RaDeR: Reasoning-aware Dense Retrieval Models

Debrup Das[1], Sam O'Nuallain[1], Razieh Rahimi[1]

[1]University of Massachusetts Amherst

debrupdas@umass.edu, sonuallain@umass.edu, rahimi@cs.umass.edu

Abstract

We propose RaDeR, a set of reasoning-based dense retrieval models trained with data derived from mathematical problem solving using large language models (LLMs). Our method leverages retrieval-augmented reasoning trajectories of an LLM and self-reflective relevance evaluation, enabling the creation of both diverse and hard-negative samples for reasoning-intensive relevance. RaDeR retrievers, trained for mathematical reasoning, effectively generalize to diverse reasoning tasks in the BRIGHT and RAR-b benchmarks, consistently outperforming strong baselines in overall performance. Notably, RaDeR achieves significantly higher performance than baselines on the *Math* and *Coding* splits. In addition, RaDeR presents the first dense retriever that outperforms BM25 when queries are Chain-of-Thought reasoning steps, underscoring the critical role of reasoning-based retrieval to augment reasoning language models. Furthermore, RaDeR achieves comparable or superior performance while using only 2.5% of the training data used by the concurrent work REASONIR, highlighting the quality of our synthesized training data. Our code, data, and retrieval models are publicly available.[1]

1 Introduction

Large language models (LLMs) have demonstrated impressive reasoning capabilities on a wide range of tasks. Yet, they often benefit from retrieval augmentation to enhance accuracy, attributability (Asai et al., 2024), and the interpretability (Nakano et al., 2022) of their outputs. Retrieval models for LLM augmentation generally perform reasonably well at lexical and semantic term matching, however they face challenges when reasoning is needed for relevance prediction (Su et al., 2024).

[1]https://anonymous.4open.science/r/project-D27D/

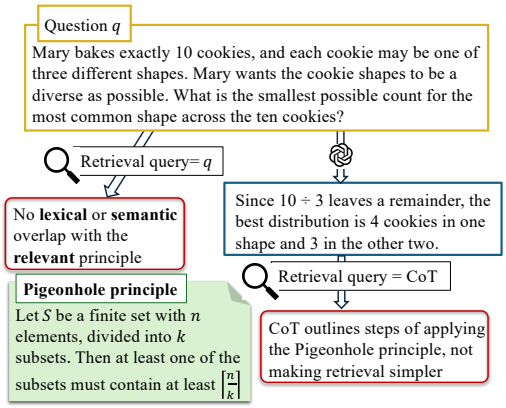

Figure 1: An example based on sample 'TheoremQA_jianyuxu/pigeonhole3' of BRIGHT, where term matching retrievers face challenges in retrieving the relevant theorem w.r.t. both questions and CoT reasoning.

Recent works have tried to address the reasoning limitation of existing models for relevance prediction. Two main approaches have emerged: (1) interleaved reasoning and retrieval (Hu et al., 2025; Jin et al., 2025; Song et al., 2025), and (2) reasoning-based re-ranking models (Weller et al., 2025; Samarinas and Zamani, 2025). While the first group is more effective than in-context retrieval augmentation, they are limited to the reasoning steps of LLMs for retrieval. As the example in Figure 1 shows, the reasoning steps of LLMs may not align with those needed for retrieval. Solving the question in this example requires retrieving the *pigeonhole principle*, where there are no matching terms between the question and the principle. The reasoning steps by GPT-4 also do not simplify the retrieval of the pigeonhole principle, since they outline the steps of applying the pigeonhole principle to solve the question. On the other hand, reranking models are inherently limited by the candidate set produced by the first-stage retriever. To the best of our knowledge, there are *no first-stage reasoning-based retrieval* models.

Developing reasoning-based retrievers poses

multiple *challenges*. The primary challenge is automatic generation of diverse and high-quality training data. Specifically, training data should include queries of *diverse formats and lengths* as well as samples with varying *degrees of reasoning complexity*. Beyond this, training retrieval models using representation learning (Dai et al., 2023) presents an additional challenge due to the need for generating hard-negative reasoning samples.

We propose RaDeR, a set of first-stage reasoning-based retrieval models trained with synthesized data from mathematical reasoning. Specifically, we use an LLM for mathematical problem solving with a retrieval-augmented search-based reasoning approach, where the LLM can retrieve and apply theorems needed for solving intermediate subproblems. To generate training data, we then sample reasoning trajectories with retrieval nodes based on the assumption that information retrieved during intermediate steps of the LLM's search process is likely to be relevant to the original question. This approach, illustrated in Figure 2, naturally yields a diverse set of queries, varying in length and complexity.

In addition, verifying LLM-generated answers to mathematical questions provides a proxy for evaluating the relevance of retrieved information. To further ensure the quality of generated training data, the relevance evaluation is enhanced by *self-reflection*. These evaluations help with generation of high-quality data. Additionally, any retrieved theorem evaluated as non-relevant is considered as a hard-negative reasoning sample.

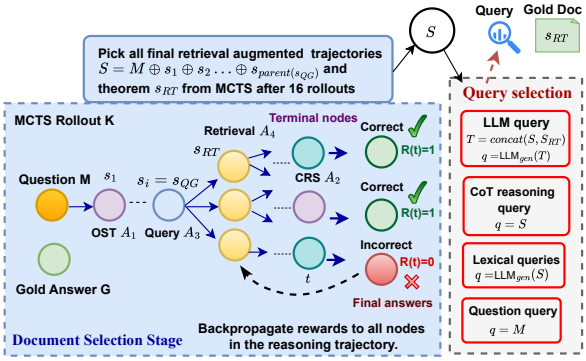

Figure 2: An overview of the **RaDeR** data generation pipeline. The *OST* action stands for *one step thought* generation, and *CRS* stands for *complete remaining solution* steps action.

We perform a comprehensive evaluation of RaDeR models, including evaluation of retrieval performance on reasoning-intensive benchmarks, traditional benchmarks mainly requiring term matching, as well as evaluation of target QA performance using retrieval augmentation. Experimental results on BRIGHT (Su et al., 2024) show that RaDeR outperforms strong baselines by at least **2** points in different settings. These findings demonstrate that training retrieval models for mathematical reasoning effectively generalizes to other types of reasoning for information retrieval. In addition to improvements in overall performance, RaDeR demonstrates particularly strong performance in *Math* and *Coding* splits of BRIGHT. We observe nDCG@10 relative improvements of **37-40%** in the theorem-Q split, and **8-26%** over the Leet coding split. RaDeR presents the *first* dense retriever that outperforms BM25 in the zero-shot setting of using reasoning steps as retrieval queries. This achievement provides strong evidence for the necessity of reasoning-based retrievers even when retrievers augment reasoning language models. On the MMTEB (Enevoldsen et al., 2025) reasoning subset, RAR-b (Xiao et al., 2024), RaDeR significantly outperforms all sparse and open-source models, performing on par with large proprietary models such as OpenAI-3-large.

In a concurrent work, Shao et al. (2025) also train reasoning-based first-stage retrieval models. RaDeR achieves a **1.1** point increase in nDCG@10 performance, corresponding to a **4.5**% relative gain. Performance of RaDeR is particularly significant given that it is trained with 43,120 samples, about **2.5**% of samples used for REASONIR (1,729,368), demonstrating the *effectiveness* of *our synthesized data*.

2 Related Works

Interleaving reasoning with retrieval. Previous works have explored interleaving chain-of-thought **(CoT) reasoning** with retrieval using *off-the-shelf retrievers* (Trivedi et al., 2023; Shao et al., 2023; Yao et al., 2023b; Schick et al., 2023) (see Appendix A.2 for details). In this paradigm, a *retrieval action* at a given step leverages a subset of the previously generated CoT reasoning steps as the *query* for retrieval. However, these methods are limited by their reliance on off-the-shelf retrievers.

Recent works like Search-R1 (Jin et al., 2025) and R1-Searcher (Song et al., 2025) use reinforcement learning to optimize **reasoning-based query generation** while keeping the retrieval model fixed. In complex retrieval tasks, direct lexical or seman-

tic overlap between the initial question and the relevant document is often limited. By incorporating intermediate CoT reasoning during retrieval, the system can better bridge this gap. Recent research explores **search-based methods** for exploring the space of CoT reasoning paths, such as random sampling (Wang et al., 2023b) and Monte Carlo Tree Search (MCTS) (Qi et al., 2025; Yao et al., 2023a; Guan et al., 2025; Hao et al., 2023). Retrieval-augmented generation (RAG) has also been integrated into MCTS-based frameworks by introducing retrieval-related *actions* such as query generation and document retrieval (Tran et al., 2024; Hu et al., 2025).

Reasoning-based re-rankers. Recent methods such as RANK-1 (Weller et al., 2025) and InteRank (Samarinas and Zamani, 2025) train re-ranking models using knowledge distillation from reasoning LLMs and reinforcement learning respectively (see Appendix A.3 for details). However, these approaches remain limited by the candidate set of initial retrievers.

Data augmentation for IR. Existing methods (Nogueira et al., 2020; Bonifacio et al., 2022; Dai et al., 2023) expand queries or documents with likely terms and generate new queries from existing documents. Recent methods (Hu et al., 2024; Lee et al., 2024) leverage LLMs to build iterative pipelines for synthetic data generation - using LLM-as-judge as the primary signal for data quality. The concurrent work ReasonIR (Shao et al., 2025) also targets reasoning-intensive search, but generates queries without labeled scalar rewards.

Our work is the first to generate synthetic datasets specifically tailored for *reasoning-intensive information retrieval*. Unlike previous methods, RaDeR directly integrates reasoning into first-stage retrieval, leveraging MCTS to create sample-efficient synthetic data.

3 RaDeR: Reasoning-aware Retrievers

We propose a framework that includes a first-stage retriever and a re-ranking model, both performing reasoning to predict relevance. For the **first-stage retriever**, we adopt a uni-embedding bi-encoder architecture of dense retrieval models (Lei et al., 2023). For **re-ranking**, we fine-tune a pointwise *cross-attention* model that takes the concatenation of the query and document. Our re-ranker directly predicts relevance scores. In contrast, existing reasoning-based re-rankers (Weller et al., 2025; Samarinas and Zamani, 2025) rely on test-time compute for reasoning, making our approach significantly more efficient at inference time.

4 Generating Retrieval Training Data

The main challenge of developing reasoning-based retrieval models is synthesizing effective data that includes queries of *diverse formats and lengths*, requiring varying degrees of reasoning complexity. This diversity is essential for adaptive RAG systems (Asai et al., 2023; Hu et al., 2025; Jin et al., 2025), where LLMs may call retrievers at any intermediate reasoning or solution step.

