Rank4Gen: RAG-Preference-Aligned Document Set Selection and Ranking

Anonymous ACL submission

Abstract

In the RAG paradigm, the information retrieval module provides context for generators by retrieving and ranking multiple documents to support the aggregation of evidence. However, existing ranking models are primarily optimized for query–document relevance, which often misaligns with generators' preferences for evidence selection and citation, limiting their impact on response quality. Moreover, most approaches do not account for preference differences across generators, resulting in unstable cross-generator performance. We propose **Rank4Gen**, a generator-aware ranker for RAG that targets the goal of *Ranking for Generators*. Rank4Gen introduces two key preference modeling strategies: (1) **From Ranking Relevance to Response Quality**, which optimizes ranking with respect to downstream response quality rather than query–document relevance; and (2) **Generator-Specific Preference Modeling**, which conditions a single ranker on different generators to capture their distinct ranking preferences. To enable such modeling, we construct **PRISM**, a dataset built from multiple open-source corpora and diverse downstream generators. Experiments on five challenging and recent RAG benchmarks demonstrate that Rank4Gen achieves strong and competitive performance for complex evidence composition in RAG.

1 Introduction

Retrieval-Augmented Generation (RAG) has emerged as a powerful framework for grounding large language model (LLM) outputs in external knowledge, enabling access to up-to-date information and reducing hallucinations in knowledge-intensive tasks (Lewis et al., 2020; Izacard and Grave, 2021; Huang et al., 2025). In a RAG system, an information retrieval module retrieves and ranks documents, which are provided as contextual evidence to an LLM generator for answer generation. Consequently, the quality of document ranking plays a critical role in determining the effectiveness of RAG.

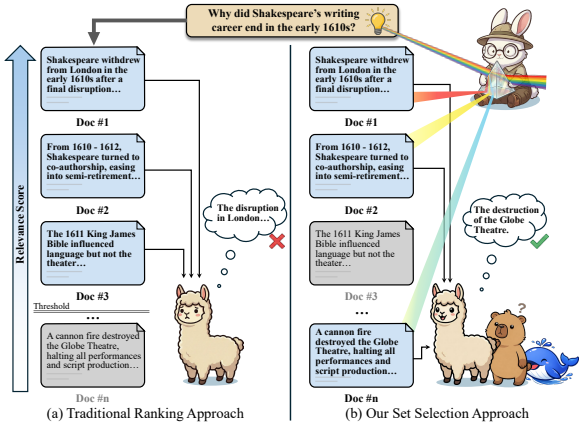

Figure 1: Comparison between traditional ranking and Rank4Gen in RAG. Traditional relevance-based ranking may truncate crucial evidence, while Rank4Gen performs generator-aware document set selection.

Most existing RAG systems rely on retrieval and ranking models optimized for query–document relevance, such as dense retrievers, neural rerankers, and more recent LLM-based ranking approaches (Karpukhin et al., 2020; Nogueira and Cho, 2019; Sun et al., 2023; Fan et al., 2025). While relevance-based ranking is effective for identifying documents related to the query, relevance alone does not fully reflect generators' preferences for evidence usage during generation. Recent studies show that retrieved documents containing correct answers may still fail to support correct generation, whereas seemingly less relevant documents can sometimes better facilitate reasoning and answer synthesis (Tian et al., 2025). As a result, the mismatch between retrieval relevance and generation utility limits the effectiveness of ranking improvements on downstream response quality.

Moreover, most existing ranking approaches assume a generator-agnostic setting, applying a sin-

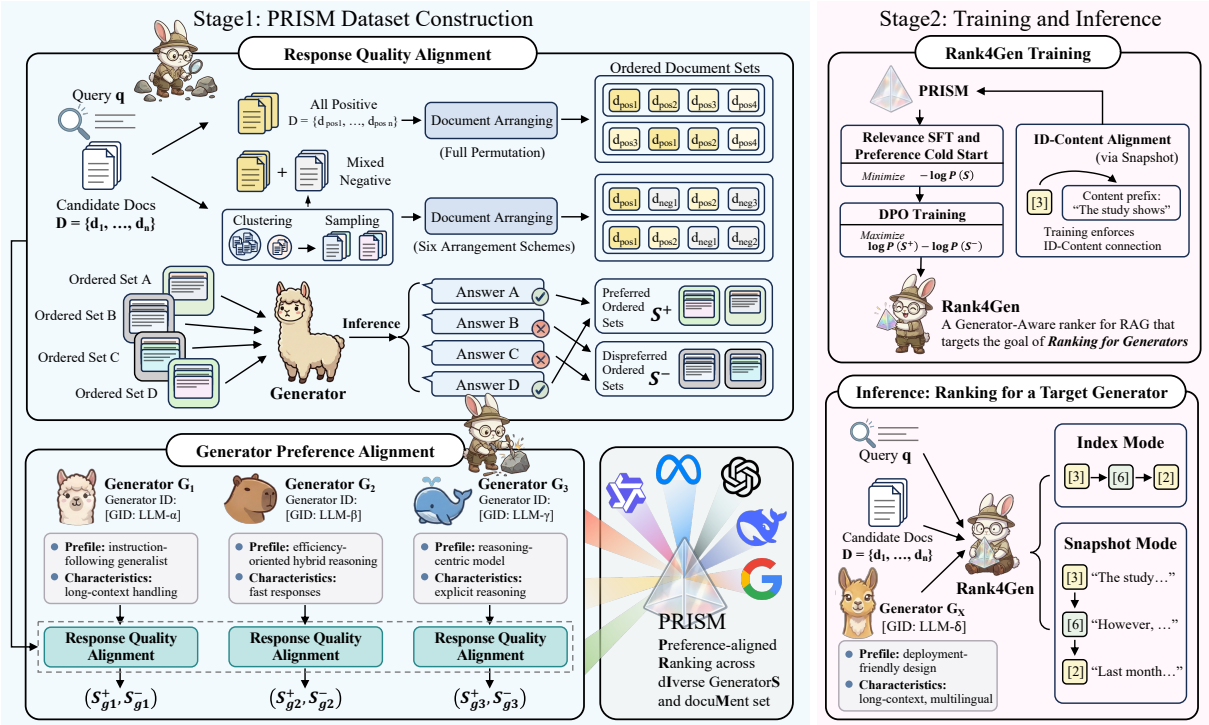

Figure 2: Overview of Rank4Gen. The framework has two stages: (1) PRISM Dataset Construction, which derives generator-aware ranking preferences aligned with downstream response quality; (2) Training and Inference, where Rank4Gen is trained with SFT and DPO on PRISM for generator-aware document set selection and ranking.

gle ranking strategy regardless of the downstream generator. In practice, different generators vary in how they utilize retrieved context, which can cause document selection and ordering strategies to generalize poorly across generators, leading to unstable performance. This issue is further exacerbated by the sensitivity of LLMs to context organization, where redundant or distracting evidence can degrade generation quality (Liu et al., 2024; Laban et al., 2024). Recent work has explored set-level formulations and LLM-based ranking approaches for RAG to better support evidence aggregation across multiple documents (Pradeep et al., 2023; Liu et al., 2025b; Lee et al., 2025). However, these methods remain largely optimized for relevance-based objectives and do not explicitly align ranking decisions with downstream generation quality or account for generator-specific preferences.

To address these issues, we pursue the goal of *Ranking for Generators* and propose **Rank4Gen**, a generator-aware ranking model for RAG. Rank4Gen aligns ranking decisions with downstream generation quality rather than query–document relevance. It incorporates two key preference modeling strategies: (1) **From Ranking Relevance to Response Quality**, which bases ranking decisions on the quality of generated responses; and (2) **Generator-Specific Preference Modeling**, which conditions a single ranker on different generators to capture their distinct ranking preferences. To support such modeling, we construct **PRISM** (Preference-aligned Ranking across dIverse generatorS and docuMent sets), a dataset built from multiple open-source corpora and diverse downstream generators. Rank4Gen adopts a set selection paradigm that outputs an ordered set of documents and is trained using a two-stage training procedure. We first perform supervised fine-tuning (SFT) to learn relevance-based document selection behavior, while also bootstrapping generator-aware conditioning and structured output capabilities. We then apply Direct Preference Optimization (DPO) (Rafailov et al., 2023) to directly optimize generator-aligned ranking preferences.

We evaluate Rank4Gen on a diverse set of challenging and recent RAG benchmarks, designed to test complex evidence aggregation and generalization. We compare Rank4Gen against a broad range of baselines, including relevance-based, LLM-based, and set selection approaches, and conduct extensive ablation and generalization analyses. Experimental results show that Rank4Gen demonstrates strong and competitive performance, particularly in terms of overall downstream generation

quality and robustness across generators.

Our contributions are summarized as follows:

- We propose **Rank4Gen**, a generator-aware ranking model for RAG that explicitly targets the goal of *Ranking for Generators*, aligning ranking decisions with downstream generation quality and generator preferences.
- We introduce **PRISM**, a preference-aligned dataset that provides unified supervision for generator-aware document set selection and ranking across multiple downstream generators, two languages (English and Chinese), and diverse document sources.
- We conduct extensive experiments on five challenging and recent RAG benchmarks, demonstrating that **Rank4Gen** achieves strong and competitive downstream generation quality and robustness across generators.

2 Related Work

2.1 Ranking Methods and Paradigms

Traditional information retrieval ranking methods typically follow pointwise, pairwise, or listwise paradigms. Pointwise methods independently estimate the relevance of each document to a given query (Nogueira and Cho, 2019). Pairwise methods compare pairs of documents to infer their relative ordering (Qin et al., 2024), while listwise methods model the entire candidate document list jointly (Tang et al., 2024). With the rapid development of LLMs, recent work has explored leveraging their strong capabilities for ranking. Some approaches directly employ LLMs via prompting to score candidate documents (Sun et al., 2023). Other methods focus on distillation, transferring the ranking ability of LLMs into smaller models, such as RankZephyr (Pradeep et al., 2023). More recent studies further introduce reinforcement learning, optimizing ranking policies using rewards derived from ranking metrics or preference signals (Sun et al., 2025a,b).

2.2 LLM for Ranking in RAG

Early RAG systems construct generation contexts via fixed Top-K retrieval or adaptive depth selection (Lewis et al., 2020; Jeong et al., 2024). Subsequent work incorporates LLMs for reranking, either to improve ranking effectiveness (Yu et al., 2024; Zhang et al., 2025b) or to enhance reasoning efficiency and reduce inference cost (Reddy et al., 2024; Liu et al., 2025a). Other studies reformulate ranking as document subset selection to control better context composition (Meng et al., 2025; Lee et al., 2025; Zhang et al., 2025a). Despite these advances, most methods optimize relevance-based objectives rather than downstream generation quality. Recent work has begun exploring joint optimization of rankers and generators (Shi et al., 2025).

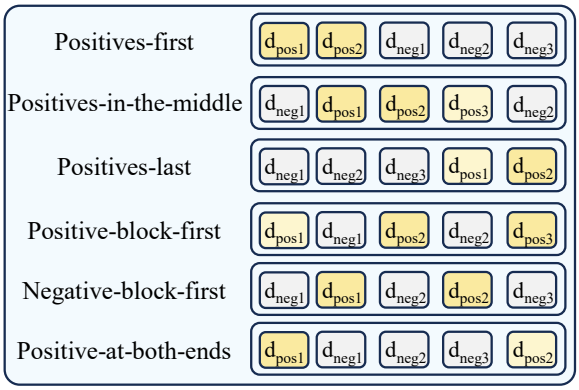

Figure 3: Six predefined Arrangement Schemes.

3 Methodology

3.1 Preliminary

Before presenting Rank4Gen, we first introduce two preliminaries. We begin by analyzing preference phenomena in RAG, showing how different document subsets and their ordering can substantially affect downstream generation quality, and how such preferences vary across generators. Based on these observations, we then formalize the ranking task adopted in this work, which is defined as selecting and ordering a document subset for RAG, rather than ranking the entire candidate set.

3.1.1 Preference Phenomena in RAG

We conduct a controlled analysis to study how different document subsets and their ordering affect RAG response quality across downstream generators. Under a fixed retrieval setting, we evaluate multiple ordered document subsets for each query and observe clear preference phenomena in downstream generation.

For each query, we use a widely adopted dense retriever bge-m3 (Chen et al., 2024) to retrieve the top-5 candidate documents. Based on this candidate pool, we construct multiple ordered document subsets under two settings. When all positive documents are available, we enumerate all permutations

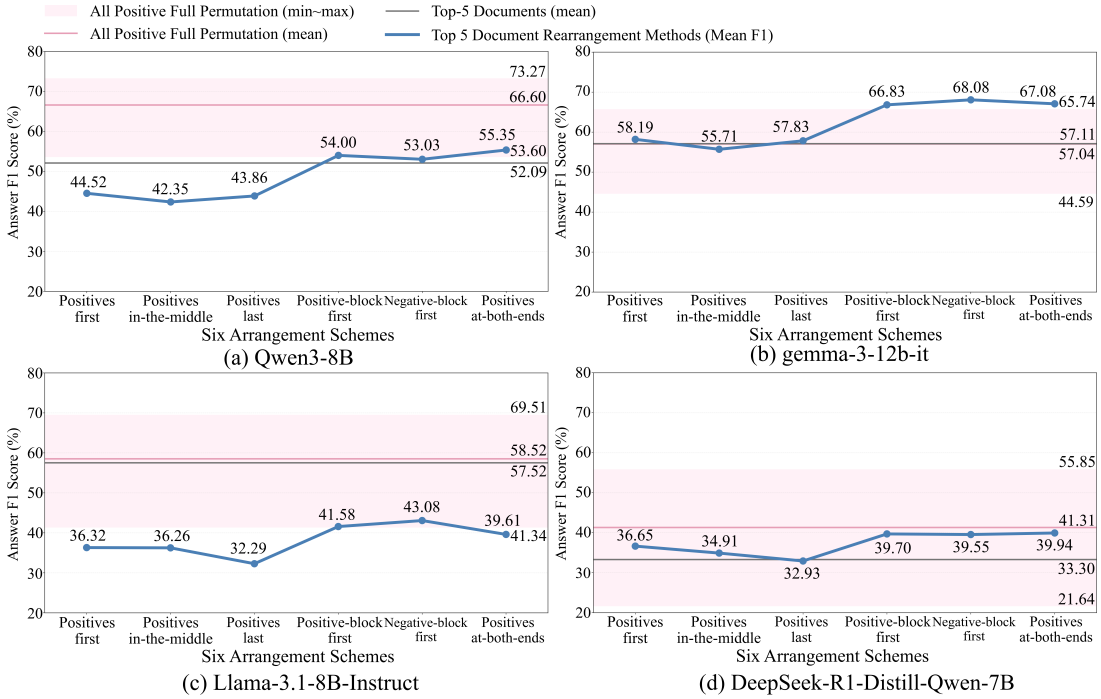

Figure 4: Impact of document subset composition and ordering on RAG performance across generators.

of the positive set to obtain different document orders. For retrieved document sets, we rearrange documents according to six predefined schemes shown in Figure 3. Each ordered subset is then provided to a downstream generator, and the resulting response quality is evaluated against the ground-truth answer.

Figure 4 summarizes the results across multiple downstream generators. Different document subsets and orderings lead to substantial performance variations, even when all documents are individually relevant, indicating that RAG performance depends not only on relevance but also on evidence composition and ordering. Moreover, preferences over document subsets vary across generators, as subsets that perform well for one generator may perform poorly for another. These observations motivate a generator-aware set selection and ranking formulation that reasons over ordered document subsets, as adopted by Rank4Gen.

3.1.2 Task Definition

In the RAG setting, we consider ranking as a generator-aware document set selection problem. Given a query q, some candidate documents $\mathcal{D} = \{d_1, \ldots, d_N\}$, and a downstream generator G, the goal of the ranker is to select a document subset and produce an ordering over it. The resulting ordered subset $\mathcal{S} = (d_{i_1}, d_{i_2}, \ldots, d_{i_k})$, where $\mathcal{S} \subseteq \mathcal{D}$, is directly provided to the generator as contextual evidence for RAG. Under this formulation, our ranker adopts a set selection paradigm that outputs an ordered set of documents tailored to the downstream generator, rather than producing a full ordering over all candidates. Figure 1 provides an intuitive diagram comparing the different paradigms.

3.2 PRISM Dataset Construction

To support Rank4Gen, we construct **PRISM** (Preference-aligned Ranking across dIverse generatorS and docuMent sets), a preference-aligned dataset built from multiple open-source corpora and diverse downstream generators. PRISM provides supervision for generator-conditioned document set selection and ranking. We next describe its design principles and construction.

3.2.1 Two Preference-Modeling Strategies

Document relevance remains a necessary foundation for retrieval and ranking in RAG. However, the preference phenomena in Section 3.1.1 suggest that relevance alone is insufficient for the goal of *Ranking for Generators*. In particular, downstream generation quality is influenced by the composition and ordering of document subsets. To address these challenges, Rank4Gen incorporates two complementary preference modeling strategies on top of relevance-based supervision.

Dataset	Samples	Language
PRISM_13K		
HotpotQA (Yang et al., 2018)	4,000	EN
2WikiMultiHopQA (Ho et al., 2020)	2,000	EN
MUSIQUE (Trivedi et al., 2022)	2,000	EN
MS MARCO (Nguyen et al., 2016)	2,000	EN
CRUD-RAG (Lyu et al., 2025)	2,994	ZH
Evaluation Datasets		
BrowseComp-Plus (Chen et al., 2025a)	830	EN
KG-MHQA (Wang et al., 2025)	307	EN
ChronoQA (Chen et al., 2025b)	5,173	ZH
SimpleQA (Wei et al., 2024)	4,326	EN
ChineseSimpleQA (He et al., 2025)	3,000	ZH

Table 1: Overview of training and evaluation datasets. Additional details on data collection and construction are provided in the Section 3.2.2 and Appendix C.

From Ranking Relevance to Response Quality. Traditional ranking models are typically optimized using query–document relevance signals, which do not directly reflect the utility of a document set for downstream generation. In contrast, Rank4Gen models preferences over ordered document subsets using the quