SumRank: Aligning Summarization Models for Long-Document Listwise Reranking

Anonymous ACL submission

Abstract

Large Language Models (LLMs) have demonstrated superior performance in listwise passage reranking task. However, directly applying them to rank long-form documents introduces both effectiveness and efficiency issues due to the substantially increased context length. To address this challenge, we propose a pointwise summarization model SumRank, aligned with downstream listwise reranking, to compress long-form documents into concise rank-aligned summaries before the final listwise reranking stage. To obtain our summarization model SumRank, we introduce a three-stage training pipeline comprising cold-start Supervised Fine-Tuning (SFT), specialized RL data construction, and rank-driven alignment via Reinforcement Learning. This paradigm aligns the SumRank with downstream ranking objectives to preserve relevance signals. We conduct extensive experiments on five benchmark datasets from the TREC Deep Learning tracks (TREC DL 19–23). Results show that our lightweight SumRank model achieves state-of-the-art (SOTA) ranking performance while significantly improving efficiency by reducing both summarization overhead and reranking complexity.

1 Introduction

Document ranking plays a central role in Information Retrieval (IR) by reranking initially retrieved documents to improve the quality of search results. Recently, Large Language Models (LLMs) have demonstrated strong capabilities in zero-shot passage ranking (Zhu et al., 2023). Among various LLM-based ranking paradigms, listwise ranking has emerged as particularly effective (Sun et al., 2023; Liu et al., 2025c). Instead of evaluating documents independently, listwise approaches consider an entire candidate list jointly, enabling the model to capture global relevance patterns across documents. Compared with pointwise (Zhuang et al., 2023) and pairwise ranking (Qin et al., 2024), listwise ranking leads to more accurate ranking decisions and has achieved state-of-the-art (SOTA) performance on many IR benchmarks (Craswell et al., 2020b,a; Thakur et al., 2021; Su et al., 2025).

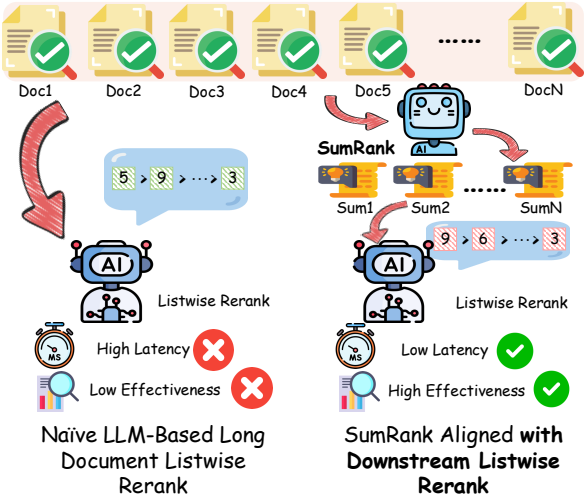

Figure 1: The left part shows the method of naive LLM-based long document listwise rerank. The right part shows our proposed SumRank, which is better than the naive listwise method both in effectiveness and efficiency. $\text{Sum}_1, \text{Sum}_2, \ldots, \text{Sum}_N$ denote the summaries generated by SumRank.

Despite this success, these studies (Zhuang et al., 2024; Sun et al., 2023; Zhuang et al., 2023; Liu et al., 2025e,c) primarily focus on reranking *short-passage style documents*, where each candidate unit contains only a limited amount of text (*e.g.*, around 100 words), while reranking *long-form documents* has received little attention. In fact, long-document ranking is of greater practical importance in many real-world retrieval scenarios (Liu et al., 2025a; Li et al., 2025a). In modern search engines, retrieved results are typically full-length webpages (with thousands of words) rather than short passages, and relevant information may be distributed across different sections of a lengthy document. Ac-

curately assessing the relevance of such long documents is therefore essential not only for traditional web search, but also for downstream applications that rely on long-document retrieval, such as deep search (Li et al., 2025b; Zhang et al., 2025; Liu et al., 2026a), where long web pages are retrieved as supporting evidence.

However, directly applying listwise reranking to long documents introduces both effectiveness and efficiency issues. This is mainly because the dramatically increased input length not only significantly increases the context modeling burden of LLM rerankers, which degrades the ranking performance (Liu et al., 2025e), but also leads to prohibitive inference latency (Yue et al., 2025). Although prior work has explored rule-based strategies for processing long documents, such as truncation (Dai and Callan, 2019) and chunking (Ma et al., 2023; Gao et al., 2023), these approaches remain fundamentally limited. Truncation discards potentially important evidence appearing later in the document, while chunking breaks global semantic coherence across passages, both of which weaken the ranking model's ability to capture complete relevance signals.

To address these limitations, we propose **SumRank**, a novel summarization model oriented toward long-document listwise reranking. Rather than just compressing text, this model is optimized to generate summaries that satisfy the strict requirements of the downstream ranking task. Specifically, we first deploy SumRank to compress each long document into a compact, query-aware summary that preserves essential relevance signals, and then perform listwise reranking over these summaries. SumRank is trained through a three-stage pipeline. We first initialize the model via cold-start supervised fine-tuning (SFT) using summaries distilled from a strong teacher model to learn basic document compression capabilities. We further align the summarizer with downstream ranking objectives via rank-driven reinforcement learning, where the reward is directly derived from the listwise ranking metric (NDCG@10) computed by an LLM-based reranker.

