25.2	14.2	18.7	17.1	19.2	16.2	17.1	4.9	2.5	9.3	12.2	14.7
w/ GAR	**20.8**	**25.4**	**15.2**	17.5	16.4	18.4	**16.8**	16.8	**6.3**	**2.9**	**10.9**	**16.2**	**15.3**
Qwen-4B	25.7	36.4	19.2	24.4	26.7	28.4	24.1	17.9	19.9	3.3	11.4	12.4	20.8
w/ GAR	**28.5**	**37.6**	**22.3**	**25.9**	**27.4**	28.1	**26.4**	**18.3**	**20.0**	**4.0**	**16.0**	**19.0**	**22.8**
Qwen-8B	28.5	33.6	20.7	25.8	29.2	26.7	23.6	16.1	18.7	3.6	11.2	12.2	20.8
w/ GAR	**34.3**	**34.3**	**24.5**	**28.8**	**30.7**	25.7	**25.7**	**16.9**	16.3	3.6	**15.0**	**17.8**	**22.8**
TFRank-0.6B	24.1	32.9	18.4	22.1	15.4	20.2	19.4	17.6	6.6	1.8	10.0	8.9	16.5
w/ GAR	**27.5**	**34.2**	**19.6**	**26.7**	**16.3**	19.4	21.2	17.3	**8.9**	**2.0**	**11.5**	**11.1**	**18.0**
TFRank-1.7B	32.3	41.2	18.1	27.0	17.2	25.8	21.2	14.1	16.4	1.5	9.4	12.0	19.7
w/ GAR	**36.4**	**43.9**	**21.6**	**28.4**	16.9	24.6	**23.3**	**14.2**	**21.6**	**1.7**	**11.0**	**17.1**	**21.7**
TFRank-4B	36.4	44.2	21.1	29.5	22.7	27.2	27.8	19.0	25.3	1.4	9.1	12.5	23.0
w/ GAR	**45.1**	**46.8**	**23.9**	**32.6**	**22.7**	26.0	**32.1**	**20.1**	24.5	1.5	**11.2**	**18.2**	**25.4**
TFRank-8B	35.8	45.8	22.7	27.2	23.7	27.9	30.2	19.0	24.8	1.5	12.7	12.7	23.7
w/ GAR	**44.8**	**48.9**	**25.3**	**28.0**	**26.5**	27.6	**32.9**	**20.3**	24.4	**1.5**	**19.0**	**19.0**	**26.5**
Reasoning and Thinking Rerankers													
RANK1-0.5B	11.6	13.6	5.3	11.9	7.1	9.6	12.0	2.7	12.2	1.3	4.0	2.5	7.8
w/ GAR	**13.4**	9.8	**5.3**	8.7	6.0	8.4	10.3	2.2	11.6	**1.6**	**4.3**	1.9	7.0
RANK1-1.5B	20.9	18.4	8.5	18.1	11.0	10.2	14.0	5.6	11.5	1.8	5.8	6.4	11.0
w/ GAR	**24.0**	15.6	8.3	16.7	8.7	8.7	11.3	3.9	9.9	1.6	**6.4**	**7.7**	10.2
RANK1-7B	34.0	33.8	17.9	25.6	16.7	20.3	24.0	9.5	25.9	4.6	10.5	13.2	19.7
w/ GAR	**40.2**	**34.7**	**20.1**	**28.4**	**17.7**	18.9	**25.0**	8.3	24.4	4.0	**12.5**	**17.5**	**21.0**

Table 2 as indicated by nDCG@10, and the retrieval performance is shown in Figure 3 as indicated by Recall@100. We also present a detailed analysis of results for answering **RQ2** by comparing the performance of GAR when using LLM rankers of different parameter scales in Section 4.4.

4.1.1 Retrieval and Ranking performance of GAR with stronger rankers. From Table 2, we observe that both medium and large scale (in terms of parameters) reasoning-based but non-thinking rerankers like Qwen- (4B, 8B) series, TFRank series, and reasoning-based thinking rerankers like RANK1 (7B), when augmented with GAR, observe improvements in ranking performance. For instance, on the Biology collection, we observe upto **25.1%** gains in nDCG@10 when TFRank-8B is augmented with GAR compared to its ranking performance without GAR. We also observe upto **49.6%** gains on TheoremQ and TheoremT collections. This is primarily because the initial BM25 retrieved results only capture a limited number of relevant documents, limiting the ranker due to the "bounded recall" problem [30, 31]. However, GAR aids in finding more relevant documents from the neighborhood in the corpus graph with the help of relevance feedback signals from the ranker, circumventing the bounded recall issue. This is also evident by analyzing

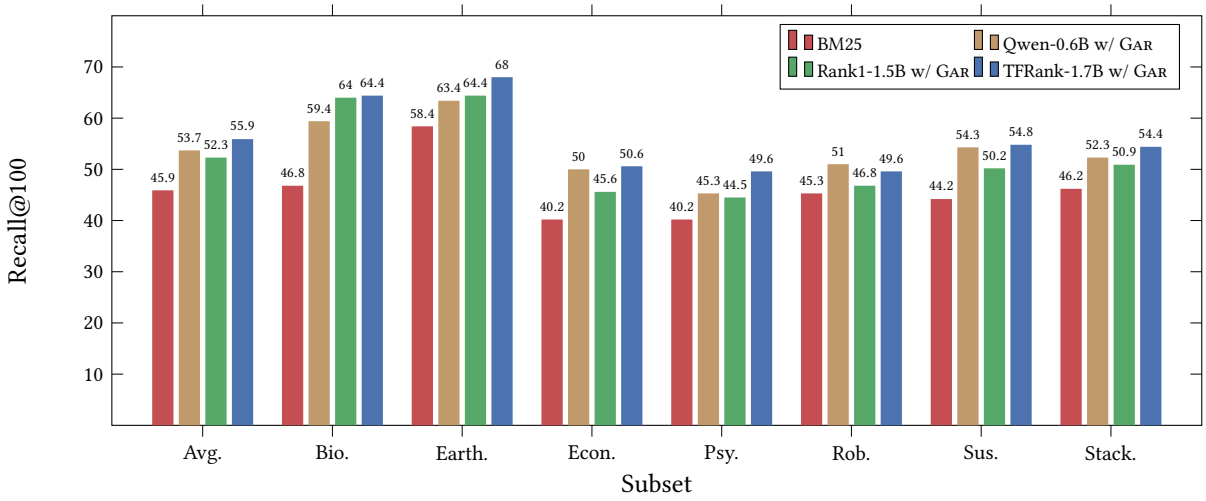

Figure 3: Retrieval performance in Recall@100 across different methods on the StackExchange subset of BRIGHT. First, BM25 retrieves the top-100 documents using the original query, then the adaptive reranking (GAR) is done using different rankers.

the retrieval performance as measured by Recall@100 in Figure 3. Due to limited space, the plots only include retrieval performance analysis from the rankers TFRank-1.7B, Qwen-0.6B, and RANK1-1.5B, though we observe similar gains for other strong rankers. We observe that when augmented with GAR Recall@100 improves upto **36.8%** (performance of RANK1-1.5B augmented with GAR, on Biology collection). We observe considerable gains in Recall@100 when employing GAR with feedback from different LLM rankers (TFRank, Qwen) as observed in Figure 3 due to its ability to capture relevant documents from the neighborhood. These additional relevant documents are then ranked higher by the ranker, resulting in higher nDCG@10 as observed in our results. Note that while GAR helps in capturing more relevant documents, it is the strength of the ranker that determines if these documents are surfaced higher up in the results.

The gains are observed across models that perform thinking by generating a detailed justification for the relevance score during inference, like RANK1, and also models that do not perform any thinking. These results demonstrate that GAR serves as a complementary module that can help enhance the ranking performance by structured traversal of the corpus graph. Hence, when the feedback signal from the ranker is strong, GAR is reproducible, offering performance gains even in a reasoning-intensive setup similar to the insights obtained in the original work [24] with marginal costs and no additional enhancements.

4.1.2 Ranking performance of GAR with weaker rankers. For weaker rerankers (like MonoT5 and RankLLaMA), augmenting with GAR does not provide any gains or actually causes a deterioration in ranking performance across all evaluation sets in BRIGHT. Note that here, MonoT5 and RankLLaMA are weaker rerankers as their nDCG@10 is lower than the BM25 baseline performance as observed in Table 2. We posit that the drop in performance is primarily due to the poor relevance signal given by the reranker, which serves as feedback to traverse the corpus graph in GAR. Due to poor signal provided by these LLM rerankers, GAR prioritizes less relevant or noisy documents from the neighborhood instead of discovering more relevant documents, resulting in poor ranking performance. We observe a similar trend with RANK1-0.5B, which also has a lower average nDCG@10 across the entire BRIGHT benchmark than the BM25 baseline. While, we observe that in the original GAR work MonoT5 based rankers lead to improved performance when augmented with GAR our results demonstrate that MonoT5 is not a strong ranker for reasoning intensive benchmark like BRIGHT. On analyzing results from the original paper, we observe that the approaches were primarily evaluated on TREC-DL test collections and MSMARCO passage ranking corpus which comprises on simpler web-based queries, and MonoT5 is an in-domain ranker trained on similar queries. These queries only require surface-level semantic matching, unlike the reasoning-intensive tasks in BRIGHT, which require the ranker to perform deeper reasoning, such as understanding the logic for code or theorem retrieval. Hence, MonoT5 variants perform poorly providing no gains or degradation of performance on such reasoning-intensive tasks, also demonstrating poor out-of-domain performance. This leads to low-quality feedback signals that prioritize noisy documents from the neighborhood graph instead of more relevant documents.

