ReasonRank: Empowering Passage Ranking with Strong Reasoning Ability

Wenhan Liu[1], **Xinyu Ma**[2], **Weiwei Sun**[3], **Yutao Zhu**[1], **Yuchen Li**[2]
Dawei Yin[2] and **Zhicheng Dou**[1*]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Baidu Inc., Beijing, China [3]Carnegie Mellon University
lwh@ruc.edu.cn, xinyuma2016@gmail.com, dou@ruc.edu.cn

Abstract

Large Language Model (LLM) based listwise ranking has shown superior performance in many passage ranking tasks. With the development of Large Reasoning Models (LRMs), many studies have demonstrated that step-by-step reasoning during test-time helps improve listwise ranking performance. However, due to the scarcity of reasoning-intensive training data, existing rerankers perform poorly in many complex ranking scenarios, and the ranking ability of reasoning-intensive rerankers remains largely underdeveloped. In this paper, we first propose an automated reasoning-intensive training data synthesis framework, which sources training queries and passages from diverse domains and applies DeepSeek-R1 to generate high-quality training labels. To empower the listwise reranker with strong reasoning ability, we further propose a two-stage training approach, which includes a cold-start supervised fine-tuning (SFT) stage and a reinforcement learning (RL) stage. During the RL stage, we design a novel multi-view ranking reward tailored to the multi-turn nature of listwise ranking. Extensive experiments demonstrate that our trained reasoning-intensive reranker **ReasonRank** outperforms existing baselines significantly and also achieves much lower latency than the pointwise reranker.

1 Introduction

Passage ranking plays a crucial role in Information Retrieval (IR) by reranking initial retrieved passages to improve the quality of search results. Recently, Large Language Models (LLMs) have demonstrated impressive capabilities in zero-shot passage ranking (Sun et al., 2023). Among these LLM-based ranking approaches, listwise ranking has emerged as particularly effective, as it evaluates and ranks a list of passages simultaneously for a given query, capturing global relevance patterns more comprehensively than pointwise ranking (Zhuang et al., 2023) and pairwise ranking (Qin et al., 2023). It has demonstrated state-of-the-art (SOTA) performance on numerous IR benchmarks (Sun et al., 2023).

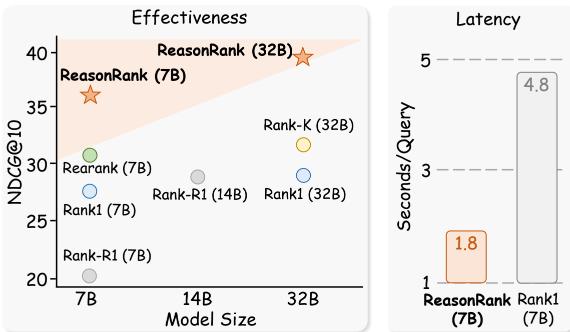

Figure 1: The left part shows the average NDCG@10 on the BRIGHT benchmark by reranking the top 100 passages retrieved by ReasonIR (8B). The right part compares the ranking latency of Rank1 (7B) and our ReasonRank (7B) on the Earth Science dataset.

Recently, test-time reasoning (DeepSeek-AI et al., 2025) has been shown to substantially improve LLM performance across a wide range of challenging NLP tasks. By explicitly generating step-by-step reasoning during inference, LLMs can produce more accurate answers. Such reasoning ability is also desirable for passage reranking, where understanding query intent and reasoning across multiple passages are critical to accurate ranking. Motivated by this, recent studies (Weller et al., 2025; Yang et al., 2025b) have attempted to inject reasoning capabilities into passage rerankers, resulting in improved performance compared to directly predicting final ranking results.

However, due to *scarcity of reasoning-intensive training data*, existing rerankers are primarily trained on traditional web search datasets such as MSMARCO (Nguyen et al., 2016), where relevance is often determined by shallow lexical or

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 22007–22031
July 2-7, 2026 ©2026 Association for Computational Linguistics

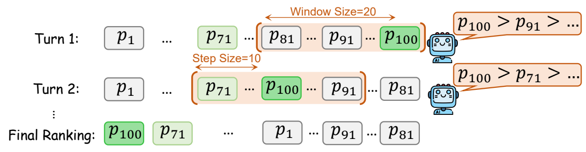

Figure 2: Ranking 100 passages based on sliding window strategy. Green squares represent relevant passages.

semantic matching. However, many real-world search platforms (*e.g.*, StackExchange) typically involve complex reasoning-intensive queries whose relevant passages are expected to provide key evidence or solutions with similar methodologies. As a result, rerankers trained on traditional web search datasets often fail to generalize to these complex search scenarios. Although collecting new training data via human annotation could alleviate this issue, it is prohibitively expensive and difficult to scale. These challenges create bottlenecks in training rerankers for complex search scenarios.

To alleviate this issue, we propose an automated data synthesis framework to construct high-quality reason-intensive training data without any human involvement. Specifically, we first collect diverse user queries from StackExchange platforms and well-established public datasets, covering four representative scenarios: complex QA, coding, math, and web search. Then we apply the strong reasoning model DeepSeek-R1 to automatically select positive passages and hard negative passages to construct training passages, and generate reasoning chains and gold ranking lists as the training labels. To ensure the quality of R1-generated training labels, we design a self-consistency data filtering mechanism that keeps only gold rankings where R1-selected positive passages are consistently ranked at top positions. Finally, we obtain 13K high-quality and diverse training data.

Based on our training data, we further propose a two-stage training framework to train our reasoning-intensive reranker ReasonRank. In the first stage, we apply cold-start SFT to teach the backbone LLM to reason over a list of passages and generate a reranked list. In the second stage, we apply RL to further optimize reranker's listwise ranking ability using task-specific rewards. Existing RL-based rerankers (Liu et al., 2025; Zhang et al., 2025) typically use ranking metrics (*e.g.*, NDCG) as rewards to evaluate reranked list. However, due to the limited context length of LLMs, existing listwise ranking approaches (Sun et al., 2023; Pradeep et al., 2023b) often adopt a sliding-window strategy which ranks a subset of passages iteratively to obtain final global ranking (see Figure 2). Under such strategy, higher NDCG on local ranking window does not guarantee that more relevant passages are recalled into later windows and therefore may fail to yield a high-NDCG global ranking (details will be discussed in Section 4.2). To address this limitation, we propose a multi-view ranking reward that considers local ranking metric while promoting the recall of relevant passages, thereby improving final ranking metric.

Extensive experiments on reasoning-intensive IR benchmarks BRIGHT (Su et al., 2025) and R2MED (Li et al., 2025) demonstrate the SOTA performance of ReasonRank. Notably, as shown in Figure 1, the 7B-scale and 32B-scale of our ReasonRank outperform the previous state-of-the-art model by 3 and 5 points on BRIGHT, respectively, and our ReasonRank performs more efficiently than the pointwise reranker Rank1 (Weller et al., 2025).

The contributions of the paper are:

(1) To address the data scarcity problem in training reasoning-intensive ranking models, we design an automated data synthesis framework and generate 13K high-quality and diverse reasoning-intensive training data.

(2) We propose a two-stage training framework, which includes a cold-start SFT strategy for reasoning pattern learning and a multi-view ranking reward-based RL approach for further ranking ability enhancement.

(3) Extensive experiments on the reasoning-intensive IR benchmarks BRIGHT and R2MED demonstrate the effectiveness and efficiency advantages of our ReasonRank.

2 Related Work

LLMs for Ranking The application of Large Language Models (LLMs) to ranking tasks has revolutionized traditional IR (Zhu et al., 2023, 2024; Sharifymoghaddam et al., 2025). Current LLM-based ranking methods can be categorized into three paradigms: pointwise, pairwise, and listwise methods. Pointwise methods (Liang et al., 2022; Sachan et al., 2022; Liu et al., 2024d; Fan et al., 2024; Huang et al., 2025; Liu et al., 2024c) evaluate each query-document pair independently. While computationally efficient, such a method lacks cross-document comparison. Pairwise methods (Qin et al., 2023) compare document pairs to establish relative relevance. Such a method usually suffers from efficiency issues due to the large number of pairwise comparisons. Listwise methods (Sun et al., 2023; Yoon et al., 2024; Liu et al., 2024b; Yoon et al., 2025; Chen et al., 2024; Liu et al., 2024a; Fan et al., 2025b) leverage LLM to rerank a passage list. Through global passage comparison, listwise ranking has achieved SOTA performance on many IR benchmarks.

Reasoning for IR Recently, test-time reasoning has been shown as a strong ability to improve LLMs and also applied to reasoning-intensive IR scenarios (Su et al., 2025; Shao et al., 2025; Qin et al., 2025; Fan et al., 2025a), particularly in passage ranking. For example, Rank1 (Weller et al., 2025) and Rank-K (Yang et al., 2025b) propose to distill the reasoning chain of Deepseek-R1 into rerankers. Zhuang et al. (2025) proposes using the RL algorithm GRPO to optimize a setwise reranker. While effective, they only use traditional web search datasets (*e.g.*, MSMARCO) for training, which makes rerankers perform poorly in reasoning-intensive benchmarks. In this paper, we propose to synthesize reasoning-intensive training data from diverse domains and propose a two-stage training approach to enhance ranking performance.

3 Preliminaries

Listwise Ranking Passage ranking aims to rerank a list of retrieved passages $[p_1, \ldots, p_N]$ based on their relevance to a query q. The listwise ranking approach takes both the query q and a passage list as input and outputs a reranked sequence of passage IDs (*e.g.*, [3] > [1] > . . .). Through comparing multiple passages, listwise ranking demonstrates SOTA performance in IR benchmarks (Sun et al., 2023). Due to the limited context length of many LLMs, listwise ranking usually applies a sliding window strategy (see Figure 2) to process a subset of passages iteratively. This strategy uses a window size w and step size s to promote relevant passage from back to the front. Following existing studies (Sun et al., 2023; Pradeep et al., 2023b), we set $N = 100$, $w = 20$, and $s = 10$ in this paper.

Reasoning-based Listwise Ranking Reasoning-based listwise reranker takes the query and a passage list as input, and produces a structured output that includes both reasoning traces (enclosed by tags <think> and </think>) and the reranked list such as "[3] > [1] > . . . " (enclosed by tags <answer> and </answer>).

4 Methodology

To train a reasoning-intensive reranker, we propose an automated data synthesis framework to collect high-quality reasoning-intensive training data and design a two-stage training framework for empowering the ranking ability.

4.1 Reasoning-intensive Data Synthesis

To mitigate the scarcity issue of reasoning-intensive training data, we introduce an automated framework for synthesizing training data. Existing reasoning-intensive IR benchmarks (Su et al., 2025; Li et al., 2025; Xiao et al., 2024) mainly evaluate queries from three main domains: **complex QA**, **math**, and **coding**. Following these benchmarks, we also use these three kinds of queries, as well as **web search** query (to ensure ReasonRank's performance on short search queries), to synthesize our training data. The overall process is shown in Figure 3. We further design a self-consistency data filtering mechanism to remove low-quality training data. Next, we will detail each part.

4.1.1 Complex QA

Complex QA queries are typically long and involve complex reasoning. We source complex QA training queries from StackExchange[1], which contains expert-level questions that require a deep understanding and complex reasoning. We select six sub-domains: Biology, Earth Science, Economics, Robotics, StackOverflow, and Sustainability.