We evaluate the effectiveness of our SumRank across five TREC Deep Learning datasets (TREC DL 19-23). Experimental results demonstrate that our method achieves state-of-the-art (SOTA) ranking accuracy. By effectively compressing lengthy documents into summaries tailored for downstream ranking tasks, it consistently outperforms simple truncation methods, traditional summarizers, and zero-shot LLMs.

The main contributions of this work are: (1) We propose using a pointwise summarization model before reranking to solve the challenges of long-document listwise reranking. (2) We introduce a three-stage training framework, which includes a cold-start SFT strategy for basic summary generation, a specialized data construction strategy for the following RL, and a rank-driven alignment via RL strategy to tailor the generated summaries specifically for the downstream ranking task. (3) We conduct extensive experiments to demonstrate the effectiveness and efficiency advantages of our SumRank.

2 Related Work

2.1 Document Reranking with LLM

With the advancement of LLMs, document reranking methodologies have evolved into three primary paradigms: pointwise, pairwise, and listwise. Pointwise methods (Nogueira et al., 2020; Liu et al., 2026b) judge the relevance of a single query-document pair independently. Pairwise methods (Qin et al., 2024) judge the relative relevance by comparing a pair of documents. Listwise approaches (Sun et al., 2023; Ma et al., 2023; Pradeep et al., 2023a,b; Liu et al., 2025d,e,c,b) directly take a document list as the input, which captures the global document relevance, and output the reranked document IDs. While these methods are effective, they primarily focus on optimizing short passages reranking. Directly applying these methods to long-document tasks often leads to performance drops due to input length limitations and lack of generalizability.

2.2 Long Document Modeling

Due to the quadratic computational complexity of self-attention (Vaswani et al., 2017), standard Transformers are inefficient for long-document processing. To resolve this issue, a line of research focuses on optimizing the architecture by replacing the global fully connected attention with sparse attention mechanisms (Beltagy et al., 2020; Zaheer et al., 2020; Jiang et al., 2020; Zhou et al., 2022). Other approaches tackle length constraints via fragmentation. Early methods relied on heuristic pooling (FirstP/MaxP/SumP) (Dai and Callan, 2019) or optimized training objectives based on cumulative gain (Wu et al., 2020) to aggregate pas-

sage scores. More recent works focus on efficiency and representation depth (Hofstätter et al., 2021; Li et al., 2024). However, these methods are primarily designed to accommodate long inputs, rather than optimizing for the downstream ranking objective. To bridge this gap, we propose SumRank, which directly compresses the long document into summaries tailored specifically for listwise ranking.

3 Methodology

In this section, we propose a comprehensive three-stage training framework to obtain a summarization model, **SumRank** strictly aligned with downstream listwise reranking tasks. As illustrated in Figure 2, this section is organized as follows: we first formulate the problem of reranking based on summarization (§ 3.1). We then detail the three stages of our framework: Cold-Start SFT (§ 3.2), RL Data Construction(§ 3.3) and Rank-Driven Alignment via GRPO (§ 3.4).

3.1 Problem Formulation

Given a query q and a candidate document list $\mathcal{D} = \{d_1, ..., d_n\}$, the ultimate goal of document ranking is to determine a relevance permutation π (*i.e.*, [5] > [1] > . . .) that maximizes the relevance to q.

To reduce computational cost and avoid the hallucination phenomenon of LLMs on long texts, we adopt the Summarize-then-Rank paradigm. Specifically, this paradigm introduces a pointwise generative summarizer π_θ and a listwise ranker R_ϕ. Each document d_i is first compressed into a query-grounded summary: $s_i = \pi_\theta(d_i, q)$. Subsequently, the ranker R_ϕ evaluates the condensed summary set $\mathcal{S} = \{s_1, ..., s_n\}$ to output the final listwise ranking. Following prior work (Sun et al., 2023), we employ a sliding window strategy (window size $w = 20$, step size $sz = 10$) to aggregate local rankings into the final global permutation π.

3.2 Cold-Start SFT

To equip our model with preliminary summarization capabilities adapted to the downstream long-document listwise reranking task, we choose to distill the summarization generation ability of a strong teacher model into a lightweight student model parameterized by θ. While powerful LLMs excel at generating high-quality summaries, deploying them directly incurs severe computational overhead (Liu et al., 2024). Therefore, we frame this initial stage as a pointwise knowledge distillation task (Hinton et al., 2015; Hsieh et al., 2023).

Specifically, for each given query q and long document D, we first use an instruction template $\mathcal{T}$ (detailed in appendix A) to prompt the teacher LLM to generate a golden summary y^*. We then construct a standardized input prompt $X = \mathcal{T}(q, D)$ using the template $\mathcal{T}$. During the Supervised Fine-Tuning (SFT) phase, the student model is trained to replicate the teacher's output by minimizing the standard autoregressive cross-entropy loss:

$$\mathcal{L}_{\text{SFT}}(\theta) = -\frac{1}{T}\sum_{t=1}^{T} \log \pi_\theta(y_t^* | X, y_{<t}^*), \quad (1)$$

where $y_{<t}^*$ denotes the sequence of prefix tokens generated prior to time step t, and π_θ represents the predicted probability distribution of the student model. By optimizing this token-level loss, our model efficiently internalizes the basic document compression capabilities, providing a solid initialization for the subsequent reinforcement learning stage.

3.3 RL Data Construction

To construct the environment for our reinforcement learning stage, we construct a standard training dataset $\mathcal{D}$ containing queries with corresponding positive and negative documents. To facilitate listwise evaluation, we construct a candidate list $\mathcal{L}_D$ of size N.

To guarantee a valid listwise learning signal, the evaluation list must simultaneously contain both positive and negative documents. We first employ an initial retriever (*e.g.*, BM25) to obtain a top-N candidate list $\mathcal{R}_q$. We then evaluate the composition of $\mathcal{R}_q$: if it already includes both positive and negative instances, we utilize it directly. Otherwise, if $\mathcal{R}_q$ lacks either positive document or negative document, we explicitly inject k missing documents $\mathcal{D}_k^*$ (sampled from the ground-truth positive or negative set, respectively) by replacing the lowest-ranked candidates in $\mathcal{R}_q$:

$$\mathcal{L}_D = \text{Shuf}(\mathcal{R}_q[1 : N - k] \cup \mathcal{D}_k^*). \quad (2)$$

Finally, we utilize the cold-started student model π_{SFT} to decode an initial summary for every document in $\mathcal{L}_D$, yielding a static background summary list $\mathcal{L}_S$:

$$\mathcal{L}_S = \left\{ s_i \mid s_i \sim \pi_{\text{SFT}}(\cdot|X_i), \forall D_i \in \mathcal{L}_D \right\}. \quad (3)$$

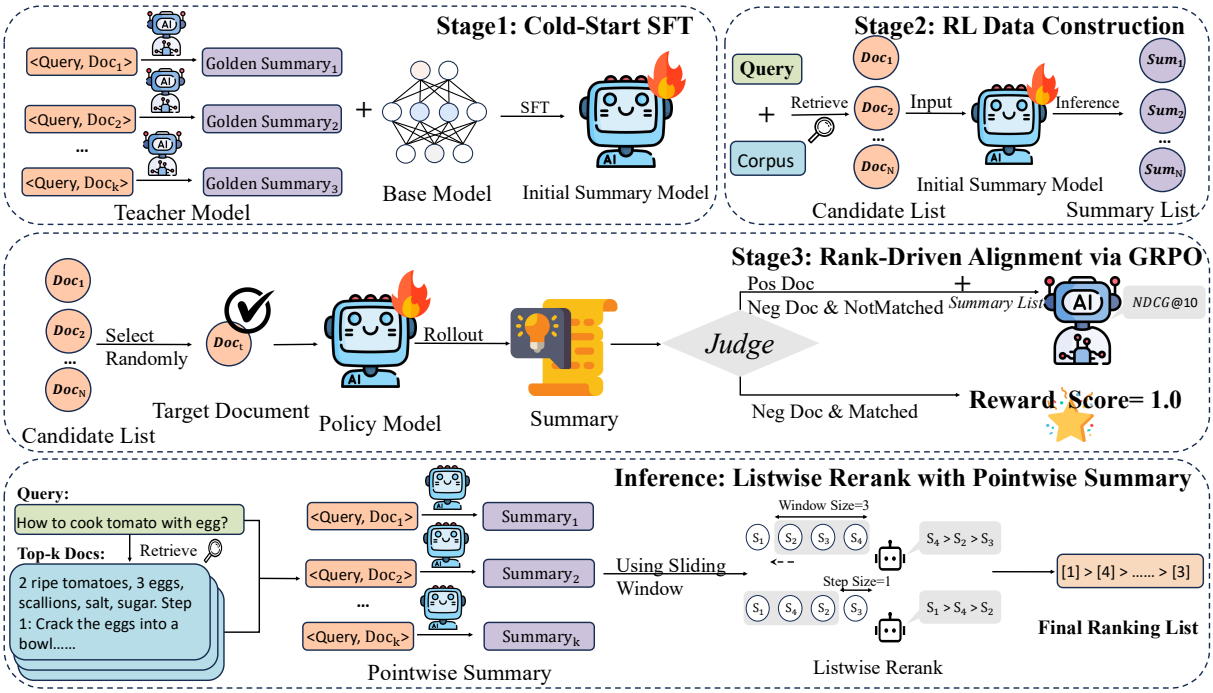

Figure 2: The overall architecture of SumRank, which consists of three training stages and the Summary-then-Rank pipeline (Listwise Rerank with Pointwise Summary).

3.4 Rank-Driven Alignment via GRPO

With the static summary list $\mathcal{L}_S$ established, we optimize the cold-start summarization model π_{SFT} using Group Relative Policy Optimization (GRPO) (Shao et al., 2024).

3.4.1 Target Selection and Rollout

During training for a given query q and its associated candidate document list $\mathcal{L}_D$, we randomly select a single target document D_t from the N-document list $\mathcal{L}_D$. Due to our data construction mechanism, D_t could be either a positive or a negative document.

Once D_t is selected, we formulate the input prompt $X_t = \mathcal{T}(q, D_t)$. To construct the comparison group required by GRPO, we prompt the active policy to auto-regressively sample a group of G candidate summary rollouts, denoted as $Y = \{y_1, y_2, \ldots, y_G\}$. Each rollout $y_i \in Y$ represents a variant action taken by the model to condense D_t.

3.4.2 Listwise Evaluation and Reward

The reward mechanism evaluates how every single rollout y_i affects the global ranking of its corresponding document. To evaluate a specific rollout $y \in Y$ with minimal computational overhead, we construct a mixed evaluation list $\hat{\mathcal{L}}_S$. We replace the t-th static summary s_t (representing D_t) in the pre-generated static Summary List $\mathcal{L}_S$ with the newly generated rollout y, while keeping the other $N-1$ summaries fixed. Formally, the assembled list is $\hat{\mathcal{L}}_S = \{s_1, \ldots, s_{t-1}, y, s_{t+1}, \ldots, s_N\}$.

This design choice is fundamentally driven by two critical motivations: reducing reward variance and ensuring computational efficiency. **First**, listwise ranking evaluates relative relevance. By fixing the $N-1$ non-target summaries using the baseline π_{SFT} model, we establish a stable reference frame. Forcing the model to compete against its own initial capabilities ensures that any rank improvement reflects a genuine gain in summarization quality, rather than environmental noise. **Second**, generating an entire N-document list for the G sampled rollouts required by GRPO would lead to cost overhead. The static background list reduces this generative cost from $\mathcal{O}(N \times G)$ to $\mathcal{O}(G)$ per step, making listwise RL practically feasible for long-document tasks.

An LLM reranker then performs listwise reranking on this updated N-document list. After the downstream listwise ranker scores and sorts the candidates in $\hat{\mathcal{L}}_S$, the reward for the generated summary y is computed. We define the reward signal $R(y)$ using the standard Normalized Discounted Cumulative Gain (NDCG@10) for the listwise reranking results. By directly optimizing this

ranking metric, the policy model learns to generate summaries that inherently maximize the comparative prominence of relevant documents.

To prevent the model from manufacturing false alignments to irrelevant candidates, we introduce a lexical matching indicator $M(y) \in \{0, 1\}$. Specifically, $M(y) = 1$ if the output y contains the safeguard phrase "No Relevant Information Found" (as explicitly instructed in our prompt template $\mathcal{T}$) or its closely related semantic variants.

The comprehensive reward dynamically conditions on the ground-truth label of the target document D_t:

$$r(y) = \begin{cases} R(y), & \text{if } D_t = D^+, \\ 1.0, & \text{if } D_t \neq D^+ \wedge M(y) = 1, \\ R(y) - \lambda, & \text{if } D_t \neq D^+ \wedge M(y) = 0, \end{cases} \quad (4)$$

where λ is a penalty factor applied when the model fails to explicitly reject a negative document but still undergoes ranking evaluation.

3.4.3 Optimization Objective

Finally, the reward r_i for each rollout y_i is Z-score normalized within the group to yield the relative advantage $\hat{A}_i = (r_i - \mu_Y)/\sigma_Y$. The active policy π_θ is then optimized by maximizing the streamlined GRPO objective:

$$\mathcal{J}_{\text{GRPO}}(\theta) = \frac{1}{|G|} \sum_{i=1}^{|G|} \frac{1}{|y_i|} \sum_{t=1}^{|y_i|} \min \Big(\rho_{i,t}(\theta) \hat{A}_{i,t}, \\ \text{clip}\big(\rho_{i,t}(\theta), 1 \pm \epsilon\big) \hat{A}_{i,t} \Big) - \beta D_{\text{KL}}, \quad (5)$$

where $|y_i|$ is the sequence length of y_i, ϵ is the clipping threshold, and β is the scaling coefficient for the KL divergence penalty.

$\rho_{i,t}(\theta)$ denotes the token-level probability ratio. Crucially, D_{KL} represents the token-level Kullback-Leibler divergence between the active policy π_θ and the cold-start reference model π_{SFT}, which effectively anchors the optimization space and prevents mode collapse. These two components are formally defined as follows:

$$\rho_{i,t}(\theta) = \frac{\pi_\theta(y_{i,t} \mid X, y_{i,<t})}{\pi_{\text{old}}(y_{i,t} \mid X, y_{i,<t})}, \quad (6)$$
$$D_{\text{KL}} = D_{\text{KL}}\big(\pi_\theta \parallel \pi_{\text{SFT}}\big).$$

4 Experimental Setup

4.1 Datasets and Metrics

We comprehensively evaluate our framework across diverse, real-world retrieval scenarios using the standard **TREC Deep Learning** (DL) tracks. Derived from the extensive MS MARCO search logs (Nguyen et al., 2016), the TREC DL benchmark serves as the gold standard for assessing ad-hoc retrieval systems.

To demonstrate the temporal robustness and consistent generalization of our approach against evolving query distributions, we expand our evaluation across five datasets of this benchmark: TREC DL19 (Craswell et al., 2020c), TREC DL20 (Craswell et al., 2021a), TREC DL21 (Craswell et al., 2021b), TREC DL22 (Craswell et al., 2025a), and TREC DL23 (Craswell et al., 2025b).

For the downstream listwise reranking phase, we employ the highly capable **Qwen2.5-32B-Instruct** (Yang et al., 2024) as our document reranker. To maximize overall ranking performance, we adopt the sliding window inference strategy (Sun et al., 2023) setting the window size to 20 and the step size to 10. We employ NDCG@10 and MAP@100 as the evaluation metric across all TREC DL datasets.

4.2 Baselines

We compare with three types of baselines.

Rule-based Methods We utilize BM25 to establish the initial ranking, alongside the widely adopted FirstP (Dai and Callan, 2019) strategy, which truncates documents to their first k tokens ($k \in \{128, 256, 512\}$) to leverage document "lead bias".

Seq2Seq Summarization Models We benchmark against sequence-to-sequence models explicitly pre-trained and fine-tuned for abstractive summarization tasks. This group includes the foundational BART-summarizer (Lewis et al., 2020), its long-document sparse-attention extension LED-summarizer (Beltagy et al., 2020), and PEGASUS (Zhang et al., 2020), which is specifically optimized for summarization generation.

LLM-based Summarization Models To assess the raw summarization capabilities of modern foundational models without our proposed listwise alignment, we employ the instruction-tuned Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct (Yang et al., 2024). We prompt these models to directly generate document summaries in a zero-shot manner.

Methods	DL 19		DL 20		DL 21		DL 22		DL 23		Avg.	
	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP	NDCG	MAP
BM25	51.76	24.34	52.86	37.93	51.16	21.26	29.93	8.01	29.46	10.46	43.03	20.40
Rule-based Methods												
FirstP-128	64.41	27.94†	61.60	42.33	66.84	24.94	40.82†	10.11	42.39†	14.07	55.21	23.88†
FirstP-256	65.22	27.53	61.07	41.60	67.56†	24.68	40.66	10.46†	41.58	13.94	55.22†	23.64
FirstP-512	65.94†	27.54	61.25	41.29	67.20	24.78	39.82	10.11	41.14	13.92	55.07	23.53
Seq2Seq Summarization Models												
BART-summarizer	62.05	26.79	61.48	41.29	67.53	25.14†	40.07	10.13	41.66	14.32†	54.56	23.53
LED-summarizer	59.80	24.77	57.82	38.80	67.22	25.06	40.18	10.11	41.36	14.11	53.28	22.57
PEGASUS	62.02	26.66	60.83	41.29	67.53	25.13	40.46	10.13	41.84	14.28	54.54	23.50
LLM-based Summarization Models												
Qwen2.5-3B-Instruct	58.97	27.32	54.45	37.05	63.67	23.90	37.17	9.50	35.52	12.70	49.96	22.09
Qwen2.5-7B-Instruct	59.72	25.88	62.03†	42.45†	66.15	24.40	35.40	9.15	40.24	13.50	52.71	23.08
Our Methods												
SumRank (3B)	67.06	28.53	**62.99**	**42.83**	68.09	25.43	42.22	10.67	43.76	14.85	56.82	24.46
SumRank (7B)	**67.30**	**28.81**	62.95	42.66	**69.30**	**25.96**	**42.74**	**10.96**	**44.31**	**14.95**	**57.32**	**24.67**

Table 1: The results (NDCG@10 and MAP@100) on TREC DL 19-23 benchmark. All baselines rerank the BM25-retrieved top-100 documents. The best results are highlighted in **bold**, and the second-best results are underlined. The strongest baseline in each column is marked with †.

4.3 Implementation Details

For the Cold-Start SFT stage, we employ LLaMA-Factory (Zheng et al., 2024) to perform full-parameter fine-tuning on Qwen2.5-3B-Instruct and Qwen2.5-7B-Instruct. Both models are trained on a 40k-instance distillation dataset, derived from the MS MARCO dataset (Nguyen et al., 2016) and generated by the teacher model, Qwen2.5-72B-Instruct. This stage is trained for 5 epochs with a learning rate of 5×10^{-6} using 4 NVIDIA A800 (80GB) GPUs.

For the Data Construction stage, we sample 2,500 queries from MS MARCO (Nguyen et al., 2016) to construct 2,500 corresponding candidate document groups. We set N to 10 and k to 1. During the subsequent GRPO training, the policy rollout for each query group is executed by one positive document and one negative document.

For the Rank-Driven Alignment via GRPO stage, we employ the verl framework (Sheng et al., 2024) for reinforcement learning. We initialize the policy with the SFT checkpoint. We train the 3B model on two NVIDIA A800 (80GB) GPUs and the 7B model on four NVIDIA A800 (80GB) GPUs, both using a reduced learning rate of 1×10^{-6}. We sample $G = 8$ rollouts per prompt, train for 5 epochs and set the KL divergence penalty $\beta = 0.001$. We employ Qwen2.5-72B-Instruct as the listwise LLM reranker to compute the NDCG@10 reward, with the penalty λ set to 0.25.

5 Experimental Results

5.1 Main Results

Table 1 presents the overall reranking performance of our proposed SumRank and all baseline methods across five TREC DL benchmarks.

SumRank (7B) achieves state-of-the-art performance across all five datasets, with SumRank (3B) closely following and consistently surpassing all competitive baselines. By directly optimizing the summarizer for the ranking task, the model successfully preserves the necessary matching signals. Beyond absolute performance metrics, the comparative analysis validates the fundamental advantages of aligning targeted summarization with downstream ranking objectives.

Comparison with Rule-based Methods Naive truncation strategies (*e.g.*, FirstP-256) establish surprisingly strong baselines. This is mainly because they take advantage of the "lead bias", which means the tendency for web pages to put key information at the very beginning. However, this simple cut-off approach inevitably loses important details hidden later in the text. SumRank solves this problem by compressing the entire document. Instead of just looking at the beginning, it successfully captures the valuable evidence from the middle and end of the text that truncation simply throws away.

Comparison with Seq2Seq Summarization Models The results of classic summarizers (BART-summarizer, LED-summarizer, PEGASUS) clearly

show that their original goals do not match the ranking task. Even with complex designs, they often score worse than simple truncation. This happens because standard summarization training actually filters out the text. Since these models are trained to write smooth, general overviews for human readers, they accidentally delete the specific keywords and exact details that a ranking model desperately needs to calculate relevance.

Comparison with LLM-based Summarization Models The comparison between SumRank (3B) and the much larger Qwen2.5-7B-Instruct highlights that zero-shot summaries generated by general LLMs simply cannot meet the strict requirements of downstream ranking. However, once our 3B model undergoes the cold-start SFT and rank-driven alignment via the GRPO training pipeline, its ranking performance improves dramatically, easily beating the 7B baseline. This clearly shows that specifically training a smaller model to preserve relevance is far more powerful than relying on the zero-shot capabilities of a larger model.

5.2 Ablation Study

To investigate the contribution of different training strategies, we conducted an ablation study by removing the cold-start SFT stage and rank-driven alignment via the GRPO stage independently. As shown in Table 2, the full model consistently achieves the best performance across the TREC DL dataset, confirming that both components are indispensable.

When we remove the rank-driven alignment via the GRPO stage (*i.e.*, "*w/o* rank-driven GRPO "), the average score suffers the biggest drop. This proves that relying on SFT alone is not enough. While SFT is great at teaching the model the basic format and length of a summary, it cannot fully align the model with the final ranking goal. Without the targeted rewards provided by rank-driven alignment via GRPO, the model easily slips back into generating generic text and forgets to highlight the specific keywords needed for rerank.

Removing the initial SFT phase (*i.e.*, "*w/o* cold-start SFT") also hurts the final results. This happens because reinforcement learning needs a solid starting point. Without SFT to first show the model exactly what a proper summary should look like, the GRPO training process has a much harder time figuring out the right direction from scratch, leading to less stable and suboptimal learning.

Model Variant	**TREC Avg.**	Δ
SumRank (3B)	**56.82**	-
Training Approach		
• *w/o* Cold-Start SFT	55.88	-0.94
• *w/o* Rank-Driven GRPO	54.92	-1.90
Data Construction		
• Truncation FirstP128	54.88	-1.94
• Truncation FirstP256	54.92	-1.90
• Update per epoch	56.32	-0.50

Table 2: Ablation study and impact anlysis of data construction stage. "TREC Avg." reports the average NDCG@10 scores evaluated across all five TREC DL benchmarks (DL19–23). Δ indicates the absolute performance degradation on specific training stages or different data construction strategies.

Ultimately, the full pipeline achieves the highest average score. This confirms that the cold-start SFT stage and the rank-driven alignment via the GRPO stage perfectly complement each other. SFT builds the necessary foundation, and GRPO acts as the crucial final step to lock in those vital matching signals for the downstream reranking.

5.3 Impact of Data Construction Stage

As presented in Table 2, we investigate the impact of different data construction strategies for representing non-target documents.

Replacing our generated summaries with simple truncated raw texts (*i.e.*, "Truncation FirstP128/256") leads to a performance drop. Notably, whether we truncate at 128 or 256 tokens, the degradation remains equally severe. This clearly demonstrates that naive truncation is fundamentally ineffective. Raw document prefixes are dominated by noise and fail to provide the policy model with relevance signals.

Furthermore, dynamically updating the context summaries at every training epoch (*i.e.*, "Update per epoch"), instead of using our default static summaries, unexpectedly decreases the score by 0.50 points. This confirms that keeping the context summaries static provides a stable and consistent reference frame for the model during RL training, whereas a constantly shifting background introduces instability. More importantly, regenerating summaries every epoch incurs computational overhead. Therefore, our simple static strategy proves to be both more efficient and more effective.

5.4 Efficiency Analysis

Method	NDCG@10	Latency
Qwen2.5-72B-Instruct	56.99	83
SumRank (3B)	56.82	1.95
SumRank (7B)	57.32	6.98

Table 3: Ranking performance and summary generation inference latency (seconds/query) on the average scores evaluated across all five TREC DL benchmarks (DL19–23).

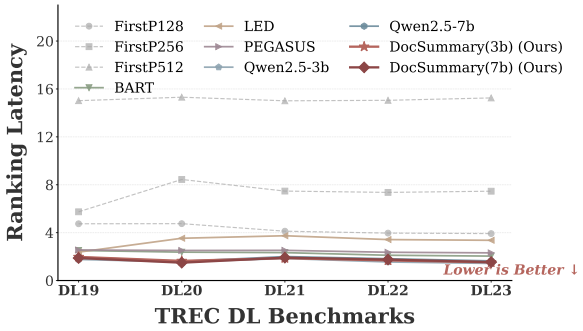

Figure 3: Ranking latency (seconds/query) across different methods on all five TREC DL benchmarks (DL19–23).

To comprehensively evaluate the practical value of our framework, we investigate how SumRank and the baseline models impact the overall efficiency of the downstream ranking task. Figure 3 illustrates the ranking latency (measured in seconds/queries) to rerank summaries from different methods across the five TREC DL datasets. The results reveal several critical insights regarding the efficiency of different reranking pipelines.

The Severe Bottleneck of Long Contexts The rule-based truncation methods (FirstP series) suffer from significant computational overhead. Even the shorter FirstP128 and FirstP256 are notably slower than our method. Because the self-attention mechanism in LLMs scales quadratically, feeding raw long documents directly into a ranker inherently causes severe latency bottlenecks. Relying on raw text, even when truncated, is not a scalable solution for long-document ranking. SumRank completely bypasses this by acting as an intelligent pre-compressor, condensing lengthy texts into a highly compact format.

High Efficiency of SumRank In sharp contrast, SumRank demonstrates exceptional efficiency. Both SumRank (3B) and SumRank (7B) maintain an extremely low latency, which is faster than the traditional Seq2Seq summarizers. This proves that the summarization step in SumRank does not add noticeable overhead. Instead, by highly compressing the documents into short, dense summaries beforehand, it heavily reduces the workload for the downstream ranker.

Trade-off-Breaking Performance When combining these latency results with the ranking accuracy from Table 1, the acceleration of SumRank delivers robust inference latency without any degradation in accuracy. While FirstP256 achieves decent accuracy, it is extremely slow. SumRank is the only method that achieves state-of-the-art accuracy while keeping the latency at the lowest level. This confirms that our rank-driven summarization effectively resolves the tension between processing speed and ranking quality.

5.5 Analysis of Performance Upper Bound

To understand the upper bound of our lightweight framework, we compare SumRank against a massive parameter model, Qwen2.5-72B-Instruct.

Initially, we considered the 72B model's performance as a theoretical upper bound. As shown in Table 3, SumRank (3B) closely approaches this upper bound while achieving an extreme 42x acceleration (1.95s vs. 83s) using a model 24 times smaller.

However, scaling our framework to 7B parameters entirely surpasses this expected upper bound. SumRank (7B) not only outperforms the 72B in ranking effectiveness but also maintains a nearly 12x speed advantage. This confirms that smart, task-aware alignment can outperform a massive LLM, saving significant computational effort without degrading performance.

6 Conclusion

We present SumRank, a summarization model aligned with downstream document listwise reranking. We designed a tailored three-stage training paradigm to synthesize high-density relevance features and utilize an LLM reranker for rank-aware preference alignment. Through this well-structured training process, SumRank effectively provides the downstream ranker with the exact matching signals extracted from the original long documents, achieving a significant improvement in ranking performance while reducing the downstream ranking latency.

Limitations

Although SumRank significantly improves the efficiency of long-document reranking, it has certain limitations. Firstly, due to our limited resources, we cannot experiment with larger open-source LLMs, such as those 14B or even 32B models. It is interesting to investigate the alignment of larger scale models. Secondly, our approach relies on a two-stage "Summary-then-Rank" pipeline. This means we still need to deploy and maintain two separate models: one for summarization and another for ranking. In the future, we plan to explore a unified, end-to-end architecture. We aim to train a single model that can simultaneously compress the document and output the final relevance score, which will further simplify deployment and reduce inference latency.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *CoRR*, abs/2004.05150.

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, and Daniel Campos. 2021a. Overview of the TREC 2020 deep learning track. *CoRR*, abs/2102.07662.

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. 2021b. Overview of the TREC 2021 deep learning track. In *TREC*, volume 500-335 of *NIST Special Publication*. National Institute of Standards and Technology (NIST).

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2025a. Overview of the TREC 2022 deep learning track. *CoRR*, abs/2507.10865.

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.

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

Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Hossein A. Rahmani, Daniel Campos, Jimmy Lin, Ellen M. Voorhees, and Ian Soboroff. 2025b. Overview of the TREC 2023 deep learning track. *CoRR*, abs/2507.08890.

Zhuyun Dai and Jamie Callan. 2019. Deeper text understanding for IR with contextual neural language modeling. In *SIGIR*, pages 985–988. ACM.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qianyu Guo, Meng Wang, and Haofen Wang. 2023. Retrieval-augmented generation for large language models: A survey. *CoRR*, abs/2312.10997.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Sebastian Hofstätter, Bhaskar Mitra, Hamed Zamani, Nick Craswell, and Allan Hanbury. 2021. Intra-document cascading: Learning to select passages for neural document ranking. In *SIGIR*, pages 1349–1358. ACM.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alex Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *ACL (Findings)*, volume ACL 2023 of *Findings of ACL*, pages 8003–8017. Association for Computational Linguistics.

Jyun-Yu Jiang, Chenyan Xiong, Chia-Jung Lee, and Wei Wang. 2020. Long document ranking with query-directed sparse transformer. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 4594–4605. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880. Association for Computational Linguistics.

Canjia Li, Andrew Yates, Sean MacAvaney, Ben He, and Yingfei Sun. 2024. PARADE: passage representation aggregation fordocument reranking. *ACM Trans. Inf. Syst.*, 42(2):36:1–36:26.

Minghan Li, Miyang Luo, Tianrui Lv, Yishuai Zhang, Siqi Zhao, Ercong Nie, and Guodong Zhou. 2025a. A survey of long-document retrieval in the PLM and LLM era. *CoRR*, abs/2509.07759.

Xiaoxi Li, Jiajie Jin, Guanting Dong, Hongjin Qian, Yongkang Wu, Ji-Rong Wen, Yutao Zhu, and Zhicheng Dou. 2025b. Webthinker: Empowering large reasoning models with deep research capability. *arXiv preprint arXiv:2504.21776*.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, Yuanxing Zhang, Zhuo Chen, Hangyu Guo, Shilong Li, Ziqiang Liu, Yong Shan, Yifan Song, Jiayi Tian, Wenhao Wu, and 18 others. 2025a. A comprehensive

survey on long context language modeling. *CoRR*, abs/2503.17407.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. 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. 2025b. Leveraging passage embeddings for efficient listwise reranking with large language models. In *WWW*, pages 4274–4283. ACM.

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

Wenhan Liu, Xinyu Ma, Yutao Zhu, Yuchen Li, Daiting Shi, Dawei Yin, and Zhicheng Dou. 2026a. Agentic-r: Learning to retrieve for agentic search. *arXiv preprint arXiv:2601.11888*.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Lixin Su, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025d. Coranking: Collaborative ranking with small and large ranking agents. In *EMNLP (Findings)*, pages 5098–5110. Association for Computational Linguistics.

Wenhan Liu, Xinyu Ma, Yutao Zhu, Ziliang Zhao, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2025e. Sliding windows are not the end: Exploring full ranking with long-context large language models. In *ACL (1)*, pages 162–176. Association for Computational Linguistics.

Wenhan Liu, Yutao Zhu, Zhicheng Dou, and Yujia Zhou. 2026b. Demorank: Selecting effective demonstrations for large language models in ranking task. *ACM Transactions on Information Systems*, 44(3):1–25.

Xueguang Ma, Xinyu Zhang, Ronak Pradeep, and Jimmy Lin. 2023. Zero-shot listwise document reranking with a large language model. *CoRR*, abs/2305.02156.

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 *CoCo@NIPS*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. 2020. Document ranking with a pre-trained sequence-to-sequence model. In *EMNLP (Findings)*, 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, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *NAACL-HLT (Findings)*, pages 1504–1518. Association for Computational Linguistics.

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.

Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. 2024. Hybridflow: A flexible and efficient rlhf framework. *arXiv preprint arXiv: 2409.19256*.

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 *EMNLP*, 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*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Zhijing Wu, Jiaxin Mao, Yiqun Liu, Jingtao Zhan, Yukun Zheng, Min Zhang, and Shaoping Ma. 2020. Leveraging passage-level cumulative gain for document ranking. In *WWW*, pages 2421–2431. ACM / IW3C2.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *CoRR*, abs/2412.15115.

Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2025. Inference scaling for long-context retrieval augmented generation. In *ICLR*. OpenReview.net.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Wenlin Zhang, Xiaopeng Li, Yingyi Zhang, Pengyue Jia, Yichao Wang, Huifeng Guo, Yong Liu, and Xiangyu Zhao. 2025. Deep research: A survey of autonomous research agents. *arXiv preprint arXiv:2508.12752*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *ACL (3)*, pages 400–410. Association for Computational Linguistics.

Yujia Zhou, Zhicheng Dou, Huaying Yuan, and Zhengyi Ma. 2022. Socialformer: Social network inspired long document modeling for document ranking. In *WWW*, pages 339–347. ACM.

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. 2024. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. In *NAACL (Short Papers)*, pages 358–370. 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 *SIGIR*, pages 2308–2313. ACM.

A Prompt for SumRank

Prompt

Task: Generate the summary from the Document that answer the Query.

Strict Rules:

1. If the Document is **NOT** relevant to the Query, you **MUST** output exactly: 'No relevant information found.'
2. Do **NOT** provide explanations, apologies, or conversational fillers like 'The document says...'. Just the answer or the refusal phrase.

Query: {query}

Document:
{document}

B Use of AI Assistants

We use Gemini[1] to improve the presentations of this paper.

[1] https://gemini.google.com/