A widely used data augmentation technique in IR involves prompting LLMs with passages to generate relevant queries (Dai et al., 2023; Bonifacio et al., 2022; Lee et al., 2024; Hu et al., 2024). However, this technique can limit the diversity of generated queries. It also poses challenges in verifying that the generated queries are relevant to the given passages through reasoning and are not generic, especially in the absence of ground truth for either the relevance reasoning steps or the queries.

To address the aforementioned challenges, we propose to generate training data using a retrieval-augmented Monte Carlo Tree Search (MCTS) reasoning approach (Kocsis and Szepesvári, 2006) to solve mathematical problems by LLMs. Our motivation is twofold. First, solving mathematical problems often requires applying theorems to subproblems, which enables the integration of retrievers. Theorems found to be relevant to subproblems are also relevant to the original question due to the reasoning steps that connect them. Second, verifying LLM answers against gold answers provides a proxy for evaluating the utility of retrieved theorems in solving subproblems.

4.1 Retrieval-Augmented Search-based Reasoning

We use a framework for solving mathematical problems by LLMs using a Monte Carlo Tree Search (MCTS) process, augmented with retrieval over a collection of theorems and guided by scalar feedback based on the gold answers.

MCTS overview. To solve a math problem M from a given dataset, the MCTS algorithm prompts an LLM denoted by LLM_{gen} to incrementally build a search tree that explores possible reasoning trajectories toward the final answer. The generation process in MCTS is driven by two basic compo-

nents: an *action space* A and a *reward function* R. The root node of the tree corresponds to the input question M. An edge from each node represents an action $a_i \in A$, and the resulting child node is an intermediate reasoning step s_i, which is generated by applying action a_i to the current reasoning trajectory $M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{i-1}$. Each node in the tree is assigned a value $Q(s, a)$, which represents the expected *reward* on taking action a from node s. Initially, all nodes have $Q(s, a) = 0$, resulting in a random tree exploration. As the algorithm performs rollouts, the Q values of the nodes are updated based on the rewards R and the search is guided toward better reasoning trajectories.

Action space. We extend the rStar framework (Qi et al., 2025) by adding two new actions for query generation and retrieval. As part of the *retrieval action*, RaDeR performs two additional steps: *self-reflection* which evaluates the relevance of retrieved theorems to the current problem context, and *self-summarization* which generates concise natural-language summaries of the theorems to facilitate their integration into subsequent reasoning steps. The action space A is described in detail below:

A_1: Propose One-Step Thought (OST). This action uses LLM_{gen} to generate a single CoT reasoning step s_i based on the context $M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{i-1}$. The result node can be a *terminal* node if s_i contains the answer in *boxed* notation (prompts in Appendix M).

A_2: Propose Complete Reasoning Steps (CRS). This action uses LLM_{gen} to complete the full solution by generating s_i containing multiple reasoning steps, based on the current context $M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{i-1}$. The result node is always a *terminal* node of the MCTS. Prompt details are provided in Appendix N.

A_3: Generate Query (QG). This action prompts LLM_{gen} to generate a retrieval query node s_{QG} based on the current context $M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{\text{parent(QG)}}$. A *generate query* action is always followed by a *retrieve theorem* action. For action A_3, we use few-shot prompting, details in Appendix O.

A_4: Retrieve Theorem (RT). This action uses a retriever to obtain the top-k theorems s_{RT} from the theorem corpus using the query s_{QG} of its parent node. Each of the k retrieved theorems that pass self-reflection is then added as a child node s_{RT}.

Self-Reflection and summarization. To avoid expanding the search tree with irrelevant theorem nodes, we adopt *self-reflection* (Asai et al., 2023; Xia et al., 2025) to evaluate the relevance of retrieved theorems. We use few-shot prompting to guide LLM_{gen} in generating both a relevance label ("relevant" or "non-relevant") and a supporting explanation. The input comprises the original math question M, the current intermediate solution path ($M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{\text{parent(QG)}}$), and the retrieved theorem s_{RT}. Only theorems labeled as relevant are added as nodes in the MCTS tree, while non-relevant ones are pruned early to reduce unnecessary expansion and computation. This mechanism enables RaDeR to focus exploration on more promising retrieval-augmented solution trajectories. We provide the prompts used for *self-reflection* and *self-summarization* in Appendix P.

Sampling solutions with MCTS rollouts. We sample multiple candidate solution trajectories with MCTS rollouts following rStar (Qi et al., 2025). The details are described in Appendix B.

The **reward function** $R(t)$ for a terminal node t is calculated based on whether the solution trajectory reaches the correct answer to the input question M. Specifically, if the trajectory leads to the correct answer, $R(t) = 1$; otherwise, $R(t) = 0$.

4.2 Synthesizing Training Data

To build training data for reasoning-based retrievers, we sample high-reward solution trajectories generated by the MCTS framework. For a math question M, we extract all solution trajectories that contain at least one retrieval node, denoted as $S = M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{\text{QG}} \oplus s_{\text{RT}} \oplus \cdots \oplus s_t$. From each selected solution trajectory S, we generate training samples in the form (q, p, N), where q denotes a query, p is a positive theorem relevant to q, and N is a set of hard-negative theorems for reasoning-intensive retrieval. The retrieved theorem s_{RT} in S is used as the positive document p of the generated samples. Any theorem retrieved with respect to s_{QG} that was not labeled as relevant in self-reflection is included in the set of hard negatives N.

Reasoning-intensive data. To generate reasoning samples, the set of positive and hard-negative theorems is paired with three types of queries. (1) *MCTS CoT reasoning queries:* we use the CoT reasoning steps (partial solution) up to the query node, $M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_{\text{parent(QG)}}$, as the retrieval query. Note that we do not use the query s_{QG} generated by MCTS for retrieval training, since the specific prompt format for their generation may limit the diversity of our training data.

These queries are denoted as q_{CoT}. (2) *LLM reasoning queries:* We prompt the LLM_{gen} to generate a query based on the math question M, the reasoning steps up to the query node (excluding the query), and the theorem s_{RT}. This group of queries is referred to as q_{llmq}. Using few-shot prompting, the LLM_{gen} is guided to generate reasoning-intensive queries that have low lexical and semantic term overlap with the positive theorems. Unlike the query generated during the QG action of MCTS, which directs LLM_{gen} to produce a hypothetical theorem for the current subproblem, q_{llmq} is a concise query that directly reflects the information need of the subproblem. The prompt template for this query is provided in Appendix Q. (3) *Questions as queries*: the input math question M is used as the query for retrieving theorem s_{RT}. Given the low lexical and semantic overlap between M and s_{RT}, it serves as a good reasoning sample for training. We denote this type of queries as q_{question}.

Term-matching data. Training a reasoning-based retriever should not hurt the performance of queries that can be addressed with lexical/semantic term matching. To achieve this, we follow the widely used approach of synthetically generating training samples for retrieval models. Specifically, to generate queries that have high term similarity with their respective relevant theorems, we prompt LLM_{gen} using only the theorem s_{RT}. Following Promptagator (Dai et al., 2023), we add a filtering step based on *round-trip consistency*. Only the queries for which BM25 retrieves the corresponding positive theorem s_{RT} within its top-k (k=20) results are kept. This filtering step typically results in queries with large term overlap. Hard negatives are extracted from top results of BM25 and RepLLaMA retrieval models. We denote this type of queries as q_{lexical}.

Synthesized data mix. We construct our retrieval training dataset by combining all query types, including both reasoning-based and lexical queries. Detailed statistics of the synthesized training data are provided in Appendix E.

5 Experimental Settings

Base language models. To train dense retrieval models, we primarily utilize the instruction-tuned variants of the *Qwen2.5* suite of LLMs (Yang et al., 2024). For ablation studies on model size, we train a series of Qwen-2.5-instruct models with varying parameter sizes ranging from 3B, 7B to 14B. We additionally perform ablation studies using different LLMs including gte-Qwen2-7B-Instruct (Li et al., 2023a) and Llama-3.1-Instruct (Grattafiori et al., 2024).

Math datasets for data generation. We incorporate two datasets for mathematical problem solving in RaDeR: MATH (Hendrycks et al., 2021) and a subset of examples from NuminaMath (Jia LI, 2024). Details are provided in Appendix C.

Retrieval-augmented MCTS. We apply the RaDeR framework using a fixed set of parameters, as detailed in Appendix D. For each input math question in a dataset for mathematical problem solving, we perform 16 rollouts. We use RepLLaMA (Ma et al., 2023) as the retriever in our MCTS algorithm. The retrieval corpus consists of formal mathematical theorems from ProofWiki,[2] which is also used in the BRIGHT benchmark (Su et al., 2024).

Training details. We train both the *retriever* and the *re-ranker* for reasoning-intensive relevance prediction. The retriever is trained using the standard contrastive *InfoNCE* loss using in-batch negatives in addition to the hard negatives from synthesized training samples. We use 12 hard negatives per query and treat passages from other examples in the batch as in-batch negatives. The retriever training details are presented in Appendix F. Similar to the retriever, the reranker is trained using contrastive loss. The reranker training details are presented in Appendix G.The hyperparameters used for training are presented in Table 10.

6 Experimental Results

We comprehensively evaluate our RaDeR retriever and re-ranker models on (1) the reasoning-intensive benchmark BRIGHT (Su et al., 2024) and MMTEB reasoning tasks based on RAR-b (Xiao et al., 2024), and (2) widely used benchmark MS MARCO (Bajaj et al., 2018) to measure term matching performance. We also evaluate the performance of reasoning LLMs when augmented with our RaDeR models. Following previous studies (Su et al., 2024), nDCG at top-10 (nDCG@10) is used as the evaluation metric to compare different models. We also report the recall and precision of retrieval models in Appendix I.