Positive Passages Selection. Training data for listwise ranking usually consists of passage lists with relevant (positive) and irrelevant (negative) passages. Relevant passages for a complex QA query often contain key concepts that help answer the query. For StackExchange questions, relevant evidence often appears in external documents linked in accepted answers. To mine the positives of each query, we first obtain the gold answer that is accepted by the user and has at least one URL, crawl the linked document of each URL, and split them into passages. We then employ a strong model, Deepseek-R1 (R1), to perform listwise *positive passages selection*, which takes the query, gold answer, and a candidate passage list as input and outputs the IDs of positive passages (prompt in Figure 6). This yields the positive set P^+, with the remaining passages treated as negatives P^-. Note that providing a passage list enables

[1] https://archive.org/download/stackexchange

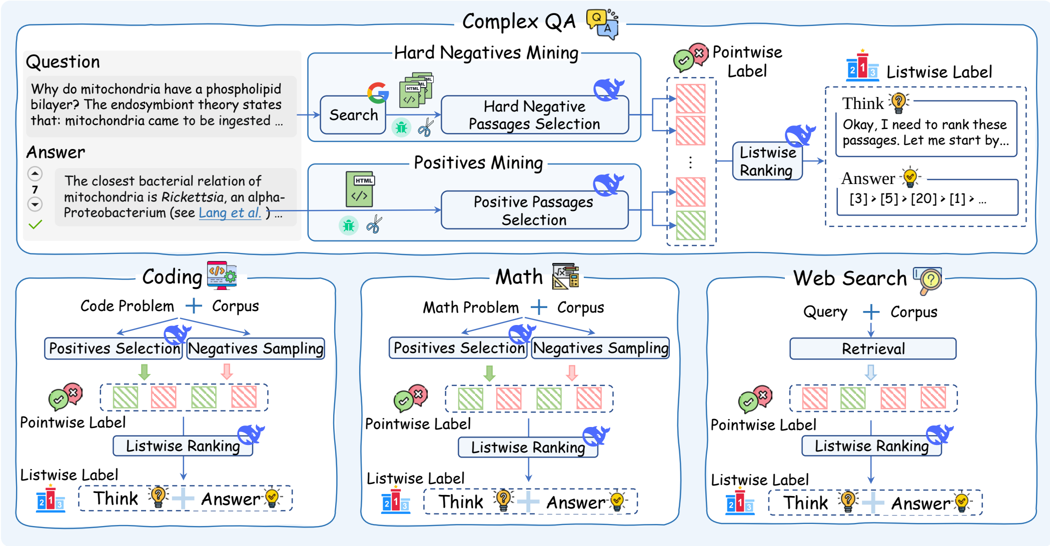

Figure 3: An overview of reasoning-intensive ranking data synthesis on four domains.

R1 to perform comparative relevance assessment, while the gold answer helps understand the query and judge the relevance.

Hard Negative Passages Selection. To further enrich the passage list, we mine hard negatives—passages that are topically related to the query yet provide no value for answering it. Specifically, we retrieve top-10 documents using the Google Search API, split them into passages, and apply R1 to perform listwise *hard negative passages selection* (prompt in Figure 7), producing a hard negative set P^-_{hard}.

Listwise Ranking. Finally, we combine all the collected passages P^+, P^-, and P^-_{hard} into a passage list capped at 20 passages (*i.e.*, the sliding window size) and apply R1 for *listwise ranking*, generating both reasoning chains and reranked lists (prompt in Figure 8). So far, we have obtained two types of training labels: (1) pointwise binary labels (1 for P^+ and 0 for P^- and P^-_{hard}) and (2) listwise labels with reasoning chains and gold rankings. Both training labels will be used in our two-stage training approach. We do not provide the gold answer when generating reasoning chains with R1, to avoid reasoning chains' reliance on the gold answer and ensure consistency with ReasonRank training (same for the following coding and math domains).

4.1.2 Coding

For coding-based queries, relevant passages typically share the same algorithmic strategy or similar logic. We use coding problems from the LeetCode dataset[2] as training queries, and adopt the corpus of the LeetCode dataset from the BRIGHT benchmark to mine positives and negatives.

Mining positives for coding-type queries is nontrivial, as relevant passages may not exhibit high lexical similarity to the query. To mine positives, we first retrieve a candidate set of $K = 40$ passages using a strong dense retriever E5-Mistral-7B-Instruct (Wang et al., 2024), and then apply R1 for positive passage selection (prompt in Figure 9). We sample negatives from the remaining passages and combine them with the selected positives to form a passage list of 20 passages. Finally, we apply R1 for listwise ranking to obtain the listwise label, including a reasoning chain and a gold ranking list.

4.1.3 Math

Math queries often require retrieving problems with similar solutions or related theorems. Following BRIGHT, we consider two math passage ranking tasks with different relevance definitions: (1) **Math-Problem**, which ranks problem–solution pairs based on similarity in problem-solving logic to the query, and (2) **Math-Theorem**, which ranks mathematical theorems according to whether they are applicable for solving the query. We use math problems from the MATH dataset (Hendrycks et al., 2021) as training queries. For the Math-Problem task, we adopt the STEM problem–solution corpus constructed by (Su et al., 2025); for the Math-Theorem task, we use 20K theorem statements

[2] https://huggingface.co/datasets/greengerong/leetcode

from ProofWiki[3] as the passage corpus.

Following the same procedure as in coding queries, we retrieve the top-K passages using E5-Mistral-7B-Instruct, apply R1 for positive passage selection (prompts in Figure 10 and 11), sample negatives to construct passage lists, and finally use R1 for listwise ranking to obtain listwise labels.

4.1.4 Web Search

Beyond the above three domains, we also incorporate web search queries to ensure the model's ranking ability in simple search tasks. We sample 4K queries from the MSMARCO training set whose passages have been annotated with pointwise relevance labels. Following previous studies (Pradeep et al., 2023a; Liu et al., 2024b), we use BM25 to retrieve top-20 passages and apply R1 for listwise ranking to obtain listwise labels.

4.1.5 Self-Consistency Data Filtering

Through the above data synthesis process, we have obtained our training queries, passage lists, and corresponding labels – specifically, the **pointwise labels and listwise labels** generated by R1. To ensure the quality of training labels, we propose a self-consistency data filtering mechanism. Inspired by the LLM self-consistency framework proposed in (Wang et al., 2023), we hypothesize that the labels derived from R1 with higher self-consistency should correspond to higher label quality. Thus, we compute the ranking metric NDCG@10 for the gold ranking lists from the listwise labels, using the pointwise labels as ground truths. We then remove all training samples with NDCG@10 values below a predefined threshold α. Through this process, we finally obtain a quality-filtered dataset for subsequent model training. Detailed information about the dataset is shown in Table 5.

4.2 Two-stage Training Framework

In this section, based on our synthesized data, we present a two-stage training approach (see Figure 4) to empower our reasoning-intensive listwise reranker with strong ranking ability. Specifically, we first utilize the listwise labels to perform supervised fine-tuning (SFT) on the backbone LLM, endowing it with initial reasoning capability for listwise ranking tasks. Then, we design a novel multi-view ranking reward tailored to the characteristics of sliding-window-based listwise ranking, and apply reinforcement learning (RL) to further enhance the reranker's ranking performance.

Cold-Start SFT To enable the LLM to learn reasoning for listwise ranking, we leverage the listwise labels, which contain both the reasoning chains and the reranked passage list, for SFT. The input to the backbone LLM comprises a query paired with a passage list. The training process optimizes the model by minimizing the standard language modeling loss, as formalized in the equation below:

$$\mathcal{L} = -\sum_{i=1}^{|y|} \log(P_\theta(y_i \mid x, y_{<i})), \tag{1}$$

where x and y represent the input prompt and the listwise label, respectively. The input prompt is shown in Figure 12.

Multi-view Ranking based RL In this section, we employ RL to further help the reranker discern subtle ranking differences and optimize for ranking metric.

The reward signal serves as the optimization objective that directly influences the policy model's training efficacy. Previous studies (Liu et al., 2025; Zhang et al., 2025) merely adopt the ranking metric NDCG@10 as the reward signal. However, as we mentioned in Section 1, such a *single-turn reward* is suboptimal for sliding window-based listwise ranking. This is because the sliding window strategy entails *multi-turn sequential ranking*, where maximizing NDCG@10 for a local ranking window does not necessarily ensure the optimal NDCG@10 of the final ranking. For example, with window size of 20 and step size of 10, ranking two relevant passages (among 20 passages) at positions 1 and 11 yields a higher NDCG@10 score (0.61) than placing them at positions 9 and 10 (0.36). Nevertheless, ranking these passages at positions 9 and 10 ensures both remain within the top-10 and propagate to subsequent windows, potentially leading to better NDCG@10 in the final ranking.

Thus, in addition to NDCG@10, we propose incorporating the metric Recall@10 as part of our ranking reward. Recall@10 is computed as the proportion of relevant passages retrieved in the top 10 positions of the current ranking window. Moreover, compared to using pointwise labels for NDCG@10 calculation, we contend that the gold list in our listwise labels contains more granular ranking signals. Consequently, we adopt the Rank-Biased Overlap

[3] https://proofwiki.org

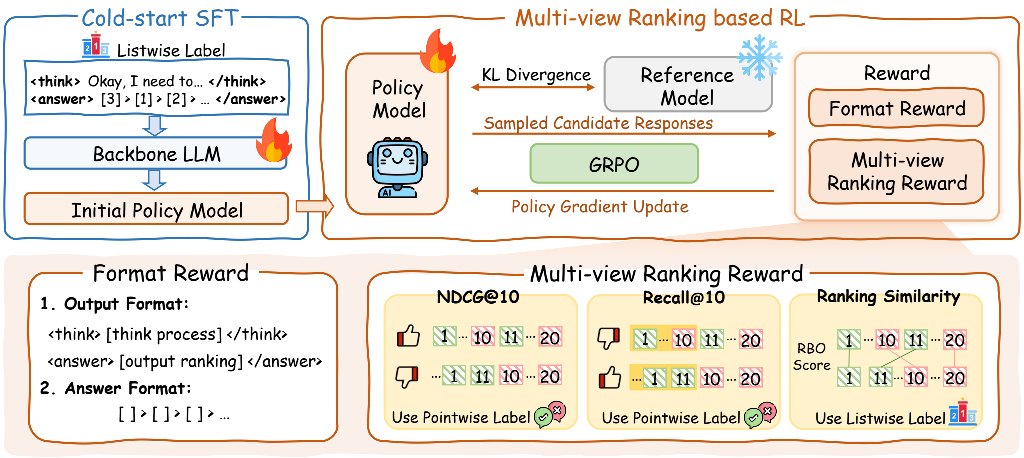

Figure 4: An overview of our two-stage training framework.

(RBO) metric (Webber et al., 2010), which measures listwise ranking similarity, as an additional supplementary ranking reward. The formula is shown in Equation (4).

A combination of NDCG@10, Recall@10, and RBO with pre-defined parameters ϕ and γ forms our **multi-view ranking reward** R^{m}:

$$R^{\mathrm{m}} = \mathrm{NDCG@10} + \phi \cdot \mathrm{Recall@10} + \gamma \cdot \mathrm{RBO}. \quad (2)$$

Furthermore, to ensure a proper output structure, we implement format rewards considering two types of formats: (1) Output format, which ensures the presence of both the <think> and <answer> tags; and (2) Answer format, which validates that the content enclosed in the <answer> tags adheres to the specified ranking format (*e.g.*, [4] > [2] ...).

The final reward R is computed as follows:

$$R = \begin{cases} R^{\mathrm{m}}, & \text{Both formats are correct,} \\ 0, & \text{Only answer format is incorrect,} \\ -1, & \text{Output format is incorrect.} \end{cases} \quad (3)$$

We employ the GRPO algorithm (Shao et al., 2024) for optimization with the reward R. Details of the GRPO algorithm are shown in Appendix B.2.

5 Experiment

5.1 Experimental Setup

Evaluation Datasets We choose two reasoning-intensive IR benchmarks, BRIGHT (Su et al., 2025) and R2MED (Li et al., 2025), for evaluation. BRIGHT consists of 12 datasets from diverse domains and is widely used to test the reasoning-intensive IR models. R2MED is a benchmark designed for reasoning-driven medical retrieval, which contains 8 datasets.