4.2 Ablation over Hyperparameters

To answer **RQ3**, we vary the different hyperparameters of GAR and report a detailed analysis of results. The pointwise rerankers can process the documents in batches, which makes them suitable for efficient reranking. Therefore, instead of reranking the batches sequentially from the initial retrieved results R_0, at each alternate iteration, it selects the batch from the graph frontier F. However, recent reasoning-based rankers, which are GPU-heavy, can not process a sufficiently large batch size. Therefore, it is important to evaluate the performance of GAR when the batch size varies from a small size to a bigger one.

The batch size b and number of neighbors k in the corpus graph play a vital role in the GAR method. As the original paper empirically

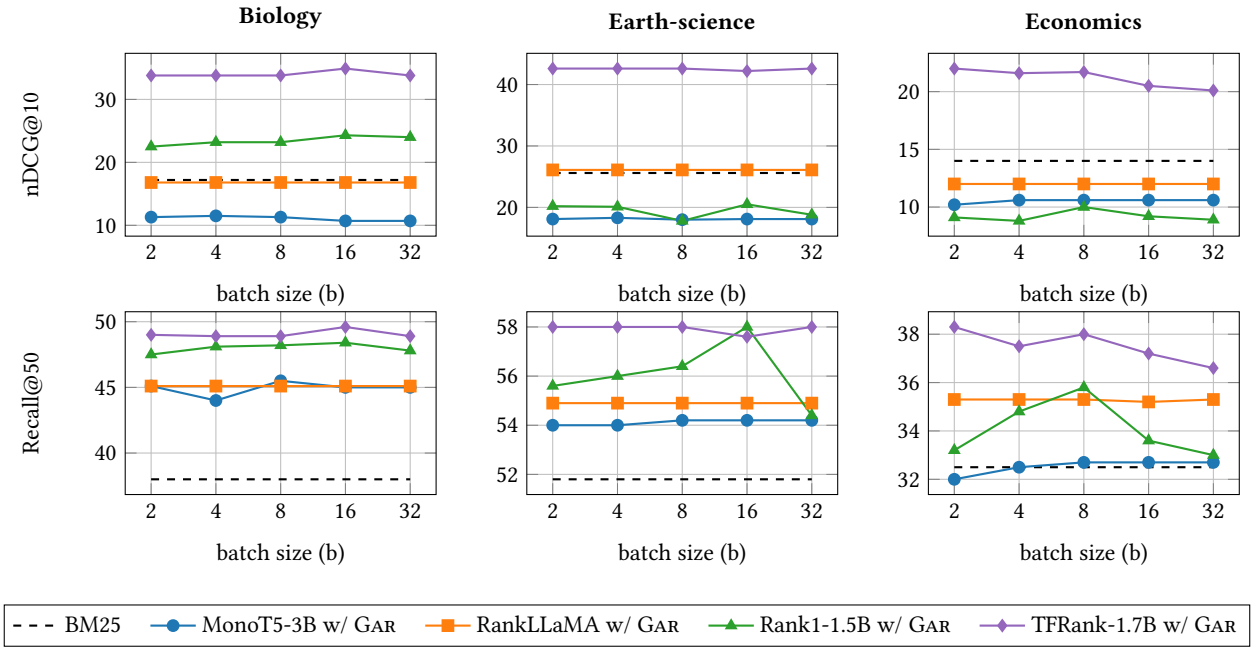

Figure 4: Retrieval and ranking performance of Gar when augmented with different re-rankers and varying batch size b. The first row shows nDCG@10 and second shows Recall@50 performance.

showed that Gar is robust to the selection of these hyperparameters. We reproduce Gar by varying batch size b and neighbors k. In the first experiment, we fix the number of neighbors k to 16 and vary $b \in [2, 4, 8, 16, 32]$. We do not go beyond a batch size of 32, since we have a ranking budget of 50. We select a budget of 50 due to limited resources, and also we observe that the Gar is effective at a lower budget, for example, 50. We have a detailed discussion in Section 4.3.

In the second experiment, we fix the batch size b to 16 and vary $k \in [2, 4, 8, 16, 32, 64, 128]$. Due to limited resources, we use MonoT5-3B, RankLLaMA-7B, Rank1-1.5B, and TFRank-1.7B as rerankers. For evaluation, we use Biology, Earth-science, and Economics subsets of BRIGHT and report nDCG@10 and Recall@50 performance. We select only three subsets of BRIGHT due to limited resources and the expensive nature of reasoning rankers.

4.2.1 Effect of Batch Size. The batch size decides how many documents should be processed in an iteration, either from the initial ranking R_0 or from the graph frontier F. In Fig. 4, we report the retrieval and ranking performance. We find that the w/ Gar pipeline remains stable with the selection of batch size b. The ranking performance (nDCG@10) remains in a ballpark over different batch sizes, which makes Gar highly effective even when resources are limited and can process smaller batches. Also, it makes it robust for 1-hop (when $b = 32$) or multi-hop (when $b \leq 16$) neighborhood traversal in the corpus graph. Though preparation and selection of batches might add some additional computational overhead, which is almost negligible compared to the computational cost of expensive rankers. On the other end, where the batch size is larger, for example $b = 16$ or 32, Gar selects only 18 documents from the neighborhood (with at most 2-hop neighbors) and remains effective. For instance on Biology, at batch size $b = 32$, Recall@50 is 48.9 and nDCG@10 is 33.8 with TFRank-1.5B is 33.8, which are significant improvements over BM25 (nDCG@10 of 17.2) and the reranking baseline (nDCG@10 of 29.6). This is similar to the original observations made in the Gar work [24] where performance of Gar is found to be stable, when the batch size is sufficiently small ($b \leq 128$).

4.2.2 Effect of Neighborhood Size. In the search system, we hope that if we explore more, the chances of finding further relevant documents are higher. Standard reranking methods focus on exploring documents in the retrieval depth; however, Gar also focus on exploring the documents that are close (neighbors in the corpus graph) to initially retrieved documents. The number of neighbors decides how many documents from a source document should be added in graph frontier F. For instance, at $k = 16$, for each reranked document, 16 neighbors are added to F and get scores of their corresponding source document. The original paper did ablation for $1 \leq k \leq 16$. We also investigate some queries manually and observe relevant documents appear in the deeper ($k > 80$) neighborhood, also shown in Figure 2. Therefore, we choose higher values of k too.

We report the performance of Gar by varying the number of neighbors in Figure 5. Most of the rerankers show a drop in Recall@50 when the number of neighbors increases, especially in the range of 16 to 128. A major drawback of the Gar method is that it can not differentiate between neighbors, since all neighbors get the same scores in the graph frontier. This is also observed in

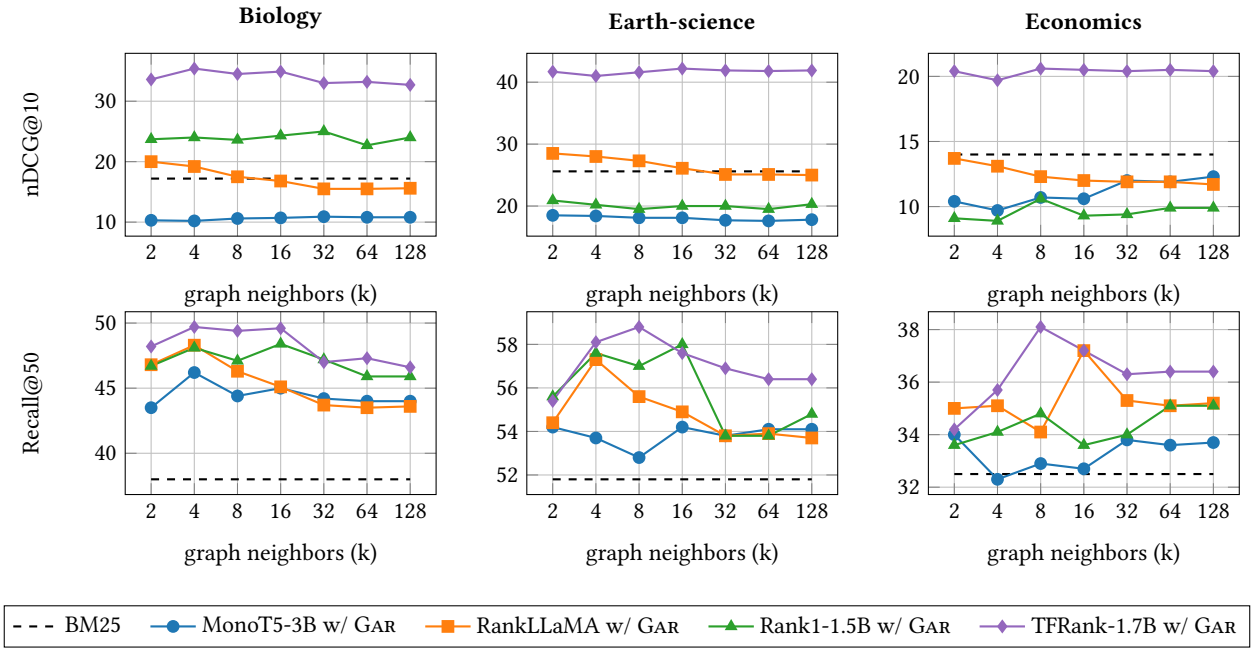

Figure 5: Effect of number of neighbors in the corpus graph. The first row shows nDCG@10 performance, and the second row Recall@50.

previous work [30]. In theory, we should find more relevant documents, however GAR can not filter out noisy neighbors. Further, these neural rankers are not *oracle* rankers, they can also rank an irrelevant document higher and then GAR end up adding all noisy neighbors in the graph frontier. On Biology, the Recall peaks in the range of 4 to 16. The most degradation happens for RankLLaMA, and its side effect can be seen in nDCG@10 performance. We also report average nDCG@10 on BRIGHT by RankLLaMA in Figure 6. We observe that the feedback signals from RankLLaMA do not help in finding relevant documents in the neighborhood. Further, we observe that GAR is most robust with TFRank-1.7B, which is a good reranker for reasoning-intensive tasks. Therefore, even if it provides high-quality relevance scores, the design of GAR algorithm forces to choose documents that are deeper into the neighborhood of only the top 1-2 documents. The neighbors of other documents might not get a chance to be selected. Even after selecting these noisy documents (as shown by the Recall@50 performance), it still ranks them lower in the final ranked list, and therefore, nDCG@10 also remains stable.