[2] https://proofwiki.org/ — a comprehensive collection of over 20K formal definitions and theorem proofs

	StackExchange							Coding		Theorem-based			Avg.
	Bio	Earth	Econ	Psy	Rob	Stack	Sus	Leet	Pony	AoPS	TheoQ	TheoT	
Sparse and Open-source Baselines													
BM25*	18.9	27.2	14.9	12.5	13.6	18.4	15.0	24.4	7.9	6.2	10.4	4.9	14.5
Qwen*	30.6	36.4	17.8	24.6	13.2	22.2	14.8	25.5	9.9	14.4	27.8	32.9	22.5
Qwen2	34.1	**42.6**	18.2	27.4	13.2	17.3	20.9	30.4	2.2	13.3	30.6	32.6	23.5
GritLM*	24.8	32.3	18.9	19.8	17.1	13.6	17.8	29.9	**22.0**	8.8	25.2	21.2	21.0
Inst-XL*	21.6	34.3	**22.4**	27.4	**18.2**	21.2	19.1	27.5	5.0	8.5	15.6	5.9	18.9
E5*	18.6	26.0	15.5	15.8	16.3	11.2	18.1	28.7	4.9	7.1	26.1	26.8	17.9
Proprietary Baselines													
Google*	22.7	34.8	19.6	27.8	15.7	20.1	17.1	29.6	3.6	9.3	23.8	15.9	20.0
Voyage*	23.1	25.4	19.9	24.9	10.8	16.3	15.4	30.6	1.5	7.5	27.4	11.6	17.9
RaDeR Models (MATH dataset, ($q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$))													
Qwen2.5-7B-instruct	25.4	30.0	16.7	25.3	14.0	21.3	16.3	37.0	8.2	**15.7**	**42.7**	44.4	24.6
gte-Qwen2-7B	**34.6**	38.9	22.1	**33.0**	14.8	**22.5**	**23.7**	37.3	5.0	10.2	28.4	35.1	**25.5**
Llama3.1-8B-Instruct	29.3	27.3	17.5	28.2	12.1	18.2	16.1	**38.6**	11.8	6.4	32.3	33.1	22.6
RaDeR Models (MATH+NuminaMath datasets , all query types)													
Qwen2.5-7B-instruct	25.1	28.3	18.2	25.9	15.3	22.2	16.3	35.9	6.9	10.4	40.8	**47.1**	24.4

Table 1: nDCG@10 performance of strong baselines and our models over the BRIGHT benchmark using the original question as queries for retrieval. Results of models with * are taken from (Su et al., 2024).

	StackExchange							Coding		Theorem-based			Avg.
	Bio	Earth	Econ	Psy	Rob	Stack	Sus	Leet	Pony	AoPS	TheoQ	TheoT	
Sparse and Open-source Baselines													
BM25*	53.6	54.1	24.3	38.7	18.9	27.7	26.3	19.3	17.6	3.9	19.2	20.8	27.0
Qwen*	35.5	43.1	24.3	33.4	15.4	22.9	23.9	25.4	5.2	4.6	28.7	34.6	24.8
Qwen2	38.3	47.3	24.0	35.2	15.9	23.3	27.9	29.5	8.9	2.9	30.8	35.1	26.6
GritLM*	33.3	39.1	22.4	28.9	17.4	21.3	24.1	31.9	12.0	6.7	27.3	30.1	24.5
Inst-XL*	46.7	51.2	29.9	40.5	20.8	30.1	26.9	**35.1**	2.1	8.2	24.2	17.0	26.9
E5*	29.3	43.9	19.9	26.6	11.6	19.8	15.6	29.1	0.9	5.3	27.0	36.6	22.1
Proprietary Baselines													
Google*	36.4	45.6	25.6	38.2	18.7	29.5	15.7	31.1	3.7	10.0	27.8	30.4	26.2
Voyage AI*	36.7	42.8	24.6	34.2	13.7	24.2	21.7	31.4	2.2	6.6	30.3	28.1	24.7
RaDeR Models (MATH dataset, $q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$)													
Qwen2.5-7B-instruct	32.4	38.0	21.5	33.2	14.5	25.5	18.1	30.1	14.0	11.4	**42.1**	47.2	27.3
gte-Qwen2-7B	36.1	42.9	25.2	37.9	16.6	27.4	25.0	34.8	11.9	**12.0**	37.7	43.4	29.2
Llama3.1-8B-Instruct	37.6	41.4	21.1	33.1	12.5	27.7	15.8	35.0	23.6	7.1	36.9	40.5	27.7
RaDeR Models Hybrid (MATH dataset, all query types)													
Qwen2.5-7B-instruct	37.5	40.0	19.3	31.1	14.1	25.8	17.6	27.2	18.7	9.9	40.9	46.4	27.4
+ **BM25 (Hybrid)**	56.4	56.8	28.0	43.6	25.0	34.5	27.3	31.3	25.5	9.9	40.9	46.4	35.5
+ **QwenRerank (top-100)**	**62.5**	**60.2**	31.6	**49.6**	29.6	38.9	34.8	31.0	**34.6**	10.0	32.5	50.2	38.8
RaDeR Models Hybrid (MATH dataset, $q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$)													
gte-Qwen2-7B	36.1	42.9	25.2	37.9	16.6	27.4	25.0	34.8	11.9	**12.0**	37.7	43.4	29.2
+ **BM25 (Hybrid)**	52.7	56.1	30.9	46.7	24.1	35.0	32.2	32.0	25.9	**12.0**	37.7	43.4	35.7
+ **QwenRerank (top-100)**	58.0	59.2	**33.0**	49.4	**31.8**	**39.0**	**36.4**	33.5	33.3	10.8	34.2	**51.6**	**39.2**

Table 2: nDCG@10 performance using GPT-4 CoT reasoning as queries for retrieval over BRIGHT. Results of models with * are taken from (Su et al., 2024).

6.1 BRIGHT Retrieval Performance

We first present the performance of our RaDeR retrieval models on the BRIGHT benchmark (Su et al., 2024). Following the benchmark, we compare our models against a diverse set of baselines including (1) BM25 (Robertson et al., 1995), (2) open-source dense retrieval models Instructor-XL (Su et al., 2023), E5-Mistral (Wang et al., 2024), GritLM (Muennighoff et al., 2024), gte-Qwen1.5 (Li et al., 2023a), gte-Qwen2 (Li et al., 2023a), and (3) proprietary models from Voyage (Voyage, 2024) and Google (Lee et al., 2024).

Following the benchmark, all models are evaluated in two settings: (1) using the original questions from BRIGHT as retrieval queries, and (2) using CoT reasoning steps generated by GPT-4, included in the benchmark, as retrieval queries. Performance results for the two settings are reported in Tables 1 and 2, respectively.

Overall performance. RaDeR-gte-Qwen2-7B achieves the best average performance of **25.5** on BRIGHT, outperforming strong baselines by at least **2** points in both query settings. These results demonstrate that training retrieval models for mathematical reasoning *generalizes* to other types of reasoning required for different retrieval tasks.

Theorem-based splits. Compared to baselines, RaDeR achieves the largest improvements on the TheoT and TheoQ splits in both settings, as shown in Tables 1 and 2. For example, RaDeR models improve the performance of TheoQ by **12.1** and **11.3** points when questions and CoT reasoning are used as queries, respectively. Although these splits of BRIGHT primarily require mathematical reasoning, the performance improvements on TheoQ are particularly noteworthy since the retrieval collec-

Model	Math	Coding
Open-source Baselines		
Contriever (w/ Inst.)	0.218	0.071
all-mpnet-base-v2 (w/ Inst.)	0.692	0.488
all-MiniLM-L6-v2 (w/ Inst.)	0.624	0.423
Dragon+ (w/ Inst.)	0.362	0.128
Instructor-XL (w/ Inst.)	0.580	0.495
bge-large (w/ Inst.)	0.498	0.453
E5-Mistral (w/ Inst.)	0.740	0.785
GritLM (w/ Inst.)	0.824	0.838
Proprietary Models		
Cohere-Embed-v3 (w/ Inst.)	0.721	0.566
OpenAI-ada-002 (w/ Inst.)	0.673	0.824
OpenAI-3-large (w/ Inst.)	**0.877**	**0.894**
RaDeR model (MATH, all query types)		
gte-Qwen2-7B-instruct (w/Inst.)	0.852	0.835

Table 3: nDCG@10 performance of retrievers on the Math and Coding splits of RAR-b. Performance of baselines are from (Xiao et al., 2024).

Model	DEV		DL19	DL20
	MRR@10	R@1k	nDCG@10	nDCG@10
BM25	18.4	85.3	50.6	48.0
ANCE	33.0	95.9	64.5	64.6
CoCondenser	38.2	98.4	71.7	68.4
TAS-B	34.0	97.5	71.2	69.3
GTR-base	36.6	98.3	-	-
GTR-XXL	38.8	99.0	-	-
OpenAI Ada2	34.4	98.6	70.4	67.6
bi-SimLM	39.1	98.6	69.8	69.2
RepLLaMA	**41.2**	**99.4**	**74.3**	**72.1**
SimLM	41.1	98.7	71.4	69.7
RaDeR	34.4	98.1	71.2	70.7

Table 4: Performance of baselines and RaDeR-gte-Qwen2-7B (trained with MATH) on MS MARCO. Performance of baselines are from (Ma et al., 2023; Wang et al., 2023a).

tion for this split consists of questions, a format not seen during the training of RaDeR models. The strong performance of RaDeR on this split demonstrates its potential for tasks such as demonstration selection in in-context learning, a task shown to have significant impacts on the performance of LLMs (Rubin et al., 2022).

Coding splits. Results in Table 1 show that RaDeR achieves the best performance on *LeetCode*, surpassing the strong baselines by **8** points. Results on *Pony* in the CoT reasoning setting, Table 2, also show improvements of RaDeR over the strongest baseline. These results are particularly significant, as they demonstrate that training our retrievers for mathematical reasoning yields substantial improvements in code retrieval, despite not having any code-specific training data. Our qualitative analysis reveals that in most cases, coding problems rely on solving underlying mathematical subproblems. We believe that the ability to recognize this mathematical substructure enables RaDeR models to enhance retrieval effectiveness on these coding splits. A qualitative example is provided in Appendix K.

Ablation studies. We investigated the performance of RaDeR retrievers under two ablation settings: (1) training with different subsets of query types in the synthesized data, and (2) using base LLM of different sizes as the retriever encoder. Results in Table 13 of Appendix J highlight the *complementary* role of the diverse query types in our synthesized training data. Additionally, Table 14 shows consistent performance improvements with scaling the size of the base LLM.

RaDeR Hybrid Approaches. Following ReasonIR (Shao et al., 2025), we evaluate a version of our RaDeR models, denoted by **+BM25 (Hybrid)** in Table 2, where the scores from our RaDeR retriever are interpolated with scores from BM25 using a hyperparameter α according to:

$$S_{\text{final}}(q,d) = \alpha S_{\text{RaDeR}}(q,d) + (1-\alpha) S_{\text{BM25}}(q,d)$$

where $S_{\text{BM25}}(q,d)$ and $S_{\text{RaDeR}}(q,d)$ are scores normalized using *mean* and *standard deviation* per query. In our evaluations, we chose $\alpha = 0.8$ for the non-math splits of BRIGHT and $\alpha = 1$ (no interpolation) for the math splits, since our models are trained specifically for math retrieval. As shown in Table 2, this BM25 interpolation approach yields substantial performance gains of **8.1** and **6.5** points with two RaDeR models in the GPT-4 CoT query setting.

Additionally, we follow a retrieve-then-rerank approach, using QWEN-2.5-32B-INSTRUCT to rerank the top 100 documents retrieved by the **+BM25 (Hybrid)** method. Following the setup in Shao et al. (2025), we use few-shot prompting to assign an integer relevance score (0–5) to each query-document pair. Since these scores are discrete which leads to ties, we interpolate the LLM reranker scores with the original retrieval scores using $\alpha = 0.5$, as in ReasonIR (Shao et al., 2025). We refer to this setup as **+Qwen-Rerank (top-100)** in Table 2. Using this method, RaDeR achieves state-of-the-art performance on the BRIGHT benchmark, reaching an nDCG@10 of **39.2** with the **RaDeR gte-Qwen2-7B** model trained on MATH queries ($q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$). The **RaDeR Qwen2.5-7B-Instruct** model, trained with MATH (all query types), also performs strongly with an nDCG@10 of **38.8** under the same reranking setting.

Reranker	top-k	StackExchange							Coding		Theorem-based			Avg.
		Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	
None*	-	19.2	27.1	14.9	12.5	13.5	16.5	15.2	24.4	7.9	6.2	9.8	4.8	14.3
MiniLM*	100	8.5	18.9	6.0	5.4	7.6	7.9	8.9	15.0	11.3	6.1	3.6	0.5	8.3
Gemini*	10	21.9	29.7	16.9	14.2	16.1	16.7	16.7	**24.5**	8.0	6.2	9.5	8.2	15.7
GPT-4*	100	**33.8**	**34.2**	16.7	27.0	**22.3**	**27.7**	11.1	3.4	15.6	1.2	2.0	8.6	17.0
RaDeR models (MATH, all query types)														
Qwen2.5-7B-instruct	100	26.9	30.6	17.0	24.9	18.2	17.8	20.5	23.7	14.3	4.4	**22.4**	13.6	19.5
gte-Qwen2-7B-instruct	100	26.1	30.4	16.8	26.6	18.7	18.5	16.5	18.7	**20.8**	2.9	20.4	12.4	19.1
RaDeR models (MATH+NuminaMath, all query types)														
gte-Qwen2-7B-instruct	100	25.5	31.8	**19.3**	**28.8**	22.0	19.8	20.1	17.1	11.9	1.6	18.9	**14.3**	19.3
Qwen2.5-7B-instruct	100	25.0	31.1	17.3	26.4	21.1	19.5	**21.3**	21.5	16.1	**5.1**	21.7	**14.3**	**20.0**

Table 5: nDCG@10 performance of different rerankers on BRIGHT. Reranking is performed on the top-10 or top-100 results retrieved using BM25 with the question as the query. Results with * are from (Su et al., 2024).

	StackExchange							Coding		Theorem-based			Avg.
	Bio.	Earth.	Econ.	Psy.	Rob.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	
BM25*	19.2	27.1	14.9	12.5	13.5	16.5	15.2	**24.4**	7.9	6.0	13.0	6.9	14.8
BM25 on GPT-4o CoT	**53.6**	**53.6**	24.3	38.6	18.8	22.7	25.9	19.3	17.7	3.9	18.9	20.2	26.5
Reranking on GPT-4o CoT k=100													
MonoT5-3B*	16.0	24.0	17.7	19.5	8.0	10.5	19.5	17.2	29.2	**7.1**	20.3	12.0	16.8
RankLLaMA-7B*	17.5	15.5	13.1	13.6	17.9	6.9	16.9	8.4	**46.8**	2.2	4.5	3.5	13.9
Rank1-7B (Weller et al., 2025)	48.8	36.7	20.8	35.0	22.0	18.7	**36.2**	12.7	31.2	6.3	**23.7**	37.8	**27.5**
RaDeR Models Reranking on GPT-4o CoT results k=100 (MATH, all query types)													
Qwen2.5-7B-instruct	40.8	31.8	25.0	39.7	21.6	25.7	27.2	17.3	29.9	1.6	22.4	36.9	26.7
gte-Qwen2-7B-instruct	36.0	29.3	23.2	40.0	23.2	24.1	22.2	17.8	34.9	1.5	20.4	35.3	25.7
RaDeR Models Reranking on GPT-4o CoT results k=100 (MATH+NuminaMath, all query types)													
Qwen2.5-7B-instruct	37.0	32.4	**25.5**	41.5	24.9	**26.7**	28.1	12.2	28.8	2.7	21.7	**39.1**	26.7
gte-Qwen2-7B-instruct	36.9	31.2	24.9	**43.1**	**26.4**	26.3	26.1	16.6	26.6	0.4	18.9	36.1	26.1

Table 6: nDCG@10 performance of rerankers on BRIGHT using questions as retrieval queries. The first-stage results are obtained by BM25 using GPT-4o CoT reasoning as queries. Results with * are taken from (Su et al., 2024).

6.2 Retrieval Performance on RAR-b

We evaluate the performance of RaDeR on the reasoning-retrieval tasks of MMTEB (Enevoldsen et al., 2025) which are based on the Math and Coding splits of the RAR-b (Xiao et al., 2024). As shown in Table 3, our model achieves performance comparable to the strongest open-source baseline, outperforming on the MATH split. Furthermore, it demonstrates performance that is comparable to or surpasses that of top-performing closed-source models. These results once again highlight the strong generalizability of RaDeR across diverse reasoning-based retrieval tasks.

6.3 Retrieval on Traditional IR Benchmarks

Table 4 presents the performance of RaDeR and strong baselines on the MS MARCO passage retrieval task (Bajaj et al., 2018). We evaluate on the official small subset of the MS MARCO development set, following RepLLaMA (Ma et al., 2023), as well as TREC-DL'19 (Craswell et al., 2020) and TREC-DL'20 (Craswell et al., 2021). These test sets primarily rely on term matching and do not require complex reasoning. Our model based on gte-Qwen2-7B trained with llmq, CoT, and lexical queries from MATH, demonstrates competitive performance with strong baselines. These results indicate that training our retrievers for reasoning-intensive tasks does not compromise their effectiveness on standard IR benchmarks where reasoning is not necessary.

Method	Accuracy(%)
Base model (no retrieval)	71.0
In-context RAG with RaDeR	75.0
In-context RAG with RepLLama	72.6
In-context RAG with gold theorems	77.6
MCTS with only OST action	75.0
MCTS with OST + RepLLaMA retrieval	78.9
MCTS with OST + RaDeR retrieval	80.2
MCTS with OST + gold theorems	81.5

Table 7: QA performance of Qwen-2.5-7B-Instruct on the TheoremQA theorems split of BRIGHT in different settings of retrieval augmentation.

6.4 BRIGHT Reranking Performance

We report the performance of our RaDeR rerankers on the reasoning-intensive BRIGHT benchmark. Table 5 shows the performance of reranking top-10 and top-100 of BM25 results using questions as queries. Our RaDeR rerankers based on Qwen2.5 and gte-Qwen2, both trained on all query types from MATH, outperform GPT-4 in the top-100 reranking setting by **+2.5** and **+2.1** nDCG points, respectively. The most significant improvements are observed on the LeetCode and TheoremQA

questions splits, with gains of **+20.3** and **+20.4** points, respectively.

Table 6 presents the performance of reranking BM25 results using GPT-4o CoT reasoning, where the rerankers receive only the question as the query. We compare RaDeR rerankers against strong baselines including RankLLaMA (Ma et al., 2023), Mono-T5-3B, and RANK1 (Weller et al., 2025). RANK1, the strongest baseline, is trained on 635K examples from MS MARCO, and utilizes test-time compute to perform reasoning before relevance prediction. In contrast, RaDeR models are trained with substantially fewer samples on mathematical reasoning, 43K from MATH and 78K from NuminaMATH, yet they achieve highly competitive performance. In addition, RaDeR models are significantly more computationally efficient since they only generate relevance scores.

6.5 RAG Performance using RaDeR

Table 7 provides the answer accuracy of Qwen-2.5-7B-Instruct on the TheoremQA split of BRIGHT in two settings of retrieval augmentation. First setting is in-context RAG where the input question is used as the query to retrieval models. We also perform evaluation using the retrieval-augmented MCTS framework, employing majority voting as the strategy for answer selection. In both settings, augmenting with the results of RaDeR outperforms augmentation with RepLLaMA. These results demonstrate the impact of augmenting strong LLMs with reasoning-based retrievers.

7 Conclusion

We introduce RaDeR, a suite of retrievers and rerankers designed for reasoning-intensive relevance ranking. Our approach employs a retrieval-augmented reasoning framework based on Monte Carlo Tree Search (MCTS) to generate high-quality training data. RaDeR achieves substantial improvements in both retrieval and reranking performance over state-of-the-art models on reasoning-intensive benchmarks, demonstrating strong generalizability and significantly higher data efficiency compared to existing methods.

8 Limitations

While RaDeR achieves strong performance on reasoning-intensive retrieval tasks, it has a few limitations. First, our training approach primarily focuses on examples where the retriever reasons over a single document in isolation. A promising direction for future work is to develop retrievers capable of reasoning over multiple documents jointly, where the relevance of each document is informed by the content of others, similar to the requirements in multi-hop QA tasks.

Second, RaDeR models focus on producing relevance scores without generating explicit explanations for document retrieval. Future work could explore the development of reasoning-aware retrievers that offer greater transparency and interpretability by generating explanations for their retrieval decisions, all while maintaining their efficiency for the first-stage ranking.

Lastly, we use rewards in our MCTS framework, based on the final answer of the math reasoning datasets. However, incorrect CoT reasoning path for solving a mathematical question can lead to a correct final answer, thus our training data can be noisy.

9 Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant number 2339932. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

Akiko Aizawa, Michael Kohlhase, and Iadh Ounis. 2013. Ntcir-10 math pilot task overview. In *NTCIR Conference on Evaluation of Information Access Technologies*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Akari Asai, Zexuan Zhong, Danqi Chen, Pang Wei Koh, Luke Zettlemoyer, Hannaneh Hajishirzi, and Wen tau Yih. 2024. Reliable, adaptable, and attributable language models with retrieval. *CoRR*, abs/2403.03187.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation

for information retrieval using large language models. *Preprint*, arXiv:2202.05144.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2021. Overview of the trec 2020 deep learning track. corr abs/2102.07662 (2021). *TREC*.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M Voorhees. 2020. Overview of the trec 2019 deep learning track. *TREC*.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith Hall, and Ming-Wei Chang. 2023. Promptagator: Few-shot dense retrieval from 8 examples. In *The Eleventh International Conference on Learning Representations*.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, Ömer Veysel Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Suppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal A Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Mariya Hendriksen, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri K, Maksimova Anna, Silvan Wehrli, Maria Tikhonova, Henil Shalin Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Validad Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. 2025. MMTEB: Massive multilingual text embedding benchmark. In *The Thirteenth International Conference on Learning Representations*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Xinyu Guan, Li Lyna Zhang, Yifei Liu, Ning Shang, Youran Sun, Yi Zhu, Fan Yang, and Mao Yang. 2025. rstar-math: Small llms can master math reasoning with self-evolved deep thinking. *Preprint*, arXiv:2501.04519.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *Preprint*, arXiv:2103.03874.

Minda Hu, Licheng Zong, Hongru Wang, Jingyan Zhou, Jingjing Li, Yichen Gao, Kam-Fai Wong, Yu Li, and Irwin King. 2024. Serts: Self-rewarding tree search for biomedical retrieval-augmented generation. *Preprint*, arXiv:2406.11258.

Yunhai Hu, Yilun Zhao, Chen Zhao, and Arman Cohan. 2025. Mcts-rag: Enhancing retrieval-augmented generation with monte carlo tree search. *Preprint*, arXiv:2503.20757.

Edward Beeching Jia LI. 2024. Numinamath. [https://github.com/project-numina/aimo-progress-prize](https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf).

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *Preprint*, arXiv:2503.09516.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R. Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, Yi Luan, Sai Meher Karthik Duddu, Gustavo Hernandez Abrego, Weiqiang Shi, Nithi Gupta, Aditya Kusupati, Prateek Jain, Siddhartha Reddy Jonnalagadda, Ming-Wei Chang, and Iftekhar Naim. 2024. Gecko: Versatile text embeddings distilled from large language models. *Preprint*, arXiv:2403.20327.

Yibin Lei, Liang Ding, Yu Cao, Changtong Zan, Andrew Yates, and Dacheng Tao. 2023. Unsupervised dense retrieval with relevance-aware contrastive pre-training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 10932–10940, Toronto, Canada. Association for Computational Linguistics.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023a. Towards general text embeddings with multi-stage contrastive learning. *Preprint*, arXiv:2308.03281.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023b. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

Xueguang Ma, Luyu Gao, Shengyao Zhuang, Jiaqi Samantha Zhan, Jamie Callan, and Jimmy Lin. 2025. Tevatron 2.0: Unified document retrieval toolkit across scale, language, and modality. *arXiv preprint arXiv:2505.02466*.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2023. Fine-tuning llama for multi-stage text retrieval. *Preprint*, arXiv:2310.08319.

Behrooz Mansouri, Vít Novotný, Anurag Agarwal, Douglas W. Oard, and Richard Zanibbi. 2022. Overview of arqmath-3 (2022): Third clef lab on answer retrieval for questions on math (working notes version). In *Conference and Labs of the Evaluation Forum*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *Preprint*, arXiv:2402.09906.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback. *Preprint*, arXiv:2112.09332.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2017. MS MARCO: A human-generated MAchine reading COmprehension dataset.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, Online. Association for Computational Linguistics.

Zhenting Qi, Mingyuan MA, Jiahang Xu, Li Lyna Zhang, Fan Yang, and Mao Yang. 2025. Mutual reasoning makes smaller LLMs stronger problem-solver. In *The Thirteenth International Conference on Learning Representations*.

Stephen Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. 1995. Okapi at trec-3. In *Overview of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. Gaithersburg, MD: NIST.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Chris Samarinas and Hamed Zamani. 2025. Distillation and refinement of reasoning in small language models for document re-ranking. *Preprint*, arXiv:2504.03947.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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, et al. 2025. Reasonir: Training retrievers for reasoning tasks. *arXiv preprint arXiv:2504.20595*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *Preprint*, arXiv:2503.05592.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada. Association for Computational Linguistics.

Hongjin Su, Howard Yen, Mengzhou Xia, Weijia Shi, Niklas Muennighoff, Han yu Wang, Haisu Liu, Quan Shi, Zachary S. Siegel, Michael Tang, Ruoxi Sun, Jinsung Yoon, Sercan O. Arik, Danqi Chen, and Tao Yu. 2024. Bright: A realistic and challenging benchmark for reasoning-intensive retrieval. *Preprint*, arXiv:2407.12883.

Hieu Tran, Zonghai Yao, Junda Wang, Yifan Zhang, Zhichao Yang, and Hong Yu. 2024. Rare: Retrieval-augmented reasoning enhancement for large language models. *Preprint*, arXiv:2412.02830.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *Preprint*, arXiv:2212.10509.

Voyage. 2024. Voyage embedding models. *https://docs.voyageai.com/docs/embeddings*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2023a. Simlm: Pre-training with representation bottleneck for dense passage retrieval. *Preprint*, arXiv:2207.02578.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916, Bangkok, Thailand. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023b. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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

Yuan Xia, Jingbo Zhou, Zhenhui Shi, Jun Chen, and Haifeng Huang. 2025. Improving retrieval augmented language model with self-reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25534–25542.

Chenghao Xiao, G Thomas Hudson, and Noura Al Moubayed. 2024. Rar-b: Reasoning as retrieval benchmark. *Preprint*, arXiv:2404.06347.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023a. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023b. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

Wei Zhong, Jheng-Hong Yang, Yuqing Xie, and Jimmy Lin. 2022. Evaluating token-level and passage-level dense retrieval models for math information retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1092–1102, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Wei Zhong, Xinyu Zhang, Ji Xin, Richard Zanibbi, and Jimmy Lin. 2021. Approach zero and anserini at the clef-2021 arqmath track: Applying substructure search and bm25 on operator tree path tokens. *Proc. CLEF 2021 (CEUR Working Notes)*.

A Detailed Discussion of Related Works

A.1 Mathematical Information Retrieval

Math Information Retrieval (Math IR) has been extensively studied within the IR community, focusing on the task of retrieving relevant mathematical documents such as theorems, formulas, similar questions, or textbooks to solve a given math problem. Early models, such as Approach0 (Zhong et al., 2021), perform retrieval by using structural similarities between the formulas in queries and documents.

MathBERT (Zhong et al., 2022) pre-trained a cross-encoder BERT-base model on a corpus of 1.69 million math documents containing both text and formulas. Several approaches (Zhong et al., 2022) explored hybrid methods combining dense neural retrievers with structural and lexical retrievers to improve retrieval performance.

The NTCIR-10 Math Pilot Task (Aizawa et al., 2013) marked one of the first collaborative efforts to establish evaluation frameworks for *mathematical formula search*, while the ARQMATH Lab tasks (Mansouri et al., 2022) extended MATH IR to a Community Question Answering (CQA) setting, utilizing user-generated data from *Math StackExchange*.

More recently, datasets such as **BRIGHT** (Su et al., 2024) and RAR-B (Xiao et al., 2024) have been introduced as evaluation benchmarks for Math IR, focusing on tasks including *Theorem Retrieval*, *Similar Questions Retrieval*, and *Answer Retrieval*. In this work, we adopt both **BRIGHT** and RAR-B as evaluation datasets to assess the effectiveness of our trained retriever.

A.2 CoT Reasoning

IRCoT (Trivedi et al., 2023) uses the last reasoning step as a retrieval query, conditioning future steps on retrieved documents. ITER-RETGEN (Shao et al., 2023) alternates between retrieval and generation, showing improvements in tasks such as multi-hop QA and fact verification. REACT (Yao et al., 2023b) iteratively generates (thought, action, observation) sequences, using intermediate reasoning to drive retrieval. Toolformer (Schick et al., 2023) employs self-supervised training to help models autonomously determine when to invoke external retrieval tools, like the *Wikipedia Search API*.

A.3 Reasoning-Based Re-Ranking Models

RANK-1(Weller et al., 2025) employs knowledge distillation from DeepSeek-R1, extracting over 600K reasoning examples from the MS-MARCO dataset(Nguyen et al., 2017) to train a re-ranking model. Similarly, InteRank (Samarinas and Zamani, 2025) uses reinforcement learning to train a 3B-parameter re-ranking model, generating reasoning explanations alongside relevance scores for (query, document) pairs. Despite their improved retrieval quality, these re-ranking models remain inherently dependent on the first-stage retriever's candidate set, typically derived from lexical or semantic matching, thus limiting their effectiveness on reasoning-intensive retrieval tasks. In contrast, RaDeR develops a dedicated first-stage retriever from pretrained language models, including *Qwen2.5* (Yang et al., 2024), *Llama 3.1* (Grattafiori et al., 2024), and *gte-Qwen2-7B-instruct* (Li et al., 2023b), tailored explicitly to reasoning-intensive search scenarios.

B Sampling MCTS Rollout Solutions

The MCTS proceeds with multiple iterations of four main processes: *selection*, *expansion*, *simulation* and *backpropagation*. To balance exploration and the exploitation, the *selection* step starts from the root node and uses Upper Confidence Bounds applied to trees (UCT) to traverse through child nodes, continuing until a leaf node is reached. Formally, we select the node with maximum UCT value at each branch of the traversal:

$$\mathrm{UCT}(s,a) = \frac{Q(s,a)}{N(s,a)} + c\sqrt{\frac{ln(N_{\mathrm{parent}}(s))}{N(s,a)}}$$

where $N(s,a)$ is the number of times node s has been visited till now and $Q(s,a)$ is the expected reward of node s under action a and c is a hyperparameter. If the leaf node is not a *terminal* node, the *expansion* step adds child nodes to the leaf node to represent potential future actions. The *simulation* step selects one of the newly added child nodes at random and performs rollouts/simulations by selecting actions randomly until we reach a terminal node t. Based on whether the terminal node t reaches the correct gold answer G, we calculate a reward value $R(t)$ and update $Q(s,a)$ values for all the nodes s_i in the collected solution trajectory $M \oplus s_1 \oplus s_2 \oplus \cdots \oplus s_t$ as:

$$Q(s_i,a) = Q(s_i,a) + R(t)$$

The $N(s, a)$ values are also incremented as

$$N(s_i, a) = N(s_i, a) + 1$$

.

C Math Datasets for Data Generation

To construct our training dataset, we leverage mathematical reasoning benchmarks consisting of natural language math questions paired with a correct answer, typically represented as a boxed numerical value in LaTeX. Our data generation pipeline does not require access to gold step-by-step solutions.

MATH The MATH dataset comprises mathematical problems across 8 different subject types (Prealgebra, Precalculus, Algebra, Geometry, Intermediate Algebra, Counting and Probability, and Number Theory) and five difficulty levels (from 1 to 5, where 1 denotes the easiest).

NuminaMath. NuminaMath is a large-scale collection comprising 860K competition-level math problems paired with solutions. Our MCTS leverages the OrcaMATH, AMC, AIME, Chinese K-12 Exam, Olympiad, and AOP forum splits from NuminaMath.

Parameter	Value
No of RT nodes per action (Top-k)	5
No of OST nodes added (per action)	2
No of QG nodes added (per action)	1
No of rollouts	16
Max depth	6
MCTS Exploration weight C	2
MCTS Weight Scheduler	const
LLM_{gen} temperature	0.8
LLM_{gen} top-k	40
LLM_{gen} top-p	0.95
BF16	Enabled
GPUs	2 A100s

Table 8: MCTS Parameters

D MCTS Parameters

Table 8 shows the values of hyperparameters for the MCTS algorithm.

E Statistics of Synthesized Data

Table 9 shows the number of synthesized samples from each dataset for mathematical problem solving.

Dataset (query type)	# of Samples
MATH (q_{llmq})	18,586
MATH (q_{CoT})	7,312
MATH (q_{question})	7,312
MATH (q_{lexical})	9,910
MATH (all queries)	43,120
NuminaMATH (q_{llmq})	39,639
NuminaMATH (q_{CoT})	24,280
NuminaMATH (q_{question})	10,241
NuminaMATH (q_{lexical})	4,158
NuminaMATH (all queries)	78,318
MATH+NuminaMATH (all queries)	121,438

Table 9: Statistics of synthesized data for retrieval training.

F Retriever Training Details

We append an end-of-sequence token (`EOS` token) to the input query or document to form the input sequence to our base LLM. Thus, the vector embedding of a query or a document (denoted as t) is computed as:

$$E_t = \text{Decoder}(t_1 t_2 \ \cdots \ t_k\texttt{<eos>})[-1],$$

where $\text{Decoder}(\cdot)$ represents the LLM model (such as Qwen or Llama), which returns the last layer token representations for each input token. We take the representation of the end-of-sequence token as the representation of the input sequence $t_1, \ldots, t_k$, which can be either a query q or a document d. Relevance of d to q is computed using the cosine similarity of their corresponding dense representations E_q and E_d as:

$$s(q, d) = \cos(E_q, E_d).$$

The model is then optimized end-to-end using the InfoNCE loss:

$$\mathcal{L}(q, p, D^-) = -\log \frac{\exp(s(q,p))}{\exp(s(q,p)) + \sum_{d^- \in D^-} \exp(s(q,d^-))}, \tag{1}$$

where p denotes a document relevant to the query q (based on human annotations), while D^- is the set of negative (non-relevant) documents.

G Reranker Training Details

Our reranker model is trained as pointwise reranker. The input to the model is a concatenation of the query and a candidate document, with the model

generating a score that indicates the relevance of the document to the query (Nogueira et al., 2020).

In more detail, our model reranks a query-document pair as shown below:

$$\text{input} = \text{query: } \{q\} \text{ document: } \{d\} \texttt{<eos>}$$
$$s(q, d) = \text{Linear}(\text{Decoder}(\text{input})[-1])$$

Here, $\text{Decoder}(\cdot)$ represents the LLM model (such as Qwen or Llama), which returns the last layer token representations for each input token , and $\text{Linear}(\cdot)$ is a linear projection layer that maps the final hidden state corresponding to the end-of-sequence token to a scalar relevance score. The training uses same loss, used for retriever training, with no use of in-batch negatives.

Parameter	Value
Train Group Size	12
Warmup Steps	28
Per Device Train Batch Size	2
Gradient Accumulation Steps	16
DDP Timeout	1800
Temperature	0.01
Learning Rate	1e-4
LR Scheduler Type	Linear
Number of Train Epochs	1
BF16	Enabled
GPUs	2 A100s

Table 10: Retriever Training Hyperparameters.

H Retriever/Reranker Training Hyperparameters

We used *Tevatron* (Ma et al., 2025) package for training. The hyperparameters used for finetuning retrieval and reranking models are presented in Table 10.

I Retrieval Performance on BRIGHT

In addition to nDCG performance of retrieval models, we compare our RaDeR models with baselines in terms of precision at top 10 documents in Table 11 and recall at top 10 documents in Table 12 when questions are used as retrieval queries.

J Ablation Studies

Effect of query types in synthesized samples. We evaluate the impact of different query types in the synthesized retrieval data by comparing variants of our RaDeR models trained on subsets including specific query types. Table 13 summarizes the results of this ablation on BRIGHT.

We observe a consistent improvement in retrieval performance as additional query types are included during training. These results highlight the *complementary* nature of the diverse query types in our synthesized training data.

Impact of base-LLM size. Table 14 presents the performance of our RaDeR retrievers using Qwen-2.5-instruct models of varying sizes (3B, 7B, 14B). We observe consistent gains with model scaling, with the 14B model outperforming the 7B variant by 2 points on average.

K Analysis

We present examples from our synthesized training data, along with qualitative examples of RaDeR from the BRIGHT evaluation. We highlight representative cases where RaDeR successfully retrieves the correct theorem, as well as illustrative failure cases to analyze its limitations.

Examples of queries from Retrieval Training Dataset. We present a qualitative example of different query types (q_{llmq}, q_{CoT} and q_{lexical}) for a question M and positive theorem p from our dataset in Figure 3. As illustrated in the figure, q_{CoT} captures the broader reasoning context of the mathematical question, thereby encouraging the model to learn a more challenging retrieval task. In contrast, q_{lexical} exhibits high lexical overlap with the associated positive document, which helps for queries which do not require reasoning. The query q_{llmq} strikes a balance between these two extremes, as it is generated using both M and P, effectively combining elements of both contextual reasoning and lexical similarity.

Examples from BRIGHT evaluation. We present examples from TheoremQA theorems split of BRIGHT, to qualitatively compare RaDeR with other baseline retrievers. For the example of Figure 4, RaDeR successfully performs nuanced reasoning to successfully retrieve the gold theorem, whereas a strong baseline, Qwen2, fails to do so. However, challenges still remain. Figure 5 presents an example, where the question is about directionality in a family tree graph. Both retrievers Qwen2 and RaDeR, fail to capture the query's focus on *acyclicity* and instead match on

	StackExchange							Coding		Theorem-based			Avg.
	Bio	Earth	Econ	Psy	Rob	Stack	Sus	Leet	Pony	AoPS	TheoQ	TheoT	
Sparse and Open-source Baselines													
BM25*	7.6	12.4	7.1	6.0	5.6	8.0	6.1	6.0	7.9	3.1	2.2	1.3	6.1
Qwen*	13.5	14.1	8.2	11.2	5.8	10.1	6.1	6.3		9.7	7.1	6.2	7.3
GritLM*	11.1	12.7	9.2	10.8	6.8	6.0	6.8	7.5	17.9	4.6	5.7	5.3	8.7
Inst-XL*	10.0	13.8	10.6	11.0	6.7	8.8	9.2	6.3	4.6	4.7	3.3	1.9	7.6
E5*	8.9	10.2	8.1	8.6	7.2	4.6	6.9	6.9	4.9	4.2	5.7	6.5	6.9
Proprietary Baselines													
Google*	10.3	12.2	8.9	11.4	5.6	8.3	7.9	6.9	3.6	5.0	4.8	4.2	7.4
Voyage AI*	11.0	9.9	9.6	11.0	5.6	7.3	7.5	7.5	1.1	5.0	5.6	3.2	7.0
RaDeR Models (MATH dataset, $q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$)													
Qwen2.5-7B-instruct	11.6	11.7	6.8	9.9	5.3	7.9	6.7	8.5	8.0	8.2	10.2	9.9	**8.7**
RaDeR Models (MATH+NuminaMath datasets, all query types)													
Qwen2.5-7B-instruct	12.2	11.5	7.8	10.6	5.7	7.4	6.8	8.1	6.2	6.0	9.2	11.3	8.6

Table 11: Precision@10 performance using question as queries for retrieval over BRIGHT.

	StackExchange							Coding		Theorem-based			Avg.
	Bio	Earth	Econ	Psy	Rob	Stack	Sus	Leet	Pony	AoPS	TheoQ	TheoT	
Sparse and Open-source Baselines													
BM25*	21.8	31.4	16.8	15.5	19.4	16.8	21.1	29.5	3.6	6.0	11.4	9.0	16.9
Qwen*	38.2	40.6	18.5	29.5	14.5	22.4	17.4	32.1	4.6	14.8	30.0	39.4	25.2
GritLM*	30.3	38.8	18.3	26.9	21.3	15.1	23.4	36.3	8.2	9.4	26.2	26.6	23.4
Inst-XL*	27.3	38	25.4	35.6	22	21.1	23.9	31.8	2.5	8.9	16.6	9.8	21.9
E5*	22.0	29.4	18.4	18.3	18.7	11.9	23.0	34.6	2.4	8.2	27.2	34.8	20.7
Proprietary Baselines													
Google*	26.1	36.9	20.6	31.4	17.7	21.6	23.7	33.5	1.9	10.4	24.0	22.1	22.5
Voyage AI*	29.3	31.2	21.0	31.0	15.0	17.5	20.5	41.5	0.6	8.7	28.5	15.4	21.7
RaDeR Models (MATH dataset, $q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$)													
Qwen2.5-7B-instruct	30.6	36.9	19.5	30.1	18.5	28.6	21.1	46.5	6.4	18.1	48.3	56.0	30.1
gte-Qwen2-7B	41.6	42.5	26.1	41.2	18.6	32.5	31.7	44.7	3.4	13.4	32.5	47.9	**31.3**
Llama3.1-8B-Instruct	35.7	33.5	18.4	36.4	14.8	28.5	20.3	45.2	6.4	7.7	37.0	43.4	27.3
RaDeR Models (MATH+NuminaMath datasets, all query types)													
Qwen2.5-7B-instruct	31.3	34.2	21.5	31.9	18.8	30.5	21.5	45.6	5.1	11.7	44.7	63.8	30.1

Table 12: Recall@10 performance using question as queries for retrieval over BRIGHT.

the topic of *structural hierarchy* in the graph, thus leading to an incorrect retrieval. This highlights the need for further advancements in reasoning-aware retrieval methods.

Analysis of Coding example from Leetcode In Figure 6, we present a qualitative example from Leetcode, which shows how recognizing mathematical subtructure in a coding problem can help RaDeR retrievers. Retrieving relevant documents for the coding task, requires application of mathematical reasoning which RaDeR models excel in, whereas BM25 only performs lexical matching based on the word *rectangle*, which results in an incorrect retrieval.

Data of RaDeR-	StackExchange							Coding		Theorem-based			Avg.
	Bio	Earth	Econ	Psy	Rob	Stack	Sus	Leet	Pony	AoPS	TheoQ	TheoT	
q_{llmq}	**25.8**	**32.0**	15.8	23.3	14.8	18.7	15.7	29.9	**16.4**	13.0	38.8	35.4	23.3
$q_{\text{llmq}} + q_{\text{CoT}}$	24.5	26.9	**17.0**	**25.6**	**14.8**	21.1	**17.1**	30.6	9.9	12.8	42.5	38.5	23.4
$q_{\text{llmq}} + q_{\text{CoT}} + q_{\text{lexical}}$	25.4	30.0	16.7	25.3	14.0	**21.3**	16.3	**37.0**	8.2	**15.7**	**42.7**	**44.4**	**24.6**

Table 13: nDCG@10 performance of RaDeR on BRIGHT when retrievers are trained using different types of samples generated from MATH. The original math question is used as the query. RaDeR Qwen2.5-7B-instruct models are used for the ablations.

RaDeR Models (all query types)	StackExchange							Coding		Theorem-based			Avg.
	Bio	Earth	Econ	Psy	Rob	Stack	Sus	Leet	Pony	AoPS	TheoQ	TheoT	
Qwen2.5-3B-instruct	26.9	29.7	16.5	24.8	13.3	19.3	15.2	37.5	**10.6**	9.4	34.9	36.9	22.9
Qwen2.5-7B-instruct	25.1	28.3	**18.2**	25.9	15.3	22.2	16.3	35.9	6.9	10.4	<u>40.8</u>	**47.1**	<u>24.4</u>
Qwen2.5-14B-instruct	**30.9**	**31.5**	17.3	**27.9**	**16.0**	**24.2**	**16.4**	**40.9**	10.2	**12.7**	**42.6**	45.8	**26.4**

Table 14: nDCG@10 performance of RaDeR on BRIGHT when retrievers are trained using different sizes of Qwen-2.5-instruct models (3B, 7B, and 14B). The original math question is used as the query.

Example

Question (M): Let $f(x)$, $g(x)$, and $h(x)$ be linear functions. Define

$$j(x) = \max\{f(x), g(x), h(x)\}, \qquad k(x) = \min\{f(x), g(x), h(x)\}.$$

Suppose the graph of $y = j(x)$ over $-3.5 \le x \le 3.5$ consists of three connected line segments: it increases linearly to $(-2, 2)$, remains constant at $y = 2$ from $x = -2$ to $x = 2$, and then increases linearly from $(2, 2)$ to $x = 3.5$. Let ℓ denote the total length of the graph of $y = k(x)$ over the same interval. Find ℓ^2.

Positive document (p): Linear Real Function - A real function f is called linear if it has the form $f(x) = ax + b$, where a and b are real numbers.

Different Query Types

LLM generated query (q_{llmq}): Calculate the length of the graph of a piecewise linear function defined by the minimum of three linear functions over a given interval.

CoT reasoning query (q_{CoT}): Question + To determine the length of the graph of $y = k(x)$ for $-3.5 \le x \le 3.5$, we first need to understand the behavior of $j(x)$, the maximum of the three linear functions $f(x)$, $g(x)$, and $h(x)$. From the graph, we can see that $j(x)$ transitions between the three linear functions at points where two of the functions are equal. The graph of $j(x)$ has horizontal segments and linear segments connecting the points where the maximum changes.

Lexical query (q_{lexical}): What is the definition of a linear real function and what form must it take for all real numbers x?

Figure 3: Examples of different query types from our retrieval training dataset built for the given math question.

Example

Query: Everyone that you invite to a party will be either a fan of *football* or a fan of *basketball*, but never both. What is the smallest number of guests you need to invite to ensure that there are either **3 people** who are all fans of football or **3 people** who are all fans of basketball?

Gold Theorem Id: 7627
Gold Theorem: *Ramsey Theorem: Ramsey's Theorem guarantees that for any edge-coloring of a sufficiently large complete graph, there exists a monochromatic complete subgraph. More formally, for integers $n_1, n_2, \ldots, n_c$, there exists a Ramsey number $R(n_1, \ldots, n_c)$ such that any c-coloring of a complete graph on $R(n_1, \ldots, n_c)$ vertices contains a monochromatic K_{n_i} in some color i. (Note that the gold theorem has no lexical overlap with the original question.)*

Qwen2 retriever

Top Retrieved Theorem: *Pigeonhole Principle: Let S be a finite set with n elements, partitioned into k subsets $S_1, S_2, \ldots, S_k$. Then at least one subset S_i satisfies:*

$$|S_i| \geq \left\lceil \frac{n}{k} \right\rceil$$

(Incorrect retrieved theorem)

RaDeR GTE-Qwen2-7B-instruct (all query types) retriever

Top Retrieved Theorem: *Ramsey's Theorem guarantees that for any edge-coloring of a sufficiently large complete graph, there exists a monochromatic complete subgraph. More formally, for integers $n_1, n_2, \ldots, n_c$, there exists a Ramsey number $R(n_1, \ldots, n_c)$ such that any c-coloring of a complete graph on $R(n_1, \ldots, n_c)$ vertices contains a monochromatic K_{n_i} in some color i.* **(Correct theorem retrieved)**

Explanation

Explanation: While the Pigeonhole Principle might appear relevant at first glance due to the presence of element selection in the query, the problem is more appropriately modeled as a graph coloring task. Specifically, it reduces to a two-coloring of the edges of a complete graph, where the objective is to guarantee the existence of a monochromatic triangle. Our RaDeR retriever demonstrates the ability to perform such nuanced reasoning, correctly interpreting the structural semantics of the query and retrieving the correct gold document: the Ramsey Theorem.

Figure 4: Example of RaDeR success case compared to Qwen2, from TheoremQA theorems of BRIGHT.

Example

Query: Imagine you have a family tree that shows the lineage from ancestors to their descendants, with arrows pointing from parents to children. Can this family tree, with its directed lineage paths, be accurately represented without the arrows while still maintaining the correct relationships? True or false?

Gold Theorem Id: 3778
Gold Theorem: *The "'girth"' of G is the smallest length of any cycle in G. An acyclic graph is defined as having a girth of infinity. (Note that the gold theorem has no lexical overlap with the original question.)*

Qwen2 retriever

Top Retrieved Theorem: *Rooted Tree Corresponds to Arborescence: Let $T = (V, E)$ be a rooted tree with root r. Then there exists a unique orientation of T that forms an r-arborescence.*

(Incorrect theorem retrieved)

RaDeR GTE-Qwen2-7B-instruct (all query types) retriever

Top Retrieved Theorem: *Rooted Tree Corresponds to Arborescence: Let $T = (V, E)$ be a rooted tree with root r. Then there exists a unique orientation of T that forms an r-arborescence.*

(Incorrect theorem retrieved)

Explanation

Explanation: This case demonstrates a semantic mismatch: while the retrieved theorem on arborescences aligns superficially with the query's mention of family trees, the actual focus is on acyclicity, better captured by the gold theorem on girth. The error highlights how surface-level similarity can mislead retrieval.

Figure 5: Example of RaDeR Failure cases from TheoremQA theorems BRIGHT

Example

Query:

Problem: Given an array of integers `heights` representing the histogram's bar heights (each bar has width 1), return the area of the largest rectangle in the histogram.
Example 1. Input: `heights = [2,1,5,6,2,3]`. Output: `10`. The largest rectangle has area 10.
Example 2. Input: `heights = [2,4]`. Output: `4`.
Constraints. $1 \leq$ `heights.length` $\leq 10^5$, $0 \leq$ `heights[i]` $\leq 10^4$.

BM25 retriever

Top Retrieved Document:
Problem: Given an array `rectangles`, where `rectangles[i] = [width`$_i$`, height`$_i$`]`, return the number of pairs (i, j) with $i < j$ such that the rectangles have the same width-to-height ratio, i.e., $\frac{width_i}{height_i} = \frac{width_j}{height_j}$ (using decimal division).
(Incorrect retrieved theorem)

RaDeR Llama3.1-8B-instruct retriever

Top Retrieved Document:
Problem: Given a binary matrix `matrix` of size `rows` $\times$ `cols`, return the area of the largest rectangle containing only `'1'`s.
Approach: For each row, treat it as the base of a histogram where the height at each column counts consecutive `'1'`s up to that row. For each histogram, compute the largest rectangle using a stack-based approach (*similar to the Largest Rectangle in Histogram problem*).
Constraints. $1 \leq$ `rows, cols` ≤ 200; `matrix[i][j]` $\in \{'0', '1'\}$. **(Correct theorem retrieved)**

Explanation

The coding problem has a mathematical substructure because it requires computing the maximum rectangular area under a histogram, which involves geometric reasoning and optimization over intervals of the graph. Such a structure helps RaDeR to do better at code retrieval tasks.

Figure 6: Analysis of RaDeR Llama3.1-8B-instruct success case for Coding Example from *Leetcode* split of BRIGHT

L Example Prompts

This section provides the prompts used in our work.

M Prompt for OST thought action

We provide the prompts used for our **One Step Thought Action** (OST) A_1 used in the MCTS, in Figure 7.

N Prompt for CRS action

We provide the prompts for the Complete Reasoning Steps action A_2 of MCTS, in Figure 8.

O Prompt for MCTS Query Generation

In this section, we provide the prompts for the query generation action A_3 used in our MCTS framework. These prompts guide LLM_{gen} to generate a plausible hypothetical theorem that could help solve the current subproblem. This format of the prompt is chosen to increase the likelihood of term matching retrievers finding the relevant theorems. The instruction is provided in Figure 9 and the few shot examples used are presented in Figure 10.

P Prompt for Self Reflection MCTS

In this section, we provide the prompts for the *self reflection* and *self-summarization* mechanisms used in our MCTS framework. The self-reflection prompt is shown in Figure 11 and the summatization prompt used is presented in Figure 12.

Q Prompt for LLM Reasoning Query generation

In this section, we provide the prompts used for generating $\text{LLM}_{\text{query}}$ based on the input math question, the reasoning CoT context and the theorem retrieved in the MCTS. The instruction with the few-shot examples are provided in Figure 13.

Instruction

A chat between a curious user and an AI assistant. The assistant gives step-by-step solutions to the user's questions. At the final step, a conclusive answer is given in the format of "The answer is: boxed{ANSWER}.", where ANSWER should be a numeric answer.

{*few shot examples*}
Question: {question}
Response: Let's think step by step.

Few Shot Examples

Question: Gracie and Joe are choosing numbers on the complex plane. Joe chooses the point $1 + 2i$. Gracie chooses $-1 + i$. How far apart are Gracie and Joe's points?
Response: Let's think step by step.
Step 1: Distance formula is $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.
Step 2: Joe's point $(1, 2)$, Gracie's $(-1, 1)$.
Step 3: Distance $\sqrt{((-1) - (1))^2 + ((1) - (2))^2} = \sqrt{4 + 1} = \sqrt{5}$.
Step 4: Points are $\boxed{\sqrt{5}}$ units apart.
Step 5: The answer is: $\sqrt{5}$.

Question: What is the sum of all positive integer values of n for which $\frac{n+6}{n}$ is an integer?
Response: Let's think step by step.
Step 1: $n + 6$ divisible by n.
Step 2: n must be factor of 6.
Step 3: Factors of 6 are 1, 2, 3, 6. Sum is $1 + 2 + 3 + 6 = \boxed{12}$.
Step 4: The answer is: 12.

Question: Abigail, Beatrice, and Carson sell eggs in cartons of 12. Abigail has 37, Beatrice 49, Carson 14 eggs. How many eggs remain after selling all cartons?
Response: Let's think step by step.
Step 1: Total eggs $37 + 49 + 14 = 100$.
Step 2: Divide by 12: $100 \div 12 = 8$ cartons, remainder 4.
Step 3: Remaining eggs $\boxed{4}$.
Step 4: The answer is: 4.

Question: Circle T has center $T(-2, 6)$, reflected across y-axis, translated 8 units down. Find coordinates of image center.
Response: Let's think step by step.
Step 1: Reflect across y-axis: $(-(-2), 6) = (2, 6)$.
Step 2: Translate down 8 units: $(2, 6 - 8) = (2, -2)$.
Step 3: Image coordinates $\boxed{(2, -2)}$.
Step 4: The answer is: $(2, -2)$.

Figure 7: Prompt for MCTS One Step Thought Action

Instruction

You are given context about a mathematical question. Your job is to generate the next steps of the solution and complete the solution. In the end of your response, a final answer is given in the format of "$\boxed<ANSWER>$" , where <ANSWER> should be a numeric result or a math expression.

Few Shot Examples

Context: Gracie and Joe are choosing numbers on the complex plane. Joe chooses the point $1 + 2i$. Gracie chooses $-1 + i$. How far apart are Gracie and Joe's points? The distance between two points (x_1, y_1) and (x_2, y_2) in the complex plane is given by $\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$. Joe's point is $(1, 2)$ and Gracie's point is $(-1, 1)$.
Next steps: The distance is $\sqrt{((-1) - (1))^2 + ((1) - (2))^2} = \sqrt{4 + 1} = \sqrt{5}$. Therefore, Gracie and Joe's points are $\boxed{\sqrt{5}}$ units apart.

Context: What is the sum of all positive integer values of n for which $\frac{n+6}{n}$ is an integer?
Next steps: We want $\frac{n+6}{n}$ to be integer, thus $n + 6$ divisible by n. Since n positive, n must factor 6. Factors: 1, 2, 3, 6. Sum is $1 + 2 + 3 + 6 = \boxed{12}$.

Context: Abigail, Beatrice, and Carson sell eggs in cartons of 12. Abigail has 37 eggs, Beatrice has 49, Carson has 14. First, total eggs: $37 + 49 + 14 = 100$. Divide by 12: $100 \div 12 = 8$ remainder 4.
Next steps: Eggs remaining: $\boxed{4}$. The answer is: 4.

Context: Circle T has center $T(-2, 6)$, reflected across y-axis, translated 8 units down. Reflecting across y-axis negates x-coordinate.
Next steps: Reflection gives $(2, 6)$. Translating down 8 units: $(2, 6 - 8) = (2, -2)$. Therefore, coordinates are $\boxed{(2, -2)}$.

Figure 8: Prompt for Complete Remaining Steps (CRS) Action

Instruction

You are given a mathematical question and an intermediate solution. What are the mathematical concepts, theorems, formulas that would be useful for solving this question. Please provide the theorem name, followed by the theorem statement, followed by the preconditions in the theorem, and why the preconditions are satisfied in the question we have. Also mention which specific subjects in math this theorem corresponds to. List out as many number of theorems that are highly relevant to this question. Do not output the final solution. Do not generate theorems which are already present in the intermediate solution.

Figure 9: Instruction for MCTS Query Generation Action

Few Shot Examples

Theorem: Polynomial Division Algorithm
Theorem Statement: For any two polynomials P(z)(dividend) and D(z) (divisor), with $deg(P(z)) \geq deg(D(z))$, there exist unique polynomials Q(z) (quotient) and R(z) (remainder) such that: $P(z) = D(z)Q(z) + R(z)$.
Question: Find quotient of $\frac{3z^4-4z^3+5z^2-11z+2}{2+3z}$.
Intermediate Solution: Apply polynomial long division.
Query: (Polynomial Division) $P(z) = D(z)Q(z) + R(z)$ if $\deg(P) \geq \deg(D)$.
Preconditions Met: $(1) D(z) = 3z + 2 \neq 0$, (2) $\deg(P) = 4 > \deg(D) = 1$.
Subject: Algebra.

Theorem: Principle of Inclusion-Exclusion
Theorem Statement: For any two finite sets A and B, the size of their union is given by: $|A \cup B| = |A| + |B| - |A \cap B|$.
Question: Probability of palindrome (letters/digits) in plates, simplified as $\frac{m}{n}$, find $m + n$.
Intermediate Solution: Compute separately, combine using inclusion-exclusion.
Query: (Inclusion-Exclusion) $|A \cup B| = |A| + |B| - |A \cap B|$.
Preconditions Met: (1) Sets finite (letters/digits), potential overlap possible.
Subject: Combinatorics.

Theorem: Basic Multiplication Principle
Theorem Statement: If there are m ways to do something and n ways to do another thing, then there are m*n to do both things.
Question: Pages written/year if 3-page letters to 2 friends twice weekly?
Intermediate Solution: Pages/week calculation, then yearly total.
Query: (Multiplication Principle) Actions with m and n ways yield $m \times n$ combined ways.
Preconditions Met: Counts defined clearly; independent actions.
Subject: Arithmetic.

Figure 10: Few shot examples for MCTS Query Generation Action

Instruction

You are given a mathematical question, an intermediate solution and a mathematical theorem which was retrieved denoted as Retrieved Document. First, please judge whether the mathematical theorem is relevant with the question and the intermediate solution, and put it in the relevant field. If the provided content is irrelevant to the question and the context, explain the reason in the relevant reason field. The format will be as follows:
Question: [question]
Intermediate solution: [intermediate solution]
Retrieved Document: [theorem]
Relevant: [relevance label]
Reason: [reason].

Few Shot Examples

Question: Let $f : \mathbb{R} \to \mathbb{R}$ be continuous, with $f(0) = 0$ and $f(x + y) = f(x) + f(y) + xy$. Find degree of f.
Intermediate Solution: Define $g(x) = f(x) - \frac{x^2}{2}$.
Retrieved Document: Cauchy's equation $f(x + y) = f(x) + f(y)$, continuous solution linear: $f(x) = cx$.
Relevant: True
Reason: Defining $g(x)$ transforms into Cauchy's form, yielding $g(x) = cx$ and thus $f(x) = \frac{x^2}{2} + cx$ degree 2.

Question: Let $f : \mathbb{R} \to \mathbb{R}$ differentiable with $f'(x) = f(x) + x$. Find $f(x)$.
Intermediate Solution: Requires solving differential equation directly.
Retrieved Document: Rolle's Theorem guarantees $f'(c) = 0$ under certain continuity/differentiability conditions.
Relevant: False
Reason: There is no direct application of Rolle's Theorem to the differential equation provided, as the theorem does not help in finding the solution to the equation $f'(x) = f(x) + x$.

Question: Battery lifetime exponential mean 10 hours. Probability battery lasts at least 15 hours?
Intermediate Solution: Use exponential distribution's CDF.
Retrieved Document: Markov's Inequality provides upper bound $P(X \geq a) \leq \frac{E[X]}{a}$.
Relevant: False
Reason: Markov's gives bounds, not exact probabilities; exact CDF calculation is necessary here.

Figure 11: Prompt for Self Reflection MCTS

Instruction

Given the statement of a mathematical theorem in a structured latex format, convert it to a simpler natural language format by removing latex notations.

Few Shot Examples

Input ProofWiki Theorem (in latex):
<*Latex Section*>: Quadratic Irrational is Root of Quadratic Equation
<*Tags*>: Algebra, Quadratic Equations, Quadratic Irrationals
<*begin theorem*> Let x be a quadratic irrational. Then x is a solution to a quadratic equation with rational coefficients.

Generated Natural language theorem: Quadratic Irrational is Root of Quadratic Equation - A quadratic irrational number is always the root of some quadratic equation with rational coefficients.

Figure 12: Prompt for Self Summarization of Retrieved theorems

Instruction

Given a math question, its partial solution (may be empty), and a retrieved theorem, do the following: Identify the preconditions of the theorem and explain why they hold in the given question. Using these preconditions, generate a general retrieval query that captures the key mathematical idea needed in the partial solution. The query should be a single sentence.

Few Shot Examples

Question: A warehouse needs to store 65 identical boxes using a set of identical shelves, each of which can hold up to 8 boxes. What is the minimum number of shelves required to store all the boxes?

Partial solution: To determine the minimum number of shelves required, we divide the total number of boxes by the capacity of each shelf. Since the number of shelves must be a whole number, we round up to 9 shelves.

Theorem: If n items are put into m containers, with $n > m$, then at least one container must contain more than one item.

Preconditions: (1) There are more items than containers. (2) The items are distributed into containers.

Why Preconditions are Satisfied: (1) The warehouse has 65 boxes (items) and needs to distribute them among shelves (containers), where each shelf can hold up to 8 boxes. (2) Since 65 is greater than 8, multiple boxes must be placed on each shelf to store all of them.

Generated Query: Minimizing the number of boxes needed to store a given number of objects with fixed capacity constraints.

Figure 13: Prompt for LLM generated query generation