As for the BRIGHT benchmark, we use a strong reasoning dense model ReasonIR (8B) (Shao et al., 2025) as the initial retriever. Following previous studies (Shao et al., 2025; Yang et al., 2025b), we use the GPT4-rewritten queries (provided in the test set) as test queries due to better retrieval performance and use the original test query for passage reranking. As for the R2MED benchmark, we use E5-Mistral-7B-Instruct as the initial retrievers and original test query for retrieval and reranking. We rerank the top 100 retrieved passages and use NDCG@10 as the evaluation metric.

Baselines We compared two types of rerankers: non-reasoning rerankers and reasoning rerankers. As for non-reasoning rerankers, we choose RankT5 (3B) (Zhuang et al., 2023) and RankZephyr (7B) (Pradeep et al., 2023b). As for reasoning rerankers, we compare with Rank1 (7B, 32B) (Weller et al., 2025), Rank-R1 (7B, 14B) (Zhuang et al., 2025), Rearank (7B) (Zhang et al., 2025) and Rank-K (32B) (Yang et al., 2025b). The baseline details are shown in Appendix A.2.

We use Qwen2.5-7B-Instruct and Qwen2.5-32B-Instruct to train our ReasonRank. Experiments based on other backbone LLMs are provided in Section C.3. Due to space limitations, we put the whole implementation details in the Appendix B.

5.2 Main Results

We evaluate our ReasonRank (7B and 32B) on BRIGHT and R2MED, and show the results in Table 1 and Table 2, respectively. From the results, we have the following observations:

(1) Our ReasonRank (7B and 32B) demonstrates superior performance compared with all baselines on the average of two benchmarks. No-

Models	Econ.	Earth.	Rob.	Bio.	Psy.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
ReasonIR (8B)	32.65	43.00	20.82	43.49	39.57	30.96	27.34	31.69	19.55	7.37	33.93	36.68	30.59
Non-reasoning reranker													
rankT5 (3B)	11.35	22.11	10.94	13.62	11.40	11.35	15.96	27.45	38.05	9.24	18.27	9.46	16.60
RankZephyr (7B)	19.87	17.36	12.35	34.90	24.72	13.35	22.34	29.29	32.37	6.05	28.98	30.07	22.64
Reasoning reranker													
Rank-R1 (7B)	20.19	27.91	18.13	36.70	30.22	11.32	29.28	17.13	9.35	3.23	14.43	28.98	20.57
Rank-R1 (14B)	27.39	38.73	23.11	44.45	37.10	27.82	36.77	21.27	19.23	8.80	31.66	39.53	29.66
Rank1 (7B)	25.34	38.91	16.77	39.91	35.32	24.81	33.47	12.71	28.14	2.58	30.67	38.18	27.23
Rank1 (32B)	25.43	37.97	17.14	42.43	34.85	23.82	31.18	12.20	**40.98**	4.79	29.28	39.95	28.34
Rearank (7B)	30.65	40.27	27.16	46.94	40.77	26.12	36.26	**30.55**	22.84	7.31	32.28	39.86	31.75
Rank-K (32B)	30.09	39.78	26.60	50.64	43.45	29.87	35.17	27.23	22.80	7.61	37.06	41.00	32.61
ReasonRank (7B)	35.05	47.75	31.22	56.70	47.77	32.47	40.87	23.17	24.95	7.68	**39.49**	41.80	35.74
ReasonRank (32B)	**36.64**	**48.90**	**33.88**	**58.17**	**53.27**	**38.68**	**45.97**	25.78	20.93	**9.41**	38.46	**46.28**	**38.03**

Table 1: The results (NDCG@10) on the BRIGHT benchmark. All baselines rerank the top 100 passages retrieved by ReasonIR. The top two rerankers are highlighted in **bold** and underlined.

Models	Biology	Bioin.	MedS.	MedE.	MedD.	PMCT.	PMCC.	IIYiC.	Avg.
E5-mistral (7B)	18.28	41.47	41.01	6.44	11.38	19.81	30.97	21.37	23.84
Non-reasoning reranker									
RankT5 (3B)	13.20	32.84	23.38	2.09	4.18	0.61	14.55	12.39	12.91
RankZephyr (7B)	22.86	43.07	48.24	6.97	10.45	26.64	7.78	14.59	22.58
Reasoning reranker									
Rank-R1 (7B)	34.04	51.63	50.96	12.79	21.96	34.79	31.70	25.08	32.87
Rank-R1 (14B)	38.78	53.81	57.94	15.15	25.49	40.57	42.52	29.63	37.99
Rank1 (7B)	32.64	55.57	54.74	12.78	19.98	34.40	30.17	18.15	32.30
Rank1 (32B)	31.84	61.65	59.74	16.56	26.90	**41.31**	45.56	29.49	39.13
Rearank (7B)	38.35	50.91	59.73	14.06	19.34	37.49	34.29	26.27	35.06
Rank-K (32B)	32.94	53.94	51.47	11.41	22.93	34.44	38.38	26.24	33.97
ReasonRank (7B)	**46.80**	59.70	60.11	16.48	24.92	39.21	39.13	**29.85**	39.53
ReasonRank (32B)	45.56	**67.73**	**63.45**	**18.90**	**30.60**	41.08	**46.11**	29.35	**42.85**

Table 2: The results (NDCG@10) on R2MED benchmark. All models rerank E5-mistral-retrieved top-100 passages.

tably, our ReasonRank (32B) outperforms the best baseline Rank-K (32B) on BRIGHT by about 5 points and Rank1 (32B) on R2MED by about 4 points. Besides, our ReasonRank (7B) even outperforms the 32B-scale baselines significantly. For example, it surpasses Rank-K (32B) by 3 and 9 points on BRIGHT and R2MED, respectively. These results demonstrate the effectiveness of our ranking data synthesis and two-stage training framework.

(2) Existing baselines struggle in reasoning-intensive reranking. In the BRIGHT benchmark, the baselines, except for Rank-K (32B), can hardly improve the initial retrieval results. In the R2MED benchmark, the two non-reasoning rerankers underperform the retriever E5-Mistral on the Avg. metric. This suggests that traditional training data and existing training methods struggle to produce an effective reasoning-intensive reranker.

(3) The performance of the reranker scales with the model size. For example, on the BRIGHT benchmark, Rank-R1 (32B) exceeds Rank-R1 (7B) by 9 points (Avg.), and our ReasonRank (32B) exceeds ReasonRank (7B) by about 2.3 points (Avg.). This indicates that larger models have stronger reasoning and ranking capabilities.

5.3 Ablation Study

We conduct ablation studies to evaluate the contribution of different components from three aspects: (1) training data, (2) training approach, and (3) *w/o* reasoning. Experiments are conducted on the BRIGHT benchmark using ReasonRank (7B). Overall results are reported in Table 3, with per-dataset results and detailed analyses provided in Table 6 and Appendix C.1. We summarize our findings as follows.

(1) Training Data. Training on only the MS-MARCO subset leads to a substantial performance drop (5.66 points), highlighting the importance of diverse, reasoning-intensive training data. Removing self-consistency filtering (*w/o* Self-Consistency) also degrades performance, demonstrating its effectiveness.

(2) Training Approach. We ablate the two-

Models	BRIGHT Average	
ReasonRank (7B)	**35.74**	-
Training data		
only MSMARCO	30.08	-5.66
w/o Self-Consistency	34.20	-1.54
Training approach		
w/o SFT (only RL)	28.69	-7.05
w/o RL (only SFT)	33.15	-2.59
w/o R^m	34.20	-1.54
w/o reasoning		
Only SFT	32.96	-2.78
SFT + RL	33.60	-2.14

Table 3: Ablation study based on ReasonRank (7B).

stage training framework by removing cold-start SFT ("*w/o* SFT (only RL)") and RL ("*w/o* RL (only SFT)"). Removing cold-start SFT leads to a substantial performance drop (7.05 points), highlighting its critical role in enabling the model to acquire reasoning ability for listwise ranking. Removing RL also degrades performance (2.59 points), indicating additional gains from reinforcement learning. Moreover, replacing the proposed multi-view ranking reward with NDCG@10 alone ("*w/o* R^m") further harms performance, validating the effectiveness of our reward design.

(3) W/o Reasoning. To validate the effectiveness of reasoning, we train non-reasoning rerankers that directly output rankings using only the gold rankings in our training data. We experimented with two training strategies: only SFT and SFT + RL (*i.e.*, GRPO). From the table, both approaches (32.96 and 33.60) perform significantly lower than our ReasonRank (35.74), confirming that explicit reasoning is essential for listwise ranking.

5.4 Traditional IR Benchmark

To evaluate the generalization of ReasonRank, we conduct experiments on the traditional IR benchmark BEIR (Thakur et al., 2021). Considering the large size of some test sets in BEIR, following the previous study (Sun et al., 2023), we selected 7 datasets with a smaller number of queries from BEIR. We choose BM25 as our retriever and compare it with several competitive baselines: RankZephyr (7B), Rank-R1 (14B), Rank1 (32B), and Rank-K (32B). From the results in Table 4, we can see that ReasonRank (32B) outperforms all baselines, demonstrating its strong generalization ability on the traditional IR benchmark. Besides, the gap of our ReasonRank over the baselines is smaller than that on BRIGHT and R2MED. This

Models	BEIR Average	
BM25	43.74	-
RankZephyr (7B)	54.14	+10.40
Rank-R1 (14B)	54.61	+10.87
Rank1 (32B)	50.99	+7.25
Rearank (7B)	54.29	+10.54
Rank-K (32B)	48.34	+4.60
ReasonRank (7B)	54.35	+10.61
ReasonRank (32B)	**55.44**	**+11.70**

Table 4: Averaged NDCG@10 on 7 BEIR datasets, including Covid, DBPedia, SciFact, NFCorpus, Signal, Robust04, and News.

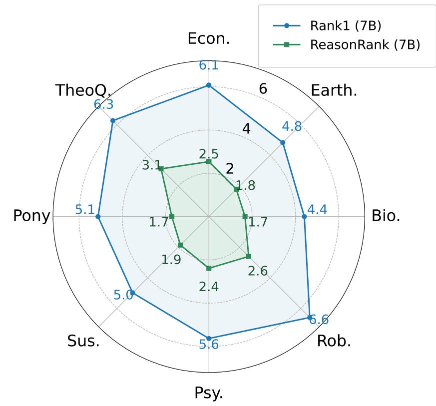

Figure 5: Ranking latency (seconds per query) of Rank1 (7B) and ReasonRank (7B) on eights datasets.

might be because the queries in BEIR are much simpler and do not require complex reasoning. Results for each dataset are shown in Table 7.

5.5 Efficiency Analysis

ReasonRank improves ranking performance through reasoning but also introduces additional latency. In this section, we test the time latency of ReasonRank (7B) and compare it with the pointwise reranker Rank1 (7B). We select 8 datasets from BRIGHT and rerank the top-100 ReasonIR-retrieved passages on 4*A800 80G GPUs. As shown in Figure 5, surprisingly, our listwise ReasonRank is 2-2.7× faster than pointwise Rank1, which is contrary to conclusions from non-reasoning rerankers (Zhuang et al., 2024). This efficiency stems from Rank1 generating a reasoning chain for each passage, while ReasonRank processes multiple passages at a time with only one reasoning chain, significantly decreasing the number of output tokens.

6 Conclusion

In this paper, we propose ReasonRank, a state-of-the-art reasoning-intensive passage reranker. To train ReasonRank, we first propose a reasoning-intensive ranking data synthesis framework and then design a two-stage training framework, which includes a cold-start SFT and a multi-view ranking-based RL. Extensive experiments demonstrate the superior performance as well as the efficiency of our ReasonRank.

Limitations

Despite the superior performance achieved by ReasonRank, we believe that our work still has some limitations: (1) ReasonRank does not include non-reasoning type data during training, which results in its inability to seamlessly switch between reasoning and non-reasoning modes when faced with search scenarios of varying difficulty. In the future, we plan to introduce non-reasoning type data into the training set to enhance ReasonRank's flexibility in handling different scenarios (like Qwen3 (Yang et al., 2025a)). (2) ReasonRank still relies on the sliding window strategy for passage reranking. Existing studies (Liu et al., 2024b) have demonstrated that LLMs have strong full-list ranking capabilities (i.e., directly ranking 100+ passages in one forward pass), which exhibit both superior efficiency and effectiveness compared to sliding window approaches. In future work, we plan to explore reasoning-intensive listwise reranking based on full ranking to further improve the scalability and performance of our framework.