Surprisingly, the performance of MonoT5-3b either remains stable or improves further as we increase the number of neighbors. We posit this is due to its overall performance at reranking budget 50, where nDCG@10 improves (contrary to budget 100).

Overall, our experimental findings align with the original paper; the GAR performance remains stable at higher graph depth and peaks at lower depth.

4.3 Reranking Budget

The computational cost and power-hungry behavior [34] of these expensive (both reasoning and non-reasoning) rankers make it practically impossible to process a larger list of documents. Also, in the extreme case, scoring more documents exhaustively results in performance degradation due to hard negatives [12]. So, it is highly beneficial if we can have high effectiveness at lower reranking budgets. Since GAR goes beyond sequential reranking by exploring the corpus graph and using the same ranking costs, we reproduce GAR on BRIGHT when we have a small reranking budget, for instance, $c = 50$. We report[6] average nDCG@10 by different rerankers in Figure 6. We observe that GAR remain highly effective at a lower budget as well. For instance, TFRank-0.6B with GAR improves average nDCG@10 from 16.5 to 18.8 and TFRank-4B from 21.7 to 24.

More interestingly, we observe that the performance of the reranking pipeline, which reranks 100 documents, can be achieved by reranking just 50 documents (0.5x), which reduces the computational cost, latency, and carbon footprint. For instance, on Biology, nDCG@10 by TFRank-1.7B after reranking 100 documents is 32.3. On the other hand, TFRank-1.7B with GAR achieves nDCG@10 of 34.9.

Another interesting observation we make is the performance of weaker rerankers such as MonoT5-base, 3B, RankLLaMA-7B, RANK1-0.5B, or RANK1-1.5B. At a higher budget, where GAR chooses more documents from the corpus graph, due to the weak signals from these rerankers, GAR selects irrelevant documents and therefore performance degrades (as shown in Section 4.1), however, at a low ranking budget of 50, GAR selects comparatively fewer documents from the corpus graph and GAR does not degrade performance.

[6]We report subtask-level performance in the code repository.

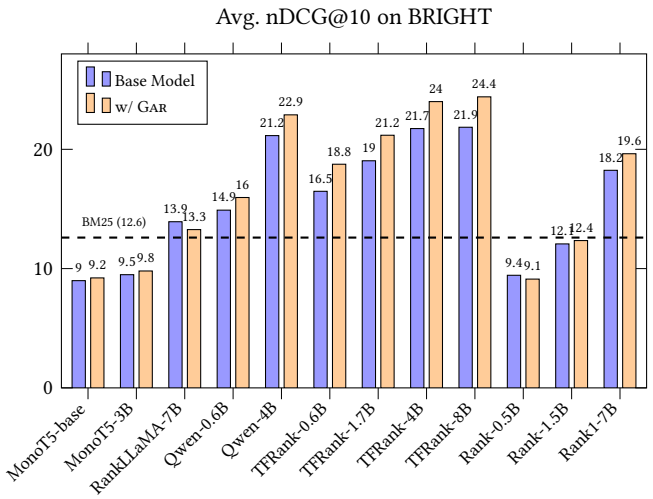

Figure 6: Average nDCG@10 comparison of various ranking models on BRIGHT when top-50 retrieved documents from BM25 are reranked. The blue bars represent the base model performance, while the orange bars show the performance with GAR augmentation. The dashed line indicates the BM25 baseline score of 12.6.

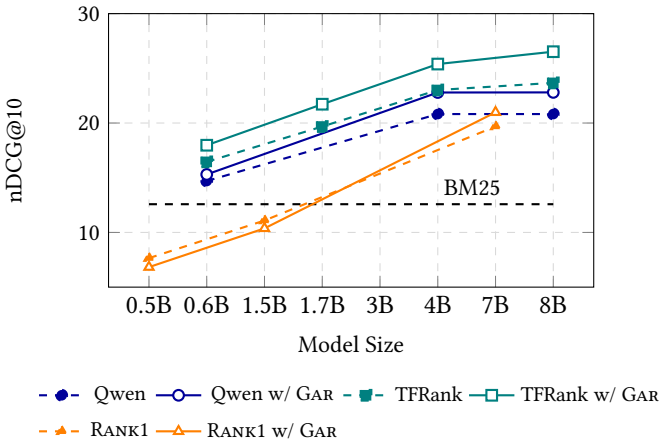

Figure 7: Performance comparison (avg. nDCG@10) on BRIGHT of reasoning models of different scales and their corresponding GAR variants.

