🦙🦙 CoRanking: Collaborative Ranking with Small and Large Ranking Agents

Wenhan Liu[1], Xinyu Ma[2], Yutao Zhu[1], Lixin Su[2], Shuaiqiang Wang[2]
Dawei Yin[2] and Zhicheng Dou[1*]
[1]Gaoling School of Artificial Intelligence, Renmin University of China
[2]Baidu Inc., Beijing, China
lwh@ruc.edu.cn, xinyuma2016@gmail.com, dou@ruc.edu.cn

Abstract

Listwise ranking based on Large Language Models (LLMs) has achieved state-of-the-art performance in Information Retrieval (IR). However, their effectiveness often depends on LLMs with massive parameter scales and computationally expensive sliding window processing, leading to substantial efficiency bottlenecks. In this paper, we propose a Collaborative Ranking framework (**CoRanking**) for LLM-based listwise ranking. Specifically, we strategically combine an efficient *small* reranker and an effective *large* reranker for collaborative ranking. The *small* reranker performs initial passage ranking, effectively filtering the passage set to a condensed top-k list (e.g., top-20 passages), and the *large* reranker (with stronger ranking capability) then reranks only this condensed subset rather than the full list, significantly improving efficiency. We further address that directly passing the top-ranked passages from the small reranker to the large reranker is suboptimal because of the LLM's strong positional bias in processing input sequences. To resolve this issue, we propose a passage order adjuster learned by RL that dynamically reorders the top passages returned by the small reranker to better align with the large LLM's input preferences. Our extensive experiments across three IR benchmarks demonstrate that CoRanking achieves superior efficiency, reducing ranking latency by approximately 70% while simultaneously improving effectiveness, compared to the standalone large reranker.

1 Introduction

In recent years, many studies (Liu et al., 2024c; Sun et al., 2023; Zhu et al., 2023) have demonstrated strong zero-shot passage ranking capabilities of large language models (LLMs). A typical approach is listwise ranking, which feeds the query and a list of passages into an LLM and instructs it to output a ranked list of passage IDs in descending order of relevance. This approach has established state-of-the-art performance (Sun et al., 2023) on major IR benchmarks, including TREC (Craswell et al., 2020b) and BEIR (Thakur et al., 2021).

Despite its effectiveness, LLM listwise passage ranking faces critical efficiency challenges, as illustrated in Figure 1(A), primarily stemming from two factors: First, its state-of-the-art performance relies on LLMs with massive parameters (such as Qwen2.5-72B or GPT-4), which introduces significant computational overhead and results in an efficiency issue. Second, due to the limitation of the context length of LLMs, existing listwise ranking methods typically employ a sliding window strategy (Sun et al., 2023). With a fixed window size (e.g., 20 passages) and step size (e.g., 10 passages), this strategy bubbles the relevant passages that are initially lowly ranked to the top. However, the interdependence between windows hinders the parallelization of sliding windows, which results in many repetitive and sequential rankings of sliding windows (*e.g.*, totally 9 sliding windows if window size and step size are set as 20 and 10 when ranking the top 20 from 100 passages) and creates an efficiency bottleneck (especially when employing LLMs with massive parameters).

The sliding window strategy of LLM reranker primarily serves to elevate initially low-ranked relevant passages into the final reranking window (*e.g.*, top 20). We believe that this repetitive and inefficient process could be replaced by a small listwise reranker.[1] By utilizing a small listwise reranker to pre-rank the whole list of passages (e.g., 100 passages) using a sliding window strategy, the large listwise reranker, which has stronger ranking capability, only needs to rerank the passages in the final window (*i.e.*, containing the top 20 passages).

---

*Corresponding author.

[1]In this work, we consider models with parameters less than or equal to 3B as small rerankers.
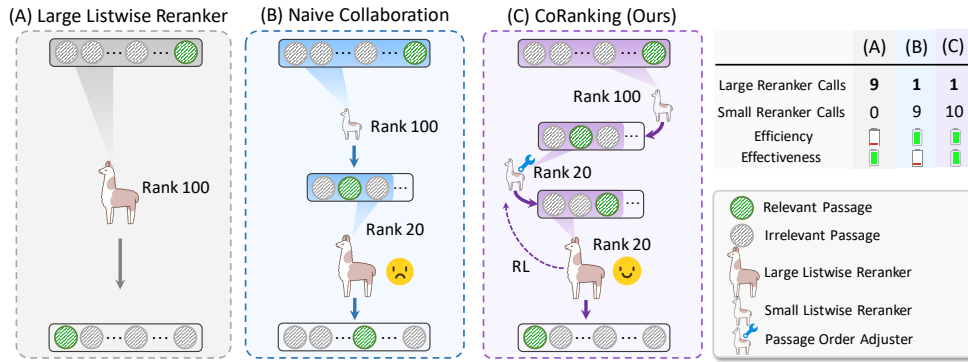
Figure 1: (A) Large Listwise Reranker: Apply a large listwise reranker to rerank 20 passages from 100 candidates based on a sliding window strategy. (B) Naive Collaboration: Pre-rank 100 passages with a small listwise reranker using a sliding window strategy, then rerank the top 20 with a large listwise reranker. (C) CoRanking: Pre-rank 100 passages with a small reranker using a sliding window strategy, adjust the order of the top 20 via the passage order adjuster, then rerank them with the large listwise reranker. "Large/Small Reranker Calls" represents the number of sliding windows for listwise ranking.

This hybrid approach could dramatically reduce computational costs and improve efficiency.

However, existing small listwise rerankers are mainly trained on golden lists whose passages are ranked solely based on relevance (Pradeep et al., 2023b,a). We argue that directly feeding the top-ranked passages returned by these small listwise rerankers into a large listwise reranker may lead to sub-optimal performance (see Figure 1 (B)). This is because the effectiveness of a listwise reranker has a complex relationship with the order of input passages, rather than being determined exclusively by their relevance, as indicated by existing studies (Sun et al., 2023; Tang et al., 2024). For example, Tang et al. (2024) reveal that relevant passages appearing in the middle of the list tend to be ranked at the bottom by the listwise reranker. This stems from LLMs' pre-training biases toward the positions of input tokens (Liu et al., 2024a). While the passage order generated by the small listwise reranker, which is trained using relevance-based ranking, may deviate from the large listwise reranker's order preference. To address this issue, we introduce a passage order adjuster (also a small listwise reranker) to rerank the top-ranked passages from the small listwise reranker so that they align with the order preference of the large listwise reranker (see Figure 1 (C)).

More specifically, we propose a collaborative ranking framework, namely **CoRanking**, that integrates small and large ranking models to perform passage ranking in a multi-stage manner. In the first stage, we train a small listwise reranker (SLR) to pre-rank all the passages to elevate the relevant

passages to the top part of the list (e.g., top-20). Different from existing methods (Pradeep et al., 2023a,b) that solely use teacher-generated ranking lists as training labels, we propose a human-label-enhanced method to improve the ranking quality. In the second stage, we train a passage order adjuster (POA) using the reinforcement learning algorithm DPO. The POA reorders the top-ranked passages to align with the large listwise reranker's order preference. In addition, we design a significance-aware selection strategy S^3 to build high-quality preference pairs for DPO training. In the third stage, a large listwise reranker is used to further rerank the top-ranked passages, enhancing the overall ranking effectiveness. Extensive experiments on diverse IR benchmarks demonstrate that our CoRanking framework achieves a 70% reduction in ranking latency while maintaining superior effectiveness, compared to a standalone large listwise reranker.

Our contributions are summarized as follows:

(1) We propose a ranking framework, CoRanking, that enables small and large LLMs to collaborate, thereby achieving efficient and effective passage ranking.

(2) We propose a novel passage order adjuster (POA) that aligns the output passages of the small listwise reranker with the order preference of the large listwise reranker. We further design the S^3 strategy, which builds valuable preference pairs for POA's training.

(3) We conduct extensive experiments on diverse IR benchmarks, and the results demonstrate that our CoRanking achieves a significant efficiency improvement compared to a standalone large listwise

reranker while exhibiting superior performance.

2 Methodology

In this section, we present a collaborative ranking framework CoRanking that integrates both small and large passage rerankers. As illustrated in Figure 2, the framework comprises three key components: (1) a small listwise reranker, (2) a passage order adjuster, and (3) a large listwise reranker. Next, we will introduce the definition of the ranking task, the inference process of CoRanking, and the training details.

2.1 Task Definition

The aim of passage ranking is 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 q and a set of passages as input, and generates a reranked sequence of passage IDs (*e.g.*, [4] [2]...). Due to the constraints of LLMs' input length, existing listwise ranking approaches typically employ a sliding window strategy that iteratively processes passage subsets. Specifically, this strategy applies a window of size w that moves from the end to the beginning of the passage list with a step size s, dynamically promoting relevant passages to higher positions. Following previous studies (Sun et al., 2023), we set the total number of passages N to 100, with window size w and step size s set to 20 and 10, respectively.

2.2 Inference Process of CoRanking

Given the passages to rerank, our CoRanking (see Figure 2) first applies the small listwise reranker to pre-rank all the passages using a sliding window strategy. Then, we use the passage order adjuster to rerank the top-20 passages in a listwise manner, so that their order meets the preference of the large listwise reranker. Finally, the large listwise reranker is applied to further rerank the top-20 passages to obtain the final ranking list. This avoids the large listwise reranker's inefficient sliding window operations on all 100 passages, significantly improving ranking efficiency. The specific prompt template for a large listwise reranker is provided in Appendix A, which is consistent with previous studies (Pradeep et al., 2023b; Liu et al., 2024c). In the following, we will illustrate the training details of our small listwise reranker and passage order adjuster.

2.3 Small Listwise Reranker (SLR)

To mitigate the latency and computational overhead of the large listwise reranker (denoted as LLR), we propose using a small listwise reranker (denoted as SLR) to pre-rank the entire passage list. The SLR aims to elevate relevant passages into the final sliding window (i.e., the top-20 passages), thereby reducing the number of LLR's sliding window iterations. Here we adopt listwise training because we find that it performs better than other approaches (*e.g.*, pointwise training (Nogueira and Cho, 2019)), and the SFT-trained SLR naturally serves as the initialization for our listwise passage order adjuster, which will be detailed in Section 2.4.

Training Label   Conventional listwise reranker training typically employs knowledge distillation (Pradeep et al., 2023a,b), which uses a strong teacher model (*e.g.*, GPT-4) to rerank BM25-retrieved top-20 passages, with the teacher-generated rankings serving as supervised fine-tuning (SFT) labels. However, the teacher model can not perform consistently well across all queries, such as long-tail queries, which can result in sub-optimal performance. To further improve the SFT label, we propose a *human-label-enhanced ranking construction* (HRC) method that improves teacher-generated rankings with human labels.

Specifically, we first iteratively rerank BM25-retrieved top-20 passages M times using a powerful teacher model in a listwise manner, where the $(i + 1)$-th reranking input is the output ranking from the i-th iteration. In this paper, we apply LLR as our teacher model and set the maximum iteration number to 5. This progressive reranking approach is motivated by prior studies (Pradeep et al., 2023b,a) which reveal that iteratively reranking enables gradual improvement of ranking lists. Then, we choose the ranking where human-annotated relevant passages are ranked at the top (*i.e.*, NDCG=1) during the iteration process as the gold ranking.

Training Loss   After obtaining the gold ranking, we concatenate the query q and the candidate passages $[p_1, \ldots, p_N]$ together as the training input x and optimize our SLR by minimizing the standard language modeling loss $\mathcal{L}$:

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

$$x = q \circ [1] \circ p_1 \circ [2] \circ p_2 \ldots [N] \circ p_N, \qquad (2)$$

where $[i]$ represents the ID of passage p_i and y is the gold ranking label (*e.g.*, "[4] [2] ...").
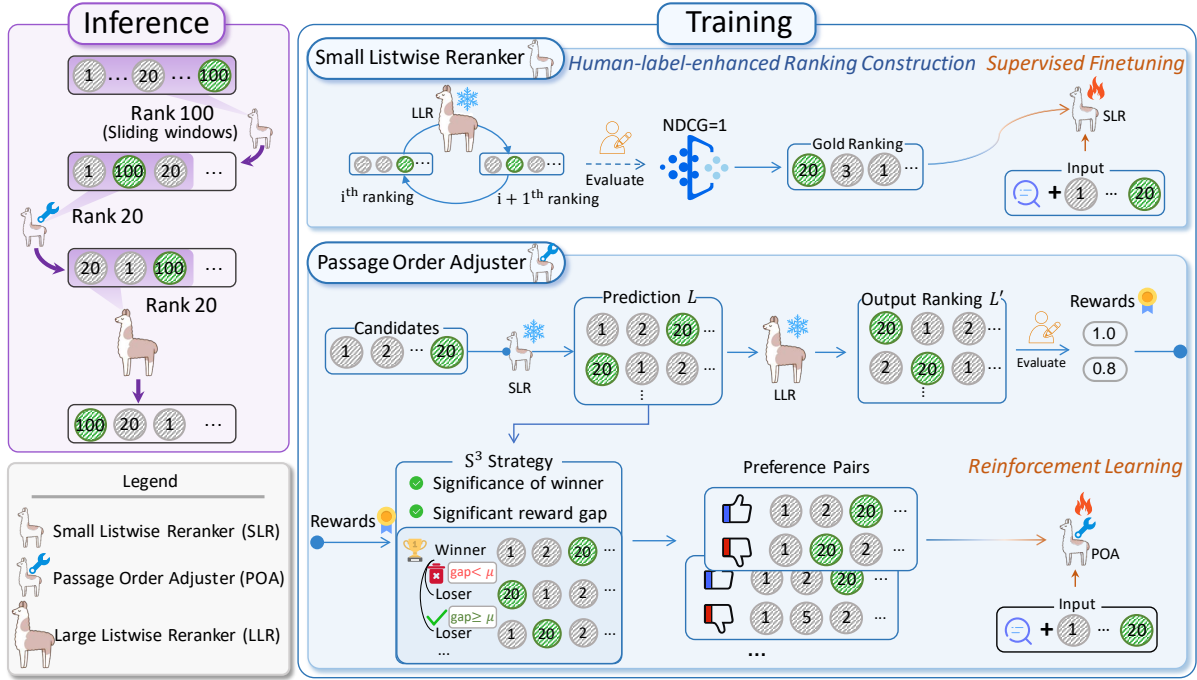
Figure 2: The "Inference" part shows how our CoRanking works with both large and small listwise rerankers. As for training of the small listwise reranker, we design a human-label-enhanced ranking construction method to generate gold rankings for supervised fine-tuning. As for training the passage order adjuster, we proposed the S^3 strategy to obtain high-quality preference pairs for DPO reinforcement learning.

2.4 Passage Order Adjuster (POA)

While our SLR is trained to rank relevant passages at the top of the list, the order of top-ranked passages (*i.e.*, the top-20 passages) may not align well with the order preference of LLR. To address this issue, we employ a reinforcement learning (RL) algorithm to train a passage order adjuster POA based on LLR's feedback to rerank these top-ranked passages. We adopt the Direct Preference Optimization (DPO) (Rafailov et al., 2023) algorithm for its optimization stability. The following sections elaborate on the construction of DPO training data and model optimization.

Training Data   The training data of DPO consists of a series of prediction pairs. Each prediction pair contains a winner and a loser prediction, namely, a passage ID list. The reward R for each prediction L is computed by feeding L to LLR and evaluating the ranking metric M of *LLR's output ranking L'* based on human-annotated relevance label y:

$$R = M(L', y), \tag{3}$$

where we use NDCG@10 as our metric M.

We apply our trained SLR to rerank BM25-retrieved top-100 passages with a sliding window strategy for training queries and use the top-20 reranked passages as training candidate passages, which aims to align with the input passage distribution encountered by POA during inference. To generate the preference pairs for training POA, we first sample m predictions from SLR by taking the candidates as the input, then evaluate these predictions through Equation 3. After that, the highest-reward list is selected as the winner prediction, while others serve as loser predictions. Next, we pair the winner prediction with all the loser predictions to construct initial training preference pairs.

To ensure the quality of these preference pairs, we introduce a *significance-aware selection strategy* S^3 to filter out low-quality pairs. Specifically, the S^3 strategy follows two rules: (1) *Significance of Winner*: The reward of the winner prediction must be 1 and exceed LLR's ranking metric (*i.e.*, the NDCG@10 of directly use LLR to rerank the candidate passages), ensuring the winner prediction provides positive improvement for final reranking of LLR; (2) *Significant Gap between winner and loser*: The reward difference between winner prediction and loser prediction must exceed a threshold μ, eliminating noisy pairs with marginal differences.

Optimization   Given constructed preference

5101

pairs, we optimize POA using the following DPO objective:

$$\mathcal{L}_{\text{DPO}} = -\mathop{E}_{x,y_w,y_l} \left[\log \sigma \left(\beta \log \frac{\pi_\theta^w \pi_f^t}{\pi_f^w \pi_\theta^t} \right) \right], \quad (4)$$

$$\pi_\theta^w = \pi_\theta(y_w \mid x), \quad \pi_f^t = \pi_f(y_l \mid x), \quad (5)$$

$$\pi_f^w = \pi_f(y_w \mid x), \quad \pi_\theta^t = \pi_\theta(y_l \mid x), \quad (6)$$

where x represents the training input whose format is the same as Equation (2). y_w, y_l represent the winner prediction and loser prediction, respectively. π_θ is the policy model to be optimized, and π_f is the original policy model, which is initialized with our trained SLR. The frozen π_f acts as a regularizer to maintain generation stability during optimization, and β balances preference alignment and regularization towards π_f.

3 Experiments

3.1 Setting

Evaluation Datasets We evaluate our framework on three established information retrieval benchmarks: TREC DL (Craswell et al., 2020b), BEIR (Thakur et al., 2021), and BRIGHT (Su et al., 2024). For the TREC DL benchmark, we adopt the TREC Deep Learning 2019 (DL19) (Craswell et al., 2020b) and 2020 (DL20) (Craswell et al., 2020a) datasets derived from MS MARCO v1. The BEIR benchmark (Thakur et al., 2021) is selected to evaluate cross-domain generalization capabilities. We choose three representative datasets: TREC-Covid, Robust04, and Trec-News. BRIGHT (Su et al., 2024) is a reasoning-intensive IR benchmark whose relevance assessment transcends simplistic keyword matching or semantic similarity, instead demanding intentional and deliberate reasoning. We choose Economics, Earth Science, and Robotics datasets for evaluation. The detailed descriptions of each dataset will be provided in Appendix B.

Implementation Details As for the supervised fine-tuning of our SLR, we use Qwen2.5-3B-Instruct as the backbone model. The training queries consist of a total of 2k queries sampled from the MS MARCO training set. We utilize Qwen2.5-72B-Instruct[2] as our teacher model to SFT our SLR and also employ it as the LLR model throughout this study unless otherwise specified. As for the RL training of POA, we use our trained

SLR for model initialization. We set the threshold μ as 0.4 and generate about 12k preference pairs from 2k queries for the DPO algorithm. Please see the Appendix D for more implementation details.

Baselines We compare our framework against a series of baselines, which can be divided into two parts: (1) Single reranker, (2) Collaborative rerankers (i.e. reranking with small and large rerankers).

As for single reranker, we compare our trained SLR with a series of state-of-the-art supervised rerankers, including **monoBERT (340M)** (Nogueira and Cho, 2019), **monoT5 (3B)** (Nogueira et al., 2020), **RankT5 (3B)** (Zhuang et al., 2023b), **RankVicuna (7B)** (Pradeep et al., 2023a), **RankZephyr (7B)** (Pradeep et al., 2023b) and **RankMistral**$_{100}$ (Liu et al., 2024c). The first three are pointwise rerankers trained from human labels in the MS MARCO dataset (Nguyen et al., 2016). The last three are listwise rerankers which are distilled from GPT-3.5, GPT-4, and GPT-4o, respectively. To ensure more fair comparison with SLR in model architecture and training data, we introduce baseline **Qwen-pointwise**, which is a pointwise reranker based on Qwen2.5-3B-Instruct. This model concatenates a query-passage pair as input and outputs a relevance score. We utilize the same gold ranking label as SLR and optimize the model with RankNet loss, which is the same as what Sun et al. (2023) did. We also report the ranking performance of our **SLR** and **LLR**, which rerank all the passages using the sliding window strategy.

As for collaborative rerankers, we introduce three naive collaborative baselines, including **monoT5 (3B) + LLR**, **RankT5 (3B) + LLR**, and **Qwen-pointwise (3B) + LLR**. The three baselines first use a corresponding small reranker (3B) to rerank all the passages and then apply LLR to rerank the top-20 passages.

3.2 Overall Results

The main experimental results are presented in Table 1, from which we could draw the following observations:

(1) Our collaborative reranking framework CoRanking achieves the best average performance (51.39) across all datasets, outperforming both single rerankers and other collaborative baselines. Notably, it even surpasses the large listwise reranker

---

Models	Strategy	TREC DL		BEIR			BRIGHT			Avg.
		DL19	DL20	Covid	Robust04	News	Econ.	Earth.	Rob.	
BM25	-	50.58	47.96	59.47	40.70	39.52	16.45	27.91	10.91	36.69
Single Reranker										
monoBERT (340M)	Pointwise	70.46	67.55	70.85	46.11	47.61	10.45	19.98	11.34	43.04
monoT5 (3B)	Pointwise	71.57	69.83	72.39	49.81	49.34	9.65	19.16	8.56	43.79
RankT5 (3B)	Pointwise	71.83	70.27	80.19	51.45	49.11	8.66	24.75	13.95	46.28
Qwen-pointwise (3B)	Pointwise	70.62	64.99	81.77	53.60	47.78	14.13	26.25	9.89	46.13
RankVicuna (7B)	Sliding	67.72	65.98	79.19	48.33	47.15	11.05	16.30	8.03	42.97
RankZepyer (7B)	Sliding	**73.39**	70.02	82.92	53.73	**52.80**	15.63	18.52	12.12	47.39
RankMistral$_{100}$ (7B)	Full	72.55	**71.29**	82.24	**57.91**	50.59	18.82	23.91	15.75	49.13
SLR (3B)	Sliding	70.24	68.85	81.85	53.68	49.85	15.56	**29.47**	13.68	47.90
LLR (72B)	Sliding	73.26	70.51	**85.46**	57.76	51.28	**21.34**	28.88	**19.04**	**50.94**
Collaborative Rerankers										
monoT5 (3B) + LLR	Collaborative	70.79	70.73	81.80	55.08	51.60	13.46	21.83	11.68	47.12
RankT5 (3B) + LLR	Collaborative	69.16	**71.41**	84.66	55.80	50.04	12.92	25.91	16.01	48.24
Qwen-pointwise (3B) + LLR	Collaborative	72.28	67.89	81.01	56.37	**52.30**	20.00	28.12	18.26	49.53
CoRanking	Collaborative	**72.79**	70.48	**84.90**	57.71	52.26	**22.72**	**31.20**	19.03	**51.39**

Table 1: Results (NDCG@10) on TREC, BEIR, and BRIGHT benchmarks. The "Pointwise", "Sliding", 'Full' and "Collaborative" in "Strategy" column represent pointwise ranking, sliding window strategy, full ranking (Liu et al., 2024c) and collaborative ranking, respectively. The best and second-best results for the "Single Reranker" part and the "Collaborative Rerankers" part are indicated in bold and underlined, respectively. "Avg." represents the averaged result of all 8 datasets.

LLR by about 0.5 points on average. These results demonstrate that our framework could effectively align the passage order generated by the SLR (3B) with the preferences of the large language model (LLR), thereby achieving better ranking performance than only using LLR. (2) Regarding single rerankers, our SLR (3B) achieves an average score of 47.90, outperforming all the pointwise rerankers, including monoBERT, monoT5, RankT5, and Qwen-pointwise, and even surpassing 7B-scale RankVicuna and RankZephyr. Furthermore, SLR surpasses Qwen-pointwise by about 1.8 average points under identical training conditions, proving listwise reranking's superiority through comparison of multiple passages for precise relevance assessment.

(3) Collaborative reranker baselines (*e.g.*, RankT5 (3B) + LLR) improve over corresponding single rerankers (*e.g.*, RankT5 (3B)). However, these baselines still lag behind LLR and our CoRanking, indicating that directly feeding small rerankers' ranking results to LLR fails to align with LLR's order preference.

3.3 Efficiency Analysis

As mentioned in Section 1, our primary target of CoRanking is to mitigate the efficiency issue of pure LLM-based reranking. To quantitatively demonstrate this advantage, we compare our
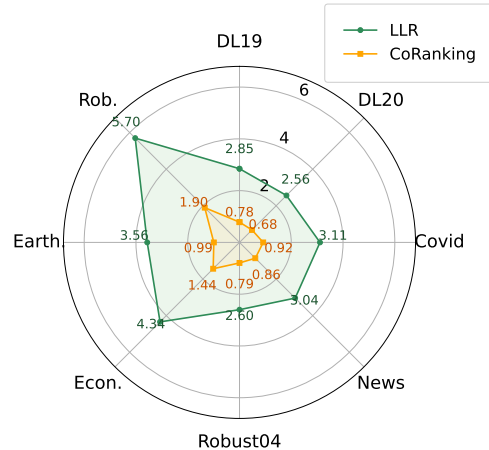

Figure 3: Ranking latency (seconds/query) of LLR and CoRanking on 8 datasets.

CoRanking against LLR across all 8 datasets. Our experiments are conducted on 8×40GB NVIDIA A100 GPUs using the vLLM inference framework[3]. We average across all queries within each dataset and show the comparison results in Figure 3.

Across all datasets, our CoRanking approach demonstrates a significant reduction in latency, achieving a 67%-74% decrease compared to the LLR. This substantial improvement highlights the efficiency advantage of CoRanking. Additionally, the latency of less than 1 second achieved

---

[3] https://github.com/vllm-project/vllm

Model	TREC Avg.	BEIR Avg.	BRIGHT Avg.
CoRanking	**71.64**	**64.96**	**24.32**
w/o S^3 strategy	70.10	63.86	23.26
w/o POA	70.23	63.80	22.74
SLR	**69.55**	**61.79**	**19.57**
w/o HRC	68.22	59.58	19.43

Table 2: Average performance (NDCG@10) of ablated models on TREC (DL19, DL20), BEIR (Covid, Robust04, News), and BRIGHT (Economics, Earth Science, Robotics).

by CoRanking across most datasets also showcases its practical efficiency in real-world search engines. Lastly, the ranking latency varies significantly across different datasets, which is primarily due to the differences in passage lengths. This is because longer passages inherently require more computational time for processing.

3.4 Ablation Study

To evaluate the contribution of each component in our framework, we conduct comprehensive ablation experiments. Table 2 presents the average performance for each individual benchmark, with detailed per-dataset results available in Table 4.

First, to evaluate our proposed S^3 sampling strategy for generating the preference pairs of DPO, we introduce a variant, denoted as "w/o S^3". This variant uses random sampling for preference pair selection, without ensuring the significant reward of winner prediction and the reward gap between the winner and loser predictions. The performance degradation across all benchmarks (*e.g.*, 1.54-point drop on TREC Avg.) demonstrates the effectiveness of our S^3 sampling in identifying valuable preference pairs for effective DPO training.

We further remove our POA to validate its alignment effectiveness, denoted as "w/o POA". In this case, the LLR directly reranks the top-20 passages after SLR's reranking (*i.e.*, the naive collaboration mentioned in Figure 1 (B)). The notable performance decline (especially 1.6-point decrease on BRIGHT Avg.) demonstrates that POA could effectively align with the order preference of LLR.

Finally, to validate our human-label-enhanced ranking construction (HRC) for fine-tuning SLR, we create a variant (w/o HRC) which generates ranking labels by utilizing LLR to rerank training passages only once and choosing not to use human labels for filtering high-quality ranked lists. The significant performance drops across benchmarks (especially BEIR) highlight HRC's effectiveness in
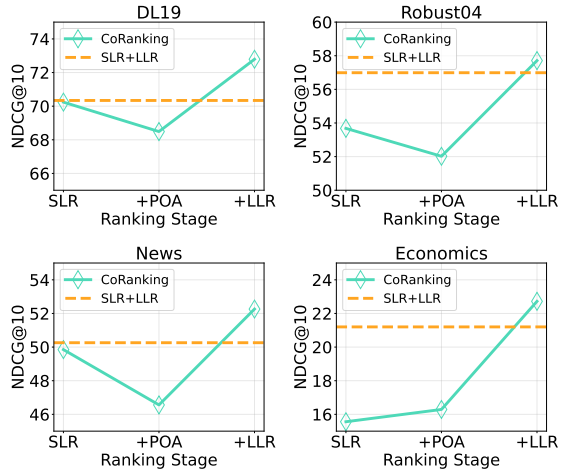

Figure 4: The changes of NDCG@10 after reranking by SLR, POA, and LLR.

generating high-quality ranking labels.

3.5 Detailed Analysis of POA

Our POA is designed to adjust the order of the top-20 passages reranked by SLR to align them more with the preferences of the LLR. However, whether this adjustment improves or degrades the ranking quality of top-20 passages remains unknown. We believe this is an interesting question because it helps us understand the reasons why POA works.

In this part, we selected four datasets—DL19, Robust04, News, and Economics—and reported the changes in the ranking metric NDCG@10 after ranking by SLR, POA, and LLR, respectively. We also report the performance of the naive collaboration SLR+LLR as a baseline. The results are shown in Figure 4. After POA's adjustment, the ranking metric on three datasets—DL19, Robust04, and News—decreases (except for the Economics dataset). This indicates that a passage ranking list with a high-ranking metric does not necessarily align with LLR's preferences. This may be due to the positional bias of input tokens during the pre-training of LLMs (Tang et al., 2024). Moreover, due to the stronger ranking capability, LLR further improves effectiveness compared to SLR and SLR+POA, demonstrating its necessity in the CoRanking framework.

3.6 Generalization on Other LLMs

In the previous experiments, we use Qwen2.5-72B-Instruct as our LLR model. However, it remains unclear whether our proposed CoRanking framework can generalize effectively to other LLR models. To evaluate this, we conduct additional experiments

Model	DL19	News	Earth.	Avg.
BM25	50.58	39.52	27.91	39.34
Using GPT-4o as LLR				
RankT5 (3B) + LLR	73.17	49.31	29.64	50.71
Qwen-pointwise (3B) + LLR	72.35	48.36	34.62	51.78
SLR+LLR	73.10	48.75	34.97	52.27
CoRanking	**74.10**	**50.19**	**35.93**	**53.41**
Using DeepSeek-V3 as LLR				
RankT5 (3B) + LLR	71.90	49.47	31.47	50.95
Qwen-pointwise (3B) + LLR	72.16	49.07	35.38	52.20
SLR+LLR	72.62	50.91	38.09	53.87
CoRanking	**73.79**	**51.48**	**39.01**	**54.76**

Table 3: The performance (NDCG@10) of CoRanking based on different LLRs. "Avg." represents the average performance across the three datasets.

with two different LLRs: GPT-4o[4] and DeepSeek-V3, and compare our CoRanking with three collaborative rerankers ("RankT5 (3B) + LLR", "Qwen-pointwise (3B) + LLR", and "SLR + LLR"). Note that "SLR + LLR" represents the naive collaboration in Figure 1 (B). For evaluation, we select one dataset from each of the three benchmarks: DL19 from TREC, News from BEIR, and Earth Science from BRIGHT. As illustrated in Table 3, our CoRanking consistently outperforms the best baseline "SLR + LLR" by about 1 point on average, regardless of the LLR used. This demonstrates that the order preference alignment of our POA could generalize to different LLMs.

4 Related Work

4.1 LLMs for Passage Ranking

The application of large language models (LLMs) into retrieval systems (Zhu et al., 2023; Li et al., 2025a) has driven substantial research efforts in passage ranking tasks. Current LLM-based ranking methods can be categorized into three categories: pointwise, pairwise, and listwise. Pointwise methods (Liang et al., 2022; Sachan et al., 2022; Zhuang et al., 2023a; Liu et al., 2024e,d; Li et al., 2025b) compute the relevance score for each query-passage pair. Pairwise methods (Qin et al., 2023; Luo et al., 2024; Li et al., 2025d) compare two passages at a time and identify the more relevant one. Listwise methods (Sun et al., 2023; Pradeep et al., 2023a; Reddy et al., 2024; Liu et al., 2024c,b; Chen et al., 2024) directly take a list of passages as the input and output the passage identifiers in descending

order of their relevance. Through comparing multiple passages, a listwise reranker could assess the relevance more comprehensively and yield better ranking results. Due to the superior performance of listwise ranking, it has attracted the interest of many researchers (Tang et al., 2024; Pradeep et al., 2023b; Liu et al., 2025; Fan et al., 2025; Li et al., 2025c).

The superior performance of LLM listwise rerankers is often attributed to the utilization of large-scale LLMs and the sliding window strategy, which leads to an efficiency issue in ranking. In this paper, we introduce a novel framework that leverages the collaborative ranking of large rerankers and small rerankers to enhance the ranking efficiency while maintaining ranking effectiveness.

4.2 Preference Alignment

Preference alignment has emerged as a critical research direction, aiming to align model behaviors with human or task-specific preferences (Sun et al., 2024). Early approaches primarily relied on reinforcement learning from human feedback (RLHF), where models are fine-tuned using reward signals derived from human preference annotations. To reduce the computational complexity of RLHF, recent studies propose more efficient alternatives, such as DPO (Rafailov et al., 2023) and Step-DPO (Lai et al., 2024). Direct preference optimization (DPO) (Rafailov et al., 2023) reformulates preference learning as a supervised objective, bypassing explicit reward modeling while achieving competitive performance. In this paper, to address the positional sensitivity of a large listwise reranker, we propose training a passage order adjuster with the DPO algorithm to align the passage order generated by the small listwise reranker with the order preference of the large listwise reranker.

5 Conclusion

In this paper, we introduce a novel collaborative ranking framework, namely CoRanking, to combine small and large listwise rerankers for more efficient and effective passage ranking. Specifically, we first train a small listwise reranker SLR to move the relevant passages to the top of the list. Then we introduce a passage order adjuster POA to rerank the top-ranked passages to better align with the order preference of a large listwise reranker. Finally, the large listwise reranker is utilized to solely rerank the top passages. Extensive experiments

---

[4]The version of GPT-4o we used is gpt-4o-2024-08-06.

demonstrate the efficiency advantage and superior performance of our CoRanking.

Limitations

We acknowledge some limitations in our work. First, our framework consists of three models, making its online deployment more complex compared to using a single unified reranker. Second, we only use Qwen2.5-3B-Instruct as the backbone model of our SLR and POA. We believe it is meaningful to attempt using different backbones (such as Llama-series models) and models of varying sizes (such as Qwen2.5-1.5B-Instruct) to further demonstrate the generalization of our CoRanking framework.

Acknowledgments

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.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, and Daniel Campos. 2020a. Overview of the TREC 2020 deep learning track. In *TREC*, volume 1266 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Ellen M. Voorhees. 2020b. Overview of the TREC 2019 deep learning track. *CoRR*, abs/2003.07820.

Yongqi Fan, Kui Xue, Zelin Li, Xiaofan Zhang, and Tong Ruan. 2025. 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.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *CoRR*, abs/2406.18629.

Yuchen Li, Hengyi Cai, Rui Kong, Xinran Chen, Jiamin Chen, Jun Yang, Haojie Zhang, Jiayi Li, Jiayi

Wu, Yiqun Chen, et al. 2025a. Towards ai search paradigm. *arXiv preprint arXiv:2506.17188*.

Yuchen Li, Haoyi Xiong, Yongqi Zhang, Jiang Bian, Tianhao Peng, Xuhong Li, Shuaiqiang Wang, Linghe Kong, and Dawei Yin. 2025b. Rankelectra: Semi-supervised pre-training of learning-to-rank electra for web-scale search. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, pages 2415–2425.

Yuchen Li, Hao Zhang, Yongqi Zhang, Hengyi Cai, Mingxin Cai, Shuaiqiang Wang, Haoyi Xiong, Dawei Yin, and Lei Chen. 2025c. Rankexpert: A mixture of textual-and-behavioral experts for multi-objective learning-to-rank in web search. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 4437–4449.

Yuchen Li, Hao Zhang, Yongqi Zhang, Xinyu Ma, Wenwen Ye, Naifei Song, Shuaiqiang Wang, Haoyi Xiong, Dawei Yin, and Lei Chen. 2025d. M2oerank: Multi-objective mixture-of-experts enhanced ranking for satisfaction-oriented web search. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 4441–4454. IEEE Computer Society.

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.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. Lost in the middle: How language models use long contexts. *Trans. Assoc. Comput. Linguistics*, 12:157–173.

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

Wenhan Liu, Xinyu Ma, Weiwei Sun, Yutao Zhu, Yuchen Li, Dawei Yin, and Zhicheng Dou. 2025. Reasonrank: Empowering passage ranking with strong reasoning ability. *arXiv preprint arXiv:2508.07050*.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024c. 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. 2024d. How to personalize and whether to personalize? candidate documents decide. *Knowl. Inf. Syst.*, 66(9):5581–5604.

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

Jian Luo, Xuanang Chen, Ben He, and Le Sun. 2024. Prp-graph: Pairwise ranking prompting to llms with graph aggregation for effective text re-ranking. In *ACL (1)*, pages 5766–5776. Association for Computational Linguistics.

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.

Rodrigo Frassetto Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with BERT. *CoRR*, abs/1901.04085.

Rodrigo Frassetto Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 708–718. Association for Computational Linguistics.

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.

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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Revanth Gangi Reddy, JaeHyeok Doo, Yifei Xu, Md. Arafat Sultan, Deevya Swain, Avirup Sil, and Heng Ji. 2024. FIRST: faster improved listwise reranking with single token decoding. *CoRR*, abs/2406.15657.

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.

Ian Soboroff, Shudong Huang, and Donna Harman. 2019. Trec 2019 news track overview. In *TREC*.

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. 2024. BRIGHT: A realistic and challenging benchmark for reasoning-intensive retrieval. *CoRR*, abs/2407.12883.

Weiwei Sun, Zhengliang Shi, Wu Jiu Long, Lingyong Yan, Xinyu Ma, Yiding Liu, Min Cao, Dawei Yin, and Zhaochun Ren. 2024. MAIR: A massive benchmark for evaluating instructed retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14044–14067, Miami, Florida, USA. Association for Computational Linguistics.

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.

Raphael Tang, Xinyu Crystina Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Ture. 2024. Found in the middle: Permutation self-consistency improves listwise ranking in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 2327–2340. 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*.

Ellen M. Voorhees. 2004. Overview of the TREC 2004 robust track. In *Proceedings of the Thirteenth Text REtrieval Conference, TREC 2004, Gaithersburg, Maryland, USA, November 16-19, 2004*, volume 500-261 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Ellen M. Voorhees, Tasmeer Alam, Steven Bedrick, Dina Demner-Fushman, William R. Hersh, Kyle Lo, Kirk Roberts, Ian Soboroff, and Lucy Lu Wang. 2020. TREC-COVID: constructing a pandemic information retrieval test collection. *SIGIR Forum*, 54(1):1:1–1:12.

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.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2023a. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. *CoRR*, abs/2310.14122.

Honglei Zhuang, Zhen Qin, Rolf Jagerman, Kai Hui, Ji Ma, Jing Lu, Jianmo Ni, Xuanhui Wang, and Michael Bendersky. 2023b. 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.

A Listwise Ranking Prompt

The listwise ranking prompt of LLR used in this paper is shown as bellow:

Prompt: Listwise Ranking Prompt

You are RankLLM, an intelligent assistant that can rank passages based on their relevancy 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 search query: {query}.
[1] {passage 1}
[2] {passage 2}
...
[{num}] {passage {num}}

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 output format should be [] > [], e.g., [4] > [2]. Only respond with the ranking results, do not say any word or explain.

B Datasets Details

In this part, we provide the details of each evaluation dataset we used:

• **DL19/DL20** The TREC Deep Learning 2019 (DL19) (Craswell et al., 2020b) and 2020 (DL20) (Craswell et al., 2020a) datasets are derived from the MS MARCO V1 passage ranking corpus. DL19 contains 43 test queries with dense relevance annotations from the TREC 2019 Deep Learning Track, while DL20 includes 54 queries from the 2020 edition. Both datasets focus on web search scenarios and are widely adopted as standard benchmarks for large-scale information retrieval systems.

• **TREC-Covid** (Voorhees et al., 2020): A biomedical dataset comprising scientific articles related to COVID-19, with 50 test queries from the TREC 2020 COVID Track.

• **Robust04** (Voorhees, 2004): A news retrieval dataset from the TREC Robust Track 2004, featuring 249 queries with ambiguous or complex information needs.

• **TREC-News** (Soboroff et al., 2019): A web-archive retrieval dataset containing news articles from the Washington Post corpus, evaluated with 57 test queries requiring event-oriented semantic matching.

• **Economics**: Focuses on technical questions in economics, including policy analysis and theoretical discussions from StackExchange. Contains 103 complex queries derived from user posts.

• **Earth Science**: Targets geophysical and climate-related inquiries, such as weather patterns and environmental processes. Includes 116 queries based on detailed observational questions.

• **Robotics**: Centers on robotics engineering challenges, including system errors and design optimizations. Comprises 101 queries from StackExchange's technical discussions.

C Baselines Details

In this paper, we include two kinds of baselines: (1) Single reranker and (2) Collaborative rerankers. The details of single reranker are shown as below:

• **monoBERT (340M)** (Nogueira and Cho, 2019): A BERT-large-based cross-encoder re-ranker.

• **monoT5 (3B)** (Nogueira et al., 2020): A T5-based sequence-to-sequence re-ranker that computes relevance scores via pointwise ranking loss.

• **RankT5 (3B)** (Zhuang et al., 2023b): A T5-based re-ranker optimized with listwise ranking loss.

• **Qwen-pointwise (3B)**: To ensure fair comparison in model architecture and training data, we fine-tune a pointwise re-ranker based on Qwen2.5-3B-Instruct. This model concatenates queries and passages as input to predict relevance scores, trained on SLR data with RankNet loss following (Sun et al., 2023).

• **RankVicuna** (Pradeep et al., 2023a): RankVicuna is a listwise re-ranker distilled from GPT-3.5-generated ranked lists.

• **RankZephyr** (Pradeep et al., 2023b): RankZephyr is a listwise reranker distilled from GPT-3.5 and GPT-4.

• **RankMistral$_{100}$** (Liu et al., 2024c): **RankMistral$_{100}$** is a full reranker trained using a full ranking list generated by GPT-4o. It directly takes all the passages as input and output the reranked list without relying on a sliding window strategy.

• **LLR (72B)**: The large listwise reranker that employs a zero-shot sliding window strategy for passage re-ranking.

Model	TREC		BEIR			BRIGHT			
	DL19	DL20	Covid	Robust04	News	Econ.	Earth.	Rob.	Avg.
CoRanking	**72.79**	**70.48**	**84.90**	**57.71**	**52.26**	**22.72**	**31.20**	**19.03**	**51.39**
w/o S^3 strategy	70.90	69.30	83.18	56.91	51.48	21.88	30.38	17.51	50.19
w/o POA	70.34	70.11	84.14	56.99	50.26	21.20	30.50	16.51	50.01
SLR	**70.24**	**68.85**	**81.85**	**53.68**	**49.85**	15.56	29.47	**13.68**	**47.90**
w/o HRC	69.53	66.90	79.36	50.61	48.77	**15.59**	**29.66**	13.04	46.68

Table 4: Results (NDCG@10) of ablated models on TREC, BEIR, and BRIGHT benchmarks. The best-performing models are marked in bold.

D   More Implementation Details

Unlike previous studies (Pradeep et al., 2023b; Liu et al., 2024c) that adopted GPT-series models (*e.g.*, GPT-4o) as the teacher model, we select Qwen2.5-72B-Instruct as it demonstrates comparable and even better performance to GPT-4o across most datasets while being more cost-effective. Nevertheless, we have included additional experimental results using GPT-4o as the LLR model in Section 3.6. We train the SLR for 4 epochs with a learning rate of 5e-6 and a batch size of 1. The batch size, learning rate, and the hyperparameter β of DPO are set as 1, 1e-6, and 0.4, respectively.