Acknowledgments

The work was supported by National Natural Science Foundation of China No. 62272467, and was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

References

Yiqun Chen, Qi Liu, Yi Zhang, Weiwei Sun, Daiting Shi, Jiaxin Mao, and Dawei Yin. 2024. Tourrank: Utilizing large language models for documents ranking with a tournament-inspired strategy. *CoRR*, abs/2406.11678.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *CoRR*.

Debrup Das, Seán Ó Nualláin, and Razieh Rahimi. 2025. Rader: Reasoning-aware dense retrieval models. *CoRR*, abs/2505.18405.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, and S. S. Li. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948.

Guanting Dong, Yifei Chen, Xiaoxi Li, Jiajie Jin, Hongjin Qian, Yutao Zhu, Hangyu Mao, Guorui Zhou, Zhicheng Dou, and Ji-Rong Wen. 2025a. Tool-star: Empowering llm-brained multi-tool reasoner via reinforcement learning. *CoRR*, abs/2505.16410.

Guanting Dong, Hangyu Mao, Kai Ma, Licheng Bao, Yifei Chen, Zhongyuan Wang, Zhongxia Chen, Jiazhen Du, Huiyang Wang, Fuzheng Zhang, et al. 2025b. Agentic reinforced policy optimization. *arXiv preprint arXiv:2507.19849*.

Yongqi Fan, Xiaoyang Chen, Dezhi Ye, Jie Liu, Haijin Liang, Jin Ma, Ben He, Yingfei Sun, and Tong Ruan. 2025a. Tfrank: Think-free reasoning enables practical pointwise llm ranking. *arXiv preprint arXiv:2508.09539*.

Yongqi Fan, Kui Xue, Zelin Li, Xiaofan Zhang, and Tong Ruan. 2025b. An llm-based framework for biomedical terminology normalization in social media via multi-agent collaboration. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 10712–10726.

Yongqi Fan, Yansha Zhu, Kui Xue, Jingping Liu, and Tong Ruan. 2024. Rrnorm: A novel framework for chinese disease diagnoses normalization via llm-driven terminology component recognition and reconstruction. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 9162–9175.

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. In *NeurIPS Datasets and Benchmarks*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *CoRR*, abs/2106.09685.

Zhaoheng Huang, Yutao Zhu, Zhicheng Dou, and Ji-Rong Wen. 2025. CAGS: context-aware document ranking with contrastive graph sampling. *IEEE Trans. Knowl. Data Eng.*, 37(1):89–101.

Lei Li, Xiao Zhou, and Zheng Liu. 2025. R2MED: A benchmark for reasoning-driven medical retrieval. *CoRR*, abs/2505.14558.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel J. Orr, Lucia Zheng, Mert Yüksekgönül, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri S. Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. 2022. Holistic evaluation of language models. *CoRR*, abs/2211.09110.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2024a. Leveraging passage embeddings for efficient listwise reranking with large language models. *CoRR*, abs/2406.14848.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Lixin Su, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025. Coranking: Collaborative ranking with small and large ranking agents. *CoRR*, abs/2503.23427.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024b. Sliding windows are not the end: Exploring full ranking with long-context large language models. *CoRR*, abs/2412.14574.

Wenhan Liu, Yujia Zhou, Yutao Zhu, and Zhicheng Dou. 2024c. How to personalize and whether to personalize? candidate documents decide. *Knowledge and Information Systems*, (9):5581–5604.

Wenhan Liu, Yutao Zhu, and Zhicheng Dou. 2024d. Demorank: Selecting effective demonstrations for large language models in ranking task. *CoRR*, abs/2406.16332.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023a. Rankvicuna: Zero-shot listwise document reranking with open-source large language models. *CoRR*, abs/2309.15088.

Ronak Pradeep, Sahel Sharifymoghaddam, and Jimmy Lin. 2023b. Rankzephyr: Effective and robust zero-shot listwise reranking is a breeze! *CoRR*, abs/2312.02724.

Xubo Qin, Jun Bai, Jiaqi Li, Zixia Jia, and Zilong Zheng. 2025. Tongsearch-qr: Reinforced query reasoning for retrieval. *CoRR*, abs/2506.11603.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2023. Large language models are effective text rankers with pairwise ranking prompting. *CoRR*, abs/2306.17563.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. KDD '20.

Devendra Singh Sachan, Mike Lewis, Mandar Joshi, Armen Aghajanyan, Wen-tau Yih, Joelle Pineau, and Luke Zettlemoyer. 2022. Improving passage retrieval with zero-shot question generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 3781–3797. Association for Computational Linguistics.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and Luke Zettlemoyer. 2025. Reasonir: Training retrievers for reasoning tasks. *CoRR*, abs/2504.20595.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Sahel Sharifymoghaddam, Ronak Pradeep, Andre Slavescu, Ryan Nguyen, Andrew Xu, Zijian Chen, Yilin Zhang, Yidi Chen, Jasper Xian, and Jimmy Lin. 2025. Rankllm: A python package for reranking with llms. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3681–3690.

Huatong Song, Jinhao Jiang, Wenqing Tian, Zhipeng Chen, Yuhuan Wu, Jiahao Zhao, Yingqian Min, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. 2025. R1-searcher++: Incentivizing the dynamic knowledge acquisition of llms via reinforcement learning. *CoRR*, abs/2505.17005.

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

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is chatgpt good at search? investigating large language models as re-ranking agents. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 14918–14937. Association for Computational Linguistics.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Improving text embeddings with large language models. In *ACL (1)*, pages 11897–11916. Association for Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*. OpenReview.net.

William Webber, Alistair Moffat, and Justin Zobel. 2010. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28(4):20:1–20:38.

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

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jian Yang, Jiaxi Yang, Jingren Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. 2025a. Qwen3 technical report. *CoRR*, abs/2505.09388.

Eugene Yang, Andrew Yates, Kathryn Ricci, Orion Weller, Vivek Chari, Benjamin Van Durme, and Dawn J. Lawrie. 2025b. Rank-k: Test-time reasoning for listwise reranking. *CoRR*, abs/2505.14432.

Soyoung Yoon, Eunbi Choi, Jiyeon Kim, Hyeongu Yun, Yireun Kim, and Seung-won Hwang. 2024. Listt5: Listwise reranking with fusion-in-decoder improves zero-shot retrieval. In *ACL (1)*, pages 2287–2308. Association for Computational Linguistics.

Soyoung Yoon, Gyuwan Kim, Gyu-Hwung Cho, and Seung-won Hwang. 2025. Acurank: Uncertainty-aware adaptive computation for listwise reranking. *CoRR*, abs/2505.18512.

Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. 2025. REARANK: reasoning re-ranking agent via reinforcement learning. *CoRR*, abs/2505.20046.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. *CoRR*, abs/2403.13372.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *CoRR*, abs/2308.07107.

Yutao Zhu, Peitian Zhang, Chenghao Zhang, Yifei Chen, Binyu Xie, Zheng Liu, Ji-Rong Wen, and Zhicheng Dou. 2024. INTERS: unlocking the power of large language models in search with instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 2782–2809. Association for Computational Linguistics.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023. Rankt5: Fine-tuning T5 for text ranking with ranking losses. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023*, pages 2308–2313. ACM.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2025. Rank-r1: Enhancing reasoning in llm-based document rerankers via reinforcement learning. *CoRR*, abs/2503.06034.

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *SIGIR*, pages 38–47. ACM.

A Benchmarks and Baselines

A.1 Benchmarks

In our experiments, we utilize three IR benchmarks for evaluation: BRIGHT (Su et al., 2025), R2MED (Li et al., 2025), and BEIR (Thakur et al., 2021). Each of these benchmarks is instrumental in assessing different aspects of our retrieval models, ensuring a comprehensive evaluation across varied scenarios. The detailed introduction of each benchmark is as follows:

- BRIGHT: The BRIGHT benchmark is designed to challenge information retrieval systems with complex queries that necessitate deep reasoning capabilities. Unlike traditional benchmarks that focus on simple keyword or semantic matches, BRIGHT comprises 1,384 queries drawn from diverse domains such as economics, psychology, mathematics, and coding. These queries are sourced from both naturally occurring data and curated human inputs, making the benchmark a robust test for advanced retrieval models. The requirement for thorough reasoning to identify relevant documents makes BRIGHT particularly demanding, as it pushes models to go beyond surface-level matching and engage in deeper cognitive processing.

- R2MED: R2MED is a specialized benchmark for medical information retrieval, emphasizing the importance of reasoning in retrieving relevant medical information. It features 876 queries that cover three main tasks: Q&A reference retrieval, clinical evidence retrieval, and clinical case retrieval. These tasks span a wide array of medical scenarios and body systems, highlighting the necessity for retrieval systems to understand and process medical information accurately. R2MED underscores the critical role reasoning plays in medical decision-making, challenging retrieval systems to move beyond simple lexical or semantic matching to support complex clinical judgments and evidence-based practice.

- BEIR: BEIR is a comprehensive benchmark designed to evaluate the performance of information retrieval systems across 18 diverse datasets. These datasets encompass a variety of tasks, including fact-checking, question answering, and more, allowing for a thorough assessment of a model's generalization capabilities. BEIR is particularly focused on the ability of retrieval systems to adapt across different domains and tasks, emphasizing the need for a balance between performance and computational efficiency. By encouraging the development of robust and adaptable retrieval systems, BEIR serves as a pivotal benchmark in the advancement of information retrieval technologies.

A.2 Baselines

The baselines we used for comparison are as follows, each selected for its unique strengths and capabilities in specific retrieval or reranking tasks:

• **ReasonIR** (Shao et al., 2025): ReasonIR (8B) is a reasoning-intensive retriever training on reasoning-intensive synthetic data based on LLAMA3.1-8B (Touvron et al., 2023). This baseline is particularly adept at handling complex queries that require in-depth reasoning, making it an ideal choice as our initial retriever on the BRIGHT benchmark.

• **E5-Mistral-7B-Instruct** (Wang et al., 2024): E5-Mistral-7B-Instruct[4] is fine-tuned on a mixture of multilingual datasets and has some multilingual capability. We use it as our initial retriever on the R2MED benchmark.

• **RankT5** (Zhuang et al., 2023): RankT5 is a reranker that leverages the T5 architecture in a pointwise manner, optimized through a ranking loss function.

• **RankZephyr** (Pradeep et al., 2023b): This listwise re-ranker is distilled from the powerful models GPT-3.5 and GPT-4, utilizing a sophisticated dual-phase training framework. During inference, it employs a sliding window method for listwise reranking.

• **Rank1** (Weller et al., 2025): Rank1 functions as a pointwise reasoning reranker, distilled from the R1 model. It computes relevance scores by evaluating the probability of the final token being true or false, providing a nuanced approach to ranking that is particularly effective in scenarios requiring detailed reasoning.

• **Rank-R1** (Zhuang et al., 2025): Rank-R1 is a setwise reasoning reranker developed through GRPO reinforcement learning. In the inference phase, it identifies the most pertinent passage from a set and employs heap sort to reorder all passages.

[4] https://huggingface.co/intfloat/e5-mistral-7b-instruct

Category	Dataset Name	Data Num
Complex QA	Biology	1700
	Earth Science	566
	Economics	787
	Robotics	451
	Sustainable Living	147
	Stackoverflow	1741
Coding	Leetcode	1633
Math	Math-QA	1726
	Math-Theorem	1673
Web Search	MS MARCO	3093

Table 5: The statistics of our synthesized reasoning-intensive training data (totally 13.5k) on four domains, including Complex QA, Coding, Math, and Web search.

• **Rearank** (Zhang et al., 2025): Rearank is a listwise reasoning reranker trained using reinforcement learning.

• **Rank-K** (Yang et al., 2025b): Rank-K represents a listwise reasoning reranker distilled from R1, trained on the MSMARCO dataset with QwQ 32B as its backbone. It utilizes a sliding window strategy during inference to rearrange all passages, which allows it to achieve a trade-off between ranking effectiveness and efficiency.

B Implementation Details

To help researchers better understand our approach, we have provided a comprehensive and detailed explanation of the implementation details of the Reasoning-intensive Ranking Data Synthesis and the Two-Stage Training Framework here.

B.1 Details about Reasoning-intensive Ranking Data Synthesis

In this part, we will introduce the construction of our training data. After constructing our data from multiple domains, we apply a self-consistency data filtering mechanism to filter low-quality training data. The parameter α is set as 0.4, which is determined by grid search with a step of 0.1 in the range of [0.1, 0.7] on our validation set. The statistics of final training data are shown in Table 5. Next, we will elaborate on the details of constructing data for each domain.

Complex QA When splitting long documents into passages, we use simple heuristics with separators such as double new lines, without making extra assumptions about the file structure. This approach allows for flexibility and adaptability to various document formats, ensuring that the passages remain coherent and contextually meaningful.

When using R1 to perform listwise positive passages selection, we input a list of candidate passages, the query, and its gold answer to R1, and output the IDs of positives. The prompt is shown in Figure 6.

As for the hard negative passages selection by R1, we input a list of candidate passages, the query, and its gold answer to the model, and output the IDs of hard negatives. The prompt we used is shown in Figure 7:

After obtaining the training candidate passages, we use R1 to perform listwise ranking on these passages using the prompt in Figure 8 (the same for the subsequent data synthesis for coding, math, and web search):

Coding As for coding queries, we apply the prompt in Figure 9 for mining positives:

Math We have two different ranking tasks for math queries, including Math-QA and Math-Theorem. We use the prompts shown in Figure 10 and Figure 11 to select positives for Math-QA and Math-Theorem tasks based on R1, respectively.

Web Search For web search, we sample 4k training queries from the MS MARCO (Nguyen et al., 2016) training set and apply the same listwise ranking prompt used for complex QA.

B.2 Details about Two-Stage Training Framework

Cold-Start SFT During the cold-start SFT stage, we use LlamaFactory (Zheng et al., 2024) to train two different backbone LLMs: Qwen2.5-7B-Instruct[5] and Qwen2.5-32B-Instruct[6].

As for Qwen2.5-7B-Instruct, we set the learning rate as 5e-6 and batch size per GPU as 1 with gradient accumulation steps as 8. We use DeepSpeed ZeRO-1 (Rasley et al., 2020) and FlashAttention2 (Dao, 2023) for our training. We apply mixed precision BF16 for training, set the maximum reasoning length as 3072, and train the model for 5 epochs.

As for Qwen2.5-32B-Instruct, we use LoRA (Hu et al., 2021) for efficient SFT. The lora parameters rank and alpha are both set to 32. We set learning

[5] `https://huggingface.co/Qwen/Qwen2.5-7B-Instruct`

[6] `https://huggingface.co/Qwen/Qwen2.5-32B-Instruct`

Prompt of listwise Positives Selection for Complex Queries using R1

You are an expert in evaluating the relevance of passages to a stackexchange query. A passage is considered relevant to the query if it helps reason through the query with critical concepts or theories and contains useful information to help users derive the answer.
Crucially, avoid selecting passages that:
- Merely mention keywords or entities from the query/answer but fail to provide explanatory details.
- Are generic descriptions, background facts, or tangential information that do not contribute to reasoning about the query's specific questions.

I will provide you with a query, the answer of the query and a list of passages to evaluate. Each passage is indicated by number identifier [].
Please output the identifiers of the relevant passages. The output format should be [] [], e.g., [2] [4]. If there are no relevant passages, output "None". Only output the relevant passage identifiers, do not output irrelevant passage identifiers and do not explain. Here are the Query, the Answer to the Query and a list of passages:
Query: {query} Answer: {answer}
Passage [1]: {passage$_1$} Passage [2]: {passage$_2$} (more passages)…

Output

[2] [4] [10]

Figure 6: The prompt of listwise positive passages selection for complex queries using R1 and an example output.

Prompt of Hard Negatives Selection for Complex Queries using R1

You are an expert in finding hard negative passages for a given query. Given a query, the answer to the query and a list of passages, you need to find the hard negative passages from these passages. The hard negative passage contains some relevant information with superficial lexical overlapping, but it should be not helpful to address the query with critical concepts or theories and does not contain useful information to help users derive the answer.
I will provide you with a query, the answer of the query, and a list of passages to evaluate. Each passage is indicated by a number identifier [].
Please output the identifiers of the hard negative passages. The output format should be [] [], e.g., [2] [4]. If there are no hard negative passages, output "None". Only output the hard negative passage identifiers, do not output relevant passage identifiers and do not explain. Here are the Query, the Answer to the Query, and a list of passages:
Query: {query} Answer: {answer}
Passage [1]: {passage$_1$} Passage [2]: {passage$_2$} (more passages)…

Output

[1] [3] [7]

Figure 7: The prompt of hard negative passages selection for complex queries using R1 and an example output.

Prompt for R1 Listwise Passage Ranking

User: You are an intelligent assistant that can rank passages based on their relevance to the query. I will provide you with {num} passages, each indicated by a numerical identifier []. Rank the passages based on their relevance to the query: {query}.
Assistant: Okay, please provide the passages.
User: [1] {passage$_1$}
Assistant: Received passage [1].
User: [2] {passage$_2$}
Assistant: Received passage [2].
(more passages)...
User: Search Query: {query}. Rank the {num} passages above based on their relevance to the search query. The passages should be listed in descending order using identifiers. The most relevant passages should be listed first. The output format should be [] > [], e.g., [4] > [2]. Only response the ranking results, do not say any word or explain.

Output

(Reasoning) Okay, let me try to figure out how to rank these passages for the query…
(Answer) [4] > [2] > [7] > [8] > [1] > [3] > …

Figure 8: The prompt of R1 listwise passage ranking and an example output.

```
Prompt of positives selection for coding query using R1
You are an expert in evaluating the relevance of passages to a query (coding problem).
The relevance between the query and a relevant passage is defined by whether the query either
requires the corresponding syntax documentation in the passage or involves the same algorithm
and/or data structure.
I will provide you with a query, the solution code of the query, and a list of passages to
evaluate. Each passage is indicated by a number identifier []. Please output the identifiers
of the relevant passages. The output format should be [] [], e.g., [2] [4]. If there are no
relevant passages, output "None". Only output the relevant passage identifiers, do not output
irrelevant passage identifiers and do not provide any explanation.
Here are the Query, the Solution to the Query, and a list of passages:
Query: {query} Solution: {solution}
Passage [1]: {passage_1} Passage [2]: {passage_2} (more passages)…
Output
[2] [4] [10]
```

Figure 9: The prompt of positives selection for coding query using R1 and an example output.

```
Prompt of positives selection for Math-QA using R1
You are an expert in evaluating the relevance of passages to a query (a math problem).
A query is relevant to a passage if the passage references exactly the same theorem (such as
Gauss's lemma) used in the query.
I will provide you with a query, the solution of the query, and a list of passages to evaluate.
Each passage is indicated by a number identifier [].
Please output the identifiers of the relevant passages. The output format should be [] [],
e.g., [2] [4]. If there are no relevant passages, output "None". Only output the relevant
passage identifiers, do not output irrelevant passage identifiers and do not provide any
explanation.
Here are the Query, the Solution of the Query, and a list of passages:
Query: {query} Solution: {solution}
Passage [1]: {passage_1} Passage [2]: {passage_2} (more passages)…
Output
[2] [4] [10]
```

Figure 10: The prompt of positives selection for Math-QA using R1 and an example output.

```
Prompt of positives selection for Math-Theorem using R1
You are an expert in evaluating the relevance of passages (a math theorem) to a query (a math
problem). A passage is relevant to a query if the query's solution either applies the same
theorem as presented in the passage, or follows reasoning steps substantially similar to those
used in the passage's theorem.
I will provide you with a query, the solution of the query, and a list of passages to evaluate.
Each passage is indicated by a number identifier []. Please output the identifiers of the
relevant passages. The output format should be [] [], e.g., [2] [4]. If there are no relevant
passages, output "None". Only output the relevant passage identifiers, do not output
irrelevant passage identifiers and do not provide any explanation.
Here are the Query, the Solution of the Query, and a list of passages:
Query: {query} Solution: {solution}
Passage [1]: {passage_1} Passage [2]: {passage_2} (more passages)…
Output
[2] [4] [10]
```

Figure 11: The prompt of positives selection for Math-Theorem using R1 and an example output.

rate to 1e-4 and batch size per GPU to 1 with gradient accumulation steps of 8. We use DeepSpeed ZeRO-3 (Rasley et al., 2020) with mixed precision BF16 and train the model for 4 epochs.

Multi-view Ranking based RL RL has been shown in many works (Dong et al., 2025a; Song et al., 2025; Dong et al., 2025b) to further enhance the performance of SFT models. After our cold-start SFT, we further use the GRPO reinforcement learning algorithm (Shao et al., 2024) based on the VERL framework [7] to train our models (*i.e.*, ReasonRank (7B) and ReasonRank (32B)).

The RBO reward in our multi-view ranking re-

[7]https://github.com/volcengine/verl

ward is calculated as:

$$\text{RBO} = (1-p)\sum_{d=1}^{|y^{\text{list}}|} p^{d-1}\cdot\frac{\left|y'_{1:d}\cap y^{\text{list}}_{1:d}\right|}{d}. \quad (4)$$

where y' and y^{list} represent the rollout ranking list and gold ranking list in our listwise label, and p is a pre-defined parameter.

During GRPO training, we sample a group of output sequences $G = \{y_1, y_2, \ldots, y_G\}$ for each input x. Each sequence y_i receives a reward r_i, which is then normalized within group G to produce advantages $\hat{A}_i$. The token-level optimization objective is formulated as:

$$\begin{aligned} \mathcal{J}_{\text{GRPO}}(\theta) &= \frac{1}{|G|}\sum_{i=1}^{|G|}\frac{1}{|y_i|}\sum_{t=1}^{|y_i|}\min\Big(r_{i,t}(\theta)\hat{A}_{i,t}, \\ &\qquad \text{clip}\left(r_{i,t}(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_{i,t}\Big) - \beta D_{\text{KL}}, \\ r_{i,t}(\theta) &= \frac{\pi_\theta(y_{i,t}\mid x, y_{i,<t})}{\pi_{\text{old}}(y_{i,t}\mid x, y_{i,<t})}, \\ D_{\text{KL}} &= D_{\text{KL}}(\pi_\theta \parallel \pi_{\text{ref}}), \end{aligned} \quad (5)$$

where ϵ and β are hyper-parameters.

For both ReasonRank (7B) and ReasonRank (32B), the training batch size is set as 16, and the mini-batch size is set as 8. The maximum output length is set to 3072, and rollout per sample is 8. Both models are trained for 200 steps on 8 NVIDIA A800 GPUs. As for ReasonRank (7B), we use the 7B-scale SFT model as our policy model and set the learning rate to 1e-6. As for ReasonRank (32B), we use the 32B-scale SFT model as our policy model and set the learning rate to 1e-5. The lora parameters rank and alpha are both set to 32.

We set the hyperparameters ϕ and γ used in the multi-view ranking reward as 0.2 and 0.1, respectively, by grid search with a step size of 0.1 in the range of [0.1, 0.5]. During cold-start SFT and multi-viewranking-basedd RL, we use the input prompt in Figure 12 to train ReasonRank.

C More Experiments for ReasonRank

C.1 Detailed Results of Ablation Study

We show the results of our ablation study based on ReasonRank (7B) on each BRIGHT dataset in Table 6. The ablated models show performance decline on almost all datasets, proving the effectiveness of the corresponding component.

C.2 Detailed Results on BEIR Benchmark

In this part, we present the comparison results on 7 datasets of the traditional IR benchmark BEIR (Thakur et al., 2021). We use BM25 as our retriever and rerank the top-100 retrieved passages. We also add rankT5 (3B), Rank-R1 (7B), and Rank1 (7B) baselines for a more comprehensive comparison. The results are shown in Table 7.

C.3 Different Backbones

To demonstrate the generalizability and effectiveness of our proposed ReasonRank framework, we conduct additional experiments using different backbone LLMs. In addition to Qwen2.5-7B-Instruct utilized in our main experiments, we further adopt a reasoning LLM Qwen3-8B and Meta-Llama-3.1-8B-Instruct, which represents a different model family from Qwen. The results are shown in Table 8. We also report the performance of ReasonRank, which is only trained using supervised fine-tuning (SFT), denoted as $\text{ReasonRank}_{\text{Only-SFT}}$.

As shown in Table 8, our ReasonRank consistently and significantly outperforms all baseline models across all backbones, validating the effectiveness of our synthetic data generation pipeline and the two-stage training strategy. Moreover, ReasonRank exhibits obvious performance improvements over $\text{ReasonRank}_{\text{Only-SFT}}$ on Avg. metric, which further substantiates the benefit of our multi-view ranking-based RL approach. Notably, when built upon Qwen3-8B, ReasonRank achieves the best overall average score of 36.14%, suggesting that the inherent strong reasoning capability of Qwen3 may further enhance the ranking performance of our framework.

C.4 Analysis of Reasoning Length

In this section, we explore the average length of reasoning chains using ReasonRank across various BRIGHT datasets. We select ReasonRank (7B) and rerank the top-20 passages retrieved by ReasonIR. The average reasoning length for queries in each dataset is shown in Figure 13. From the results, we observe that the reasoning length of coding tasks (*i.e.*, Leetcode and Pony) and theorem tasks (*e.g.*, AoPS, TheoremQA-Q and TheoremQA-T) tend to be longer than StackExchange tasks (*e.g.*, Robotics and Stack Overflow). This may be because coding queries and theorem queries are more complex and challenging compared to StackEx-

Models	Econ.	Earth.	Rob.	Bio.	Psy.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
ReasonRank (7B)	35.05	47.75	31.22	56.70	47.77	32.47	40.87	23.17	24.95	7.68	39.49	41.80	**35.74**
Training Data													
only MSMARCO	29.08	36.97	24.34	43.60	37.95	26.05	31.42	26.96	24.51	7.08	33.22	39.75	30.08
w/o Self Consistency	33.96	50.35	27.94	55.48	46.63	31.96	40.17	22.50	18.29	7.32	35.16	40.67	34.20
Training Approach													
w/o SFT (only RL)	27.44	35.23	23.73	50.36	34.48	25.76	29.62	18.59	23.51	6.78	34.46	34.28	28.69
w/o RL (only SFT)	30.47	44.83	27.42	50.53	43.10	30.38	38.79	25.11	22.46	7.06	37.53	40.12	33.15
w/o R^m	30.86	47.14	29.75	59.6	44.40	35.04	36.87	20.47	20.80	8.23	37.52	39.66	34.20
non-reasoning SFT	26.70	45.30	27.89	51.48	43.65	29.72	41.44	25.60	17.64	5.39	38.31	42.4	32.96

Table 6: The performance (NDCG@10) of ablated models on each BRIGHT dataset.

Input Prompt during ReasonRank Training

```
You are RankLLM, an intelligent assistant that can rank passages based on their relevance to
the query. Given a query and a passage list, you first thinks about the reasoning process in
the mind and then provides the answer (i.e., the reranked passage list). The reasoning process
and answer are enclosed within <think> </think> and <answer> </answer> tags, respectively,
i.e., <think> reasoning process here </think> <answer> answer here </answer>.
I will provide you with {num} passages, each indicated by a numerical identifier []. Rank the
passages based on their relevance to the search query: {query}.
[1] {passage_1}
[2] {passage_2}
(more passages)...
Search Query: {query}. Rank the {num} passages above based on their relevance to the search
query. All the passages should be included and listed using identifiers, in descending order
of relevance. The format of the answer should be [] > [], e.g., [2] > [1].
```

Output

```
<think> … </think> <answer> [4] > [2] > [7] > … <answer>
```

Figure 12: The input prompt of ReasonRank during training and an example output.

Models	Covid	DBPedia	SciFact	NFCorpus	Signal	Robust04	News	Avg.
BM25	59.47	31.80	67.89	33.75	33.04	40.70	39.52	43.74
Non-reasoning reranker								
RankT5 (3B)	80.19	44.85	74.60	37.40	31.73	51.45	49.11	52.76
RankZephyr (7B)	82.92	44.42	75.42	38.26	31.41	53.73	**52.80**	54.14
Reasoning reranker								
Rank-R1 (7B)	83.71	42.27	72.16	38.94	**33.08**	54.46	50.60	53.60
Rank-R1 (14B)	**84.63**	44.05	75.96	38.58	32.95	56.91	49.20	54.61
Rank1 (7B)	79.04	35.79	73.32	37.52	25.41	57.11	47.67	50.84
Rank1 (32B)	80.59	34.79	74.78	37.29	25.58	58.30	45.57	50.99
Rearank (7B)	81.28	44.99	75.02	38.83	32.69	56.16	51.04	54.29
Rank-K (32B)	74.76	38.29	60.83	36.48	32.64	49.79	45.59	48.34
ReasonRank (7B)	82.01	**46.03**	75.55	39.60	31.36	55.40	50.50	54.35
ReasonRank (32B)	83.16	45.65	**77.24**	**40.04**	31.07	**58.67**	52.24	**55.44**

Table 7: The results (NDCG@10) on 7 BEIR datasets. All models rerank BM25-retrieved top-100 passages. The best reranker is highlighted in **bold**.

change queries, thus requiring longer reasoning to understand queries and rerank the passages.

C.5 Comparison with Teacher DeepSeek-R1

During our reasoning-intensive training data synthesis, we use DeepSeek-R1 to mine positive and negative passages based on the gold answer to obtain pointwise training labels. Theoretically, this should generate better labels compared to pure teacher distillation (which does not use gold answers and is based only on the teacher), potentially allowing the trained model to outperform DeepSeek-R1. To verify this, we compare the ranking performance of our trained ReasonRank (7B, 32B) with DeepSeek-R1 on BRIGHT. Considering the high time and API cost of reranking with DeepSeek-R1, we choose to rank the top-20 (in-

Models	Econ.	Earth.	Rob.	Bio.	Psy.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
ReasonIR (8B)	32.65	43.00	20.82	43.49	39.57	30.96	27.34	31.69	19.55	7.37	33.93	36.68	30.59
rankT5 (3B)	11.35	22.11	10.94	13.62	11.40	11.35	15.96	27.45	38.05	**9.24**	18.27	9.46	16.60
RankZepyer (7B)	19.87	17.36	12.35	34.90	24.72	13.35	22.34	29.29	32.37	6.05	28.98	30.07	22.64
Rank-R1 (7B)	20.19	27.91	18.13	36.70	30.22	11.32	29.28	17.13	9.35	3.23	14.43	28.98	20.57
Rank-R1 (14B)	27.39	38.73	23.11	44.45	37.10	27.82	36.77	21.27	19.23	8.80	31.66	39.53	29.66
Rank1 (7B)	25.34	38.91	16.77	39.91	35.32	24.81	33.47	12.71	28.14	2.58	30.67	38.18	27.23
Rank1 (32B)	25.43	37.97	17.14	42.43	34.85	23.82	31.18	12.20	**40.98**	4.79	29.28	39.95	28.34
Rearank (7B)	30.65	40.27	27.16	46.94	40.77	26.12	36.26	**30.55**	22.84	7.31	32.28	39.86	31.75
Rank-K (32B)	30.09	39.78	26.60	50.64	43.45	29.87	35.17	27.23	22.80	7.61	37.06	41.00	32.61
Using Qwen2.5-7B-Instruct as the backbone													
ReasonRank$_{\text{Only-SFT}}$ (7B)	30.47	44.83	27.42	50.53	43.10	30.38	38.79	25.11	22.46	7.06	37.53	40.12	33.15
ReasonRank (7B)	**35.05**	**47.75**	31.22	**56.70**	47.77	32.47	40.87	23.17	24.95	7.68	**39.49**	41.80	35.74
Using Meta-Llama-3.1-8B-Instruct as the backbone													
ReasonRank$_{\text{Only-SFT}}$ (8B)	30.58	41.81	27.58	50.48	48.24	27.49	39.61	22.52	25.40	7.45	35.46	35.54	32.68
ReasonRank (8B)	34.47	40.60	27.46	53.44	**51.86**	29.52	43.58	22.40	23.79	7.69	34.44	39.80	34.08
Using Qwen3-8B as the backbone													
ReasonRank$_{\text{Only-SFT}}$ (8B)	34.67	44.18	**31.73**	53.31	45.55	30.58	**44.12**	24.00	25.37	8.11	37.50	43.60	35.22
ReasonRank (8B)	34.71	44.23	29.46	55.16	50.10	**32.50**	43.96	24.23	28.44	8.84	37.82	**44.19**	**36.14**

Table 8: The performance of ReasonRank (7B) based on different backbones.

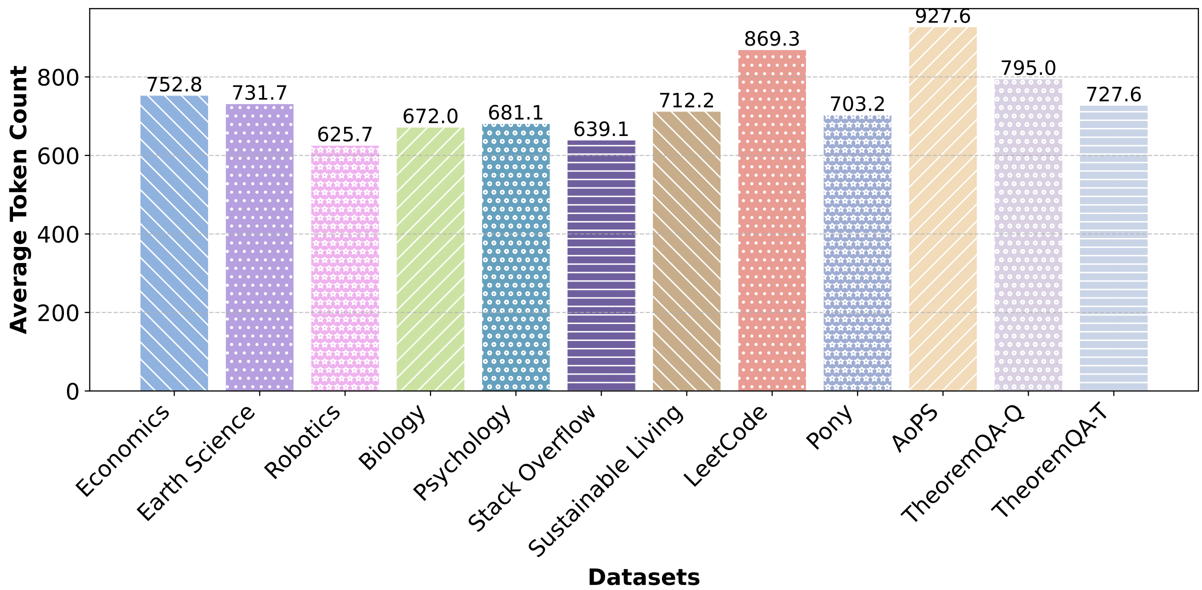

Figure 13: The reasoning length of ReasonRank (7B) on BRIGHT.

Models	Econ.	Earth.	Rob.	Bio.	Psy.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
ReasonIR (8B)	32.65	43.00	20.82	43.49	39.57	30.96	27.34	31.69	19.55	7.37	33.93	36.68	30.59
ReasonRank (7B)	31.32	43.18	27.07	50.06	45.02	33.46	37.25	24.14	**22.93**	8.23	**35.39**	40.69	33.23
ReasonRank (32B)	**33.98**	42.30	**28.66**	**51.97**	47.45	36.31	**38.51**	23.15	22.48	**8.59**	34.51	42.27	**34.18**
DeepSeek-R1 (671B)	33.75	**45.64**	26.91	50.34	**49.47**	**37.22**	36.92	**25.74**	21.08	6.81	33.30	**42.42**	34.13

Table 9: The comparison between ReasonRank and DeepSeek-R1. All models rerank ReasonIR-retrieved top-20 passages. We use GPT-4 rewritten queries during retrieval and use the original query for reranking. The best reranker is marked in bold.

stead of top-100) passages retrieved by ReasonIR. The ranking prompt used by R1 is the same as the prompt used when generating listwise labels based on R1. The results are shown in Table 9. From the results, we can see that ReasonRank (32B) achieves comparable performance to DeepSeek-R1 (34.18 vs 34.13) and surpasses DeepSeek-R1 on 7 datasets (Economics, Robotics, Biology, Sustainable Liv-

Models	Econ.	Earth.	Rob.	Bio.	Psy.	Stack.	Sus.	Leet.	Pony	AoPS	TheoQ.	TheoT.	Avg.
ReasonIR (8B)	32.65	43.00	20.82	43.49	39.57	30.96	27.34	31.69	19.55	7.37	33.93	36.68	30.59
ReasonRank (32B)	36.64	48.90	33.88	58.17	53.27	38.68	45.97	25.78	20.93	9.41	38.46	46.28	38.03
RaDeR + BM25 (Hybrid)	30.85	56.08	24.13	52.66	46.69	34.93	32.22	31.91	25.87	11.98	37.73	43.43	35.71
(1) ReasonRank (32B)	37.67	50.61	35.44	59.04	53.06	39.04	45.75	22.64	25.30	11.41	41.27	47.73	39.08
(2) ReasonRank’ (32B)	36.70	55.53	35.69	62.72	54.64	38.03	44.81	29.46	25.56	14.38	41.99	50.06	**40.80**

Table 10: Further ranking enhancement on BRIGHT. Both retrievers “ReasonIR (8B)” and “RaDeR + BM25 (Hybrid)” use GPT4-rewritten queries for retrieval.

ing, Pony, AoPS, and TheoremQA-Q), proving the superiority of training with our synthesized data compared to pure teacher distillation.

C.6 Further Ranking Enhancement

Although our ReasonRank has already achieved superior results compared to existing rerankers, we believe its ranking performance can be further improved by adjusting some ranking settings. Previous studies (Pradeep et al., 2023a,b) have revealed that the quality of retrieval results has an impact on ranking performance: better retrieval results can lead to better ranking performance to some extent. In this section, we used the retrieval results provided by RaDeR (Das et al., 2025), which combines the retrieval results of the dense retriever RaDeR and the sparse retriever BM25 and achieves higher retrieval quality than ReasonIR. We apply ReasonRank (32B) as our reranker to rerank the top-100 retrieved passages. As shown in Figure 10 (1), with better retrieval results, the ranking performance further improved to 39.08 (1 point higher than the ranking results based on ReasonIR), which aligns with the previous conclusion. Furthermore, previous work (Sun et al., 2023) pointed out that the size and stride of the sliding window also affect listwise ranking performance. In this section, different from the original settings with a window size of 20 and a stride of 10, we experimented with new sliding window parameters with a window size of 10 and a stride of 5 (denoted as ReasonRank’ (32B)). The results in Table 10 (2) show that this change brought about a 1.7-point improvement. This may be because a smaller window size can reduce the difficulty of modeling context and ranking in listwise ranking, thereby improving performance. **Remarkably, with an average score of 40.6, ReasonRank (32B) has achieved SOTA performance on the BRIGHT leaderboard on August 9, 2025.**

D Case Studies of ReasonRank (7B)

To better understand the reasoning process of our ReasonRank, we present the reasoning chains of our ReasonRank (7B) on several BRIGHT datasets, including Earth Science (Table 11), Biology (Table 12), Robotics (Table 13), LeetCode (Table 14), AoPS (Table 15), and TheoremQA-Theorems (Table 16). We use the ReasonIR-retrieved top-20 passages as the candidate passages to rerank.

E Use of AI Assistants

We use ChatGPT to improve the presentations of this paper.[8]

[8] https://chatgpt.com/

Table 11: Case on Earth Science. The model first analyzes the search query and considers the criteria that make a passage relevant. Then, it examines some relevant passages and compares them to the criteria. Finally, it quickly goes through the remaining passages and determines the final ranking.

Query

The line on the ground at the Greenwich Observatory is itself on a tectonic plate that is moving, so is the prime meridian still 0 degrees, 0 minutes, 0 seconds?
If not - what are the implications (if any)? For example, if my GPS says I am 100 km west of Greenwich Observatory, am I 100 km west of 0 0 0 or 100 km west of the line on the ground? Who (and what) decides this kind of thing?
Curious punter, not a scientist....

ReasonRank Output

<think> Okay, let me try to figure out how to rank these passages for the user's query. The main question is whether the prime meridian at Greenwich is still exactly 0° 0' 0" because the tectonic plate is moving, and what that means for GPS coordinates. The user is also curious about who decides this.
First, I need to identify which passages address the movement of the prime meridian due to tectonic shifts and how that affects GPS. The query mentions tectonic plates moving, so passages explaining that the physical line has moved are key. Also, implications like GPS readings and reference frames are important.
Passage [1] seems very relevant. It talks about the Airy Transit Circle at Greenwich and how its longitude is now 0°00'05.3" West in ITRF and WGS84. It mentions the offset of 102m and discusses astronomical vs geodetic coordinates. This directly answers if the prime meridian has moved and explains GPS references.
Passage [3] also discusses the offset and explains it through deflection of the vertical and geodetic coordinates. It addresses GPS receivers showing east of zero, which relates to the user's example. So this is probably high relevance.
Passage [8] talks about Earth's drift due to mantle movements, ice melting, etc., and mentions the pole drift. While not directly about Greenwich, it explains why tectonic/mass movements cause shifts, which ties into the prime meridian's position.
Passage [9] discusses polar motion, which is about Earth's rotational axis, but the user's query is about longitude. However, it might touch on reference frames, so could be somewhat relevant.
Passage [10] explains why Greenwich was chosen in 1884 and mentions tectonic maps using British ones. It's historical but might not address the current movement issue.
Passage [11] is titled "Why the Greenwich meridian moved" – that sounds directly relevant. It likely explains the reasons for the shift, so high priority.
Passage [12] mentions the prime meridian through Greenwich and longitude measurement. It's more about definition than movement, so maybe less relevant.
Passage [14] discusses geographical coordinates and WGS, which relates to GPS references. Explains how modern systems differ from simple models.
Passage [7] and [18] seem to be about the 1884 conference and references to the meridian movement, so they might explain the decision-making body (BIH, IERS), which answers who decides.
Passage [2] mentions BIH Terrestrial System and geodetic references, which ties into reference frames and who decides.
Passage [4] and [5] talk about Earth's equator and axis changes, but not directly about the prime meridian's movement.
Passage [16] is about geomagnetic poles, which is a different type of movement, so less relevant.
Passage [19] and [20] explain longitude but not the tectonic aspect.
So the top ones would be [1], [3], [11], [8], [7], [18], [2], [14], [10], then others. Need to check each again.
[1] is definitely top as it directly answers the movement and GPS. [3] explains the offset and geodetic systems. [11] is about why it moved, so very relevant. [8] explains Earth drift causes. [7] and [18] mention BIH and IERS, which decide references. [2] talks about reference frames. [14] about WGS. [10] historical context. Others less so.
So the order might be [1] > [3] > [11] > [8] > [7] > [18] > [2] > [14] > [10] > [9] > [4] > [5] > [16] > [12] > [17] > [20] > [13] > [6] > [19] > [15] </think>
<answer>
[1] > [3] > [11] > [8] > [7] > [18] > [2] > [14] > [10] > [9] > [4] > [5] > [16] > [12] > [17] > [20] > [13] > [6] > [19] > [15]
</answer>

Table 12: Case on Biology. The model first analyzes the search query and considers the criteria that make a passage relevant. Then, it examines some relevant passages and compares them to the criteria. Finally, it quickly goes through the remaining passages and determines the final ranking.

Query
Why does hair turn grey or white, and why does it happen later for some? The question is pretty simple: what is happening molecularly when hair turns grey or white? I would imagine that it is due to the lack of a particular compound. I'm also interested in why some people's onset of grey hair happens much later than others.
ReasonRank Output
<think> Okay, let me try to figure out how to rank these passages. The user wants to know why hair turns gray or white molecularly and why some people gray later. So I need to focus on passages that explain the biological mechanisms, like lack of a compound, melanocyte stem cells, oxidative stress, telomeres, etc. Passage [1] talks about melanocyte stem cells and Bcl2, which affects survival. That's relevant. Passage [2] discusses mouse models and factors like oxidative damage and telomerase, so that's good. Passage [4] mentions melanocyte stem cells and homeostasis, which ties into the molecular process. Passage [5] is an abstract about melanocyte stem cells and aging, so key. Passage [6] details melanocyte compartments and stem cells, explaining the process. Passage [8] covers melanocyte stem cell properties and bulge niche, which is crucial. Passage [15] includes DNA repair and telomere theories, which relate to aging and molecular causes. Passage [16] mentions melanocyte stem cell maintenance in the niche, so that's relevant. Passages [3], [7], [10], [11], [12], [18], [20] have some info but less directly on the molecular mechanisms. Others like [9], [13], [14], [17] are more about hair care or demographics, so less relevant. So the top ones are those explaining stem cell depletion, oxidative stress, telomeres, and niche environment. Then others that support those points. The care-related ones are less important here. </think> <answer> [4] > [1] > [5] > [6] > [8] > [2] > [15] > [16] > [10] > [3] > [7] > [12] > [20] > [18] > [9] > [11] > [13] > [14] > [17] > [19] </answer>

Table 13: Case study on Robotics. The model successfully rank the passage with relatively short reasoning length.

Query
ROS2: Adding parameters to YAML launch file How do I use a YAML launch file to pass parameters to a node? The tutorial covers passing parameters using a python launch file but not YAML.
ReasonRank Output
<think> Okay, let me try to figure out how to rank these passages for the user's query about using YAML launch files to pass parameters in ROS2. The user mentioned that the tutorial covers Python launch files but not YAML, so they're specifically looking for YAML examples. First, I'll go through each passage to see which ones directly address YAML launch files and parameters. Passage [2] has a YAML config example in the launch file, mentioning `turtlesim_custom_params.yaml`, which seems very relevant. Passage [3] talks about the `<rosparam>` tag used in launch files with YAML, which is key here. Passage [8] also explains the `<rosparam>` tag in the context of YAML files. Passages [13]–[18] all mention `rosparam` and using YAML with `roslaunch`, which is related. Passage [19] discusses debugging with YAML files. Passage [6] gives an example of YAML parameters for Nav2, which is a good example. Passage [5] mentions YAML config files for parameters. Passage [1] also talks about YAML config files for parameters. Passages [10] and [11] are about parameter defaults and loading from YAML. The others, like [4], [7], [12], and [20], are either about Python or C++ or not directly answering the YAML question. So the most relevant are [2], [3], [8], then [13]-[18], [6], [5], [19], [1], followed by others. Need to make sure the order prioritizes direct YAML examples and explanations of the '<rosparam>' tag. </think> <answer> [2] > [3] > [8] > [13] > [14] > [15] > [16] > [17] > [18] > [6] > [5] > [19] > [1] > [10] > [11] > [9] > [4] > [7] > [12] > [20] </answer>

Table 14: Case on LeetCode. The model rethinks its initial ranking and repeatedly verifies and compares the top-ranked passages..

Query
Given an ‘m x n‘ integers ‘matrix‘, return the length of the longest increasing path in ‘matrix‘. From each cell, you can either move in four directions: left, right, up, or down. You **may not** move **diagonally** or move **outside the boundary** (i.e., wrap-around is not allowed).
ReasonRank Output
<think> Okay, let me try to figure out how to rank these passages based on the query about the longest increasing path in a matrix. The user is looking for the length of the longest path. First, I need to go through each passage and see which ones are relevant. The query is specifically about dynamic programming approaches for this problem, possibly using memoization since it’s a grid traversal with constraints. Passage [3] talks about the longest increasing subsequence using dynamic programming. That’s related but not exactly the same as the matrix problem. It’s about sequences, not paths in a grid. So maybe not the top. Passage [6] is the length of the longest increasing subsequence again. Similar to [3], so not directly relevant. Passage [13] discusses maximal square area, which uses DP but not for paths. Not relevant. Passage [16] is about the longest palindromic subsequence. Again, subsequence, not grid paths. Not relevant. Passage [19] deals with unique paths with obstacles using DP. It’s about grid paths but not increasing paths. Might be somewhat related but not directly. Passage [11] is shortest clear path in a binary matrix using BFS. Different problem but grid paths. Maybe somewhat related but not the same. Passage [15] involves collecting gold in a grid with DFS. It’s about grid traversal but not increasing paths. Not directly relevant. Passage [20] is about longest increasing subsequence with difference constraints. Subsequence again, not grid. Not relevant. Passage [4] counts the number of longest increasing subsequences. Still subsequence, not grid paths. Passage [9] is longest common subsequence. Not relevant. Passage [10] is LCS again. Not applicable. Passage [14] involves min operations for robots on a 2xN grid. Not related to increasing paths. Passage [5] is about Excel cells, not relevant. Passage [18] is about work hours and palindromic intervals. Not related. Passage [2] is about strictly increasing paths in a grid, returning the count. Passage [1] discusses max points in queries starting from top-left, using DFS. It involves grid traversal and paths but not the specific increasing path problem. Might be somewhat related but not directly. Passage [17] is about LCS length again. Not relevant. So the most relevant is passage [2], which directly addresses strictly increasing paths in a grid. Then passage [19] and [8] deal with grid paths but not increasing. Passage [3] and [6] are about subsequences, not paths. Passage [15] is about grid traversal but not increasing. Wait, passage [2] has code that seems to count strictly increasing paths, which is exactly the problem. The example in [2] matches the problem’s examples. So [2] is top. Then maybe [19] and [8] for grid DP, but they’re not directly solving the same problem. Passage [20] is about subsequences with difference constraints, not paths. So the ranking would have [2] first, then perhaps [19], [8], [7], etc., but the key is [2] is the most relevant. Then maybe [3] and [6] since they handle increasing sequences, but not grid paths. So the order should start with [2], then others that involve grid paths or DP for paths. But the query is specifically about the longest increasing path in a matrix. So [2] is the most relevant. Then maybe [19] and [8] for grid DP, but they’re not the same. So the top passages are [2], followed by [19], [8], [7], [15], [1], etc. But need to check each again. Looking at passage [20], it’s about subsequences, not paths. Passage [13] is about squares, not paths. Passage [16] is LCS. So the most relevant is [2], then maybe [19] and [8] for grid DP. But the problem in the query is a graph problem. Passage [2] might use a similar approach. The code in [2] seems to count paths, which could be part of the solution. So the ranking would be: [2] > [19] > [8] > [7] > [15] > [1] > [3] > [6] > [20] > [4] > [9] > [10] > [13] > [16] > [17] > [11] > [14] > [5] > [18] > [12] But need to verify. The problem in the query is a classic DP with memoization on the grid. Passage [2] might be about counting all paths, which is a different problem but uses similar techniques. However, the code in [2] might not directly solve the longest path. Alternatively, passage [19] is about unique paths with obstacles, which uses DP but not for increasing paths. Passage [8] is min path sum, which uses DP but again not for increasing. So the most relevant is [2], then maybe [19], [8], [7], [15], [1], etc. But I’m not sure. The query’s problem is a graph traversal with DP, so passages that deal with grid paths and DP would be more relevant. Passage [2] is about strictly increasing paths, which is exactly the problem. So [2] is definitely the top. Then perhaps [19] and [8] for grid DP. So the final ranking would start with [2], then others that involve grid paths or DP for paths. The rest are less relevant. </think> <answer> [2] > [19] > [8] > [7] > [15] > [1] > [3] > [6] > [20] > [4] > [9] > [10] > [13] > [16] > [17] > [11] > [14] > [5] > [18] > [12] </answer>

Table 15: Case on AoPS. We can see that the model first analyzes the search query. Then, it looks at each passage and judges their relevance. Finally, it takes a closer look at the most relevant ones and makes a detailed comparison.

Query
Triangle ABC is inscribed in circle ω. Points P and Q are on side $\overline{AB}$ with $AP < AQ$. Rays CP and CQ meet ω again at S and T (other than C), respectively. If $AP = 4, PQ = 3, QB = 6, BT = 5$, and $AS = 7$, then $ST = \frac{m}{n}$, where m and n are relatively prime positive integers. Find $m + n$.
ReasonRank Output
<think> Okay, let me try to figure out how to rank these passages for the given problem. The query is about a triangle inscribed in a circle with some points and intersections. The problem involves calculating ST given various lengths. First, I need to understand what the problem is asking. It's a geometry problem involving a circle, triangle ABC, points P and Q on AB, rays CP and CQ intersecting the circle again at S and T, and given specific segment lengths like AP=4, PQ=3, QB=6, BT=5, AS=7. The goal is to find ST as a fraction m/n and then m+n. Now, I need to rank the 20 passages based on relevance. The most relevant ones would be those that deal with similar geometry problems, especially involving circles, triangles, intersecting lines, power of a point, chords, or Ceva's theorem, Menelaus, etc. Looking at the passages: [1] Talks about a circle, chord lengths, power of a point. Might be relevant since the problem involves chords and circle properties. [2] Involves triangle inscribed in a circle, tangents, and intersections. Could be relevant as it's about triangle geometry with circle. [3] Circles intersecting, common tangents, trapezoid area. Less relevant, seems more about trapezoids and tangents. [4] Circumcenter and distance calculation. Maybe somewhat related but not directly. [5] Incircle trisecting median. Not directly related. [6] Tangent from origin to circle. Maybe not. [7] Parabola and circle, tangent length. Not relevant. [8] Parallelogram and circle tangent. Not relevant. [9] Circle tangent to other circle. Not relevant. [10] Perimeter with Law of Cosines. Maybe trigonometry, but not directly. [11] Circumference calculation. Not relevant. [12] Circumcircle and tangents. Could be relevant as it involves circumcircle and intersections. [13] Point inside triangle with congruent angles. Not directly related. [14] Concentric circles and tangents. Not relevant. [15] Circle tangent to square. Not relevant. [16] Hexagon inscribed in circle. Not relevant. [17] Tangent and angle bisector. Maybe involves some circle theorems. [18] Similar to [14], concentric circles. Not relevant. [19] Right triangle legs. Basic Pythagorean, not relevant. [20] Circle tangent to parabola. Not relevant. So the most relevant ones are probably [1], [2], [12], [17], maybe [4], [20]. Passage [1] discusses chord lengths and power of a point, which might be useful here since the problem involves ST as a chord. Passage [2] is about triangle inscribed in a circle with tangents and intersections, which is very similar to the query's setup. Passage [12] involves circumcircle and tangent points, which could relate to the problem's intersections at S and T. Passage [17] deals with tangents and angle bisectors, which might involve some theorems used in the problem. Passage [4] uses circumradius and power of a point, which could be applicable. Passage [20] involves circle tangents and equations, but maybe less so. So the ranking would start with [2], then [1], [12], [17], [4], followed by others. Need to check which directly address circle geometry with intersecting chords, power of a point, or Ceva's/Menelaus. </think> <answer> [2] > [1] > [12] > [17] > [4] > [20] > [8] > [14] > [18] > [5] > [3] > [6] > [7] > [10] > [13] > [9] > [15] > [16] > [19] > [11] </answer>

Table 16: Case study on TheoremQA-Theorems. The model first attempts to solve the problem, then checks the candidate passages.

Query

You are working with four possible machines, each of which is represented by a matrix that operates on an input vector v: (a) $(\Phi^T\Phi)^{-1}$. (b) $\Phi(\Phi^T\Phi)^{-1}$. (c) $\Phi(\Phi^T\Phi)^{-1}\Phi^T$. (d) $\Phi^T(\Phi^T\Phi)^{-1}\Phi^T$. Which of them takes any vector v and projects it onto the space spanned by the columns of Φ?

ReasonRank Output

<think> Okay, let me try to figure out how to rank these passages for the query. The user is asking which of the four matrices projects any vector $\mathbf{v}$ onto the space spanned by the columns of Φ. The options are:

(a) $(\Phi^T\Phi)^{-1}$

(b) $\Phi(\Phi^T\Phi)^{-1}$

(c) $\Phi(\Phi^T\Phi)^{-1}\Phi^T$

(d) $\Phi^T(\Phi^T\Phi)^{-1}\Phi^T$

First, I need to recall what an orthogonal projection matrix looks like. From linear algebra, the projection matrix onto the column space of Φ is usually given by $\Phi(\Phi^T\Phi)^{-1}\Phi^T$. So options (c) and (d) might be candidates. But let me check each one.
Option (a) is just the inverse of $\Phi^T\Phi$, which isn't a projection matrix. Projection matrices must be idempotent ($P^2 = P$) and symmetric. (a) is likely not idempotent.
Option (b) is Φ multiplied by that inverse. Multiplying Φ by that matrix might not yield a projection. Let me think: $\Phi(\Phi^T\Phi)^{-1}$ is the Moore-Penrose pseudoinverse of Φ, but the projection is Φ times that times Φ^T. So (c) is $\Phi\Phi^\dagger\Phi^T$, which is the projection.
Option (d) is Φ^T pseudoinverse times Φ^T. Wait, no—that's not right. The correct form is Φ pseudoinverse times Φ^T. So (c) is correct. Option (d), written as $\Phi^T(\Phi^T\Phi)^{-1}\Phi^T$, is not the same. Therefore, the correct answer is (c).
Now, looking at the passages, which ones discuss orthogonal projections onto the column space? Passage [2] talks about orthogonal projection onto the closed linear span using an orthonormal basis, which is exactly what's needed here. Passage [3] mentions orthogonal projections as linear transformations, which is relevant. Passage [8] discusses properties of projections, like P composed with P is P. Passage [17] defines projections in Hilbert spaces, which includes the kernel condition. Passage [20] talks about bounded projections, which might be related but less directly. Passage [6] explains linear transformations as matrix products, which is relevant for understanding how these matrices act. Passage [9] involves inner products and matrix factors, which could relate to the projection formula. Passage [14] is about matrix products as linear transformations, which is key here.
So the most relevant passages would be those that explain orthogonal projections, especially using the formula $\Phi(\Phi^T\Phi)^{-1}\Phi^T$. Passage [2] gives the formula for orthogonal projection onto a subspace. Passage [3] confirms it's a linear transformation. Passage [8] discusses idempotence, a property of projections. Passage [17] defines projections. Passage [6] connects matrices to linear transformations.
Others, like [9] and [14], might offer supporting information but are less directly relevant.
So the ranking would start with [2], then [3], [8], [17], [6], followed by others like [20], [9], [14], etc. Need to check which passages explicitly mention the projection formula or idempotence. **</think>**

<answer>
[2] > [3] > [8] > [17] > [6] > [20] > [9] > [14] > [12] > [1] > [7] > [10] > [18] > [13] > [15] > [4] > [5] > [11] > [16] > [19]
</answer>