4.4 Models Comparison

To answer **RQ2** partially, we study the behaviour of Gar with reasoning rankers, Qwen, TFRank, and Rank1 of different scales, in terms of the number of parameters. The size of the reasoning model plays a crucial role in processing the query-document pairs, especially in terms of latency and the hardware needed to run the experiments. The goal of this experiment is to see how Gar performance varies with the size of the ranking model. The key hypothesis is that performance should scale with the size of the ranking model. We report the average nDCG@10 by rerankers of different sizes in Figure 7 when the reranking budget is 100, and the original query is used for retrieval.

The first observation we make is that Gar does not show performance gains with the small scale ranker, Rank1-0.5B and Rank1-1.5B, but rather, it shows a drop in performance. We again posit that this is due to their poor performance in the ranking task, since the average nDCG@10 is worse than the BM25 baseline as discussed in Section 4.1. On the other hand, the better ranking models of similar size from Qwen and TFRank, mainly Qwen-0.6B, TFRank-0.6B, and TFRank-1.7B, have the better ranking performance than BM25. Gar shows improvements when used with such rerankers. For example, Gar improves the performance of TFRank-1.7b from 19.7 to 21.7. However, the gains for Qwen-0.6B are comparatively lower. Further, at 4B size, both Qwen-4B and TFRank-4B show performance gains when Gar is used. Further, the 7B variant of Rank1 demonstrates performance gains when augmented with Gar, as it shows stronger ranking capabilities over BM25. Gar improves nDCG@10 from 19.7 to 21. Moreover, where the performance of Qwen saturates as we go from 4B to 8B variant, the gains from Gar also saturate. Contrary to this, TFRank rankers still improve nDCG@10 from scale of 4B to 8B, and hence corresponding performance when augmented with Gar also improves. At 8B, TFRank with Gar improves nDCG@10 from 23.7 to 26.5.

Another observation we make is that the ranking performance of a larger scale model can be achieved by using a smaller scale model when used with Gar. For instance, TFRank-1.7B with Gar achieves better performance than Qwen-4B, 8B, and Rank1-7B. This demonstrates the effectiveness of Gar in resource-constrained scenarios, such as when one is limited to running a 1.7B parameter model.

Another benefit of Gar is its minimal computational overhead. As shown in the original paper, Gar adds only 3–4ms of latency per 100 documents, negligible compared to the latency of reasoning rankers. We do not perform a separate latency costs study in this work; since these overheads are independent of the specific task or ranker, the cost is driven solely by graph frontier updates and alternative batch selection. Indeed, the total cost is dominated by the reranker (especially for large reasoning and thinking tasks), and we observed no discernible difference in runtime between systems with and without Gar.

5 Conclusion

We reproduce Gar, Graph-based Adaptive Re-ranking for reasoning-intensive information retrieval benchmark, BRIGHT. We consider a variety of reasoning and non-reasoning rankers. We find that Gar helps in improving both retrieval and ranking performance on reasoning-intensive IR task. It serves as a training-free module that is complementary to other enhancements such as using a large scale ranker. However, there is strong correlation between the performance of base ranker and the performance of Gar. If the feedback signals from ranker are not weak, Gar can negatively impact the performance. Performance of Gar also depends on the other type of feedback that comes from the quality of initial retrieval stage and type of the corpus graph. We leave it for future study. Further,

we observe that GAR is robust to the selection of hyperparameters, especially when used with strong rankers. Finally, we observe that a small scale strong ranker with GAR can achieve the ranking performance of the large scale ranker.

Acknowledgments

This work was partially funded by the Bundesministerium für Wirtschaft und Energie (BMWE), Germany, in the context of the 8ra Initiative ("Soofi", 13IPC040E), and by the European Union's Horizon Europe Research and Innovation Programme JustREACH under Grant Agreement No 101214666.

References

[1] Seonho An, Chaejeong Hyun, and Min-Soo Kim. 2026. FastInsight: Fast and Insightful Retrieval via Fusion Operators for Graph RAG. *arXiv preprint arXiv:2601.18579* (2026).

[2] Akari Asai, Timo Schick, Patrick Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware Retrieval with Instructions. In *Findings of the Association for Computational Linguistics: ACL 2023*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 3650–3675. doi:10.18653/v1/2023.findings-acl.225

[3] Sebastian Bruch, Siyu Gai, and Amir Ingber. 2023. An analysis of fusion functions for hybrid retrieval. *ACM Transactions on Information Systems* 42, 1 (2023), 1–35.

[4] Tao Chen, Mingyang Zhang, Jing Lu, Michael Bendersky, and Marc Najork. 2022. Out-of-domain semantics to the rescue! zero-shot hybrid retrieval models. In *European Conference on Information Retrieval*. Springer, 95–110.

[5] Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Sahel Sharifymoghaddam, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, et al. 2025. BrowseComp-Plus: A More Fair and Transparent Evaluation Benchmark of Deep-Research Agent. In *First Workshop on Multi-Turn Interactions in Large Language Models*.

[6] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 758–759. doi:10.1145/1571941.1572114

[7] Lachlan Dunn, Luke Gallagher, and Joel Mackenzie. 2025. Approximate Bag-of-Words Top-k Corpus Graphs. In *Advances in Information Retrieval*, Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonellotto (Eds.). Springer Nature Switzerland, Cham, 174–182.

[8] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130* (2024).

[9] Yongqi Fan, Xiaoyang Chen, Dezhi Ye, Jie Liu, Haijin Liang, Jin Ma, Ben He, Yingfei Sun, and Tong Ruan. 2025. TFRank: Think-Free Reasoning Enables Practical Pointwise LLM Ranking. arXiv:2508.09539 [cs.IR] https://arxiv.org/abs/2508.09539

[10] Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Virtual Event</city>, <country>Canada</country>, </conf-loc>) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2288–2292. doi:10.1145/3404835.3463098

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948* (2025).

[12] Mathew Jacob, Erik Lindgren, Matei Zaharia, Michael Carbin, Omar Khattab, and Andrew Drozdov. 2024. Drowning in documents: consequences of scaling reranker inference. *arXiv preprint arXiv:2411.11767* (2024).

[13] N. Jardine and Cornelis Joost van Rijsbergen. 1971. The use of hierarchic clustering in information retrieval. *Inf. Storage Retr.* 7, 5 (1971), 217–240. doi:10.1016/0020-0271(71)90051-9

[14] 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)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 6769–6781. doi:10.18653/v1/2020.emnlp-main.550

[15] Julian Killingback and Hamed Zamani. 2025. Benchmarking Information Retrieval Models on Complex Retrieval Tasks. *arXiv preprint arXiv:2509.07253* (2025).

[16] Jongho Kim, Jaeyoung Kim, Seung-won Hwang, Jihyuk Kim, Yu Jin Kim, and Moontae Lee. 2026. Adaptive Retrieval for Reasoning-Intensive Retrieval. *arXiv preprint arXiv:2601.04618* (2026).

[17] Hrishikesh Kulkarni, Nazli Goharian, Ophir Frieder, and Sean MacAvaney. 2024. LexBoost: Improving Lexical Document Retrieval with Nearest Neighbors. In *Proceedings of the ACM Symposium on Document Engineering 2024* (San Jose, CA, USA) *(DocEng '24)*. Association for Computing Machinery, New York, NY, USA, Article 16, 10 pages. doi:10.1145/3685650.3685658

[18] 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 2023, Taipei, Taiwan, July 23-27, 2023*, Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.). ACM, 152–162. doi:10.1145/3539618.3591715

[19] Junwei Lan, Jianlyu Chen, Zheng Liu, Chaofan Li, Siqi Bao, and Defu Lian. 2026. Retro*: Optimizing LLMs for Reasoning-Intensive Document Retrieval. In *The Fourteenth International Conference on Learning Representations*. https://openreview.net/forum?id=0WGl8PNMSA

[20] Sangam Lee, Ryang Heo, SeongKu Kang, and Dongha Lee. 2025. Imagine All The Relevance: Scenario-Profiled Indexing with Knowledge Expansion for Dense Retrieval. *arXiv preprint arXiv:2503.23033* (2025).

[21] Sheng-Chieh Lin, Jheng-Hong Yang, and Jimmy Lin. 2021. In-Batch Negatives for Knowledge Distillation with Tightly-Coupled Teachers for Dense Retrieval. In *Proceedings of the 6th Workshop on Representation Learning for NLP, RepL4NLP@ACL-IJCNLP 2021, Online, August 6, 2021*, Anna Rogers, Iacer Calixto, Ivan Vulic, Naomi Saphra, Nora Kassner, Oana-Maria Camburu, Trapit Bansal, and Vered Shwartz (Eds.). Association for Computational Linguistics, 163–173. doi:10.18653/V1/2021.REPL4NLP-1.17

[22] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2421–2425.

[23] Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Expansion via Prediction of Importance with Contextualization. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1573–1576. doi:10.1145/3397271.3401262

[24] 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, Atlanta, GA, USA, October 17-21, 2022*, Mohammad Al Hasan and Li Xiong (Eds.). ACM, 1491–1500. doi:10.1145/3511808.3557231

[25] Antonio Mallia, Michal Siedlaczek, Joel M. Mackenzie, and Torsten Suel. 2019. PISA: Performant Indexes and Search for Academia. In *Proceedings of the Open-Source IR Replicability Challenge co-located with 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, OSIRRC@SIGIR 2019, Paris, France, July 25, 2019 (CEUR Workshop Proceedings, Vol. 2409)*, Ryan Clancy, Nicola Ferro, Claudia Hauff, Jimmy Lin, Tetsuya Sakai, and Ze Zhong Wu (Eds.). CEUR-WS.org, 50–56. https://ceur-ws.org/Vol-2409/docker08.pdf

[26] Niklas Muennighoff, Hongjin SU, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative Representational Instruction Tuning. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=BC4lIvfSzv

[27] Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document Ranking with a Pretrained Sequence-to-Sequence Model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020 (Findings of ACL, Vol. EMNLP 2020)*, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, 708–718. doi:10.18653/V1/2020.FINDINGS-EMNLP.63

[28] Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023. RankZephyr: Effective and Robust Zero-Shot Listwise Reranking is a Breeze! *arXiv preprint arXiv:2312.02724* (2023).

[29] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025. Guiding Retrieval Using LLM-Based Listwise Rankers. In *Advances in Information Retrieval*, Claudia Hauff, Craig Macdonald, Dietmar Jannach, Gabriella Kazai, Franco Maria Nardini, Fabio Pinelli, Fabrizio Silvestri, and Nicola Tonellotto (Eds.). Springer Nature Switzerland, Cham, 230–246.

[30] Mandeep Rathee, Sean MacAvaney, and Avishek Anand. 2025. Quam: Adaptive Retrieval through Query Affinity Modelling. In *Proceedings of the Eighteenth ACM International Conference on Web Search and Data Mining* (Hannover, Germany) *(WSDM '25)*. Association for Computing Machinery, New York, NY, USA, 954–962. doi:10.1145/3701551.3703584

[31] Mandeep Rathee, Venktesh V, Sean MacAvaney, and Avishek Anand. 2025. Breaking the Lens of the Telescope: Online Relevance Estimation over Large

Retrieval Sets. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Padua, Italy) *(SIGIR '25)*. Association for Computing Machinery, New York, NY, USA, 2287–2297. doi:10.1145/3726302.3729910

[32] Stephen Robertson. 2008. On the history of evaluation in IR. *J. Inf. Sci.* 34, 4 (2008), 439–456. doi:10.1177/0165551507086989

[33] Stephen Robertson and Hugo Zaragoza. 2009. The Probabilistic Relevance Framework: BM25 and Beyond. *Found. Trends Inf. Retr.* 3, 4 (apr 2009), 333–389. doi:10.1561/1500000019

[34] Harrisen Scells, Shengyao Zhuang, and Guido Zuccon. 2022. Reduce, Reuse, Recycle: Green Information Retrieval Research. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Madrid, Spain) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2825–2837. doi:10.1145/3477495.3531766

[35] Nilanjan Sinhababu, Andrew Parry, Debasis Ganguly, Debasis Samanta, and Pabitra Mitra. 2024. Few-shot Prompting for Pairwise Ranking: An Effective Non-Parametric Retrieval Model. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 12363–12377. doi:10.18653/v1/2024.findings-emnlp.720

[36] 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, Sercan 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*. https://openreview.net/forum?id=ykuc5q381b

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

[38] Nandan Thakur, Jimmy Lin, Sam Havens, Michael Carbin, Omar Khattab, and Andrew Drozdov. 2025. FreshStack: Building Realistic Benchmarks for Evaluating Retrieval on Technical Documents. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*. https://openreview.net/forum?id=54TTgXlS2U

[39] 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)*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 5818–5835. doi:10.18653/v1/2025.naacl-long.300

[40] Lidan Wang, Jimmy Lin, and Donald Metzler. 2011. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. 105–114.

[41] Shuai Wang, Shengyao Zhuang, and Guido Zuccon. 2021. Bert-based dense retrievers require interpolation with bm25 for effective passage retrieval. In *Proceedings of the 2021 ACM SIGIR international conference on theory of information retrieval*. 317–324.

[42] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 24824–24837. https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf

[43] Orion Weller, Benjamin Chang, Sean MacAvaney, Kyle Lo, Arman Cohan, Benjamin Van Durme, Dawn Lawrie, and Luca Soldaini. 2025. FollowIR: Evaluating and Teaching Information Retrieval Models to Follow Instructions. 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)*, Luis Chiruzzo, Alan Ritter, and Lu Wang (Eds.). Association for Computational Linguistics, Albuquerque, New Mexico, 11926–11942. doi:10.18653/v1/2025.naacl-long.597

[44] Orion Weller, Benjamin Van Durme, Dawn J. Lawrie, Ashwin Paranjape, Yuhao Zhang, and Jack Hessel. 2025. Promptriever: Instruction-Trained Retrievers Can Be Prompted Like Language Models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net. https://openreview.net/forum?id=odvSjn416y

[45] Orion Weller, Kathryn Ricci, Eugene Yang, Andrew Yates, Dawn Lawrie, and Benjamin Van Durme. 2025. Rank1: Test-time compute for reranking in information retrieval. *arXiv preprint arXiv:2502.18418* (2025).

[46] Soyoung Yoon, Jongho Kim, Daeyong Kwon, Avishek Anand, and Seung-won Hwang. 2025. On Listwise Reranking for Corpus Feedback. *arXiv preprint arXiv:2510.00887* (2025).

[47] Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. 2025. REARANK: Reasoning Re-ranking Agent via Reinforcement Learning. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (Eds.). Association for Computational Linguistics, Suzhou, China, 2458–2471. doi:10.18653/v1/2025.emnlp-main.125

[48] 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. *arXiv preprint arXiv:2506.05176* (2025).

[49] Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-R1: Enhancing Reasoning in LLM-based Document Rerankers via Reinforcement Learning. arXiv:2503.06034 [cs.IR] https://arxiv.org/abs/2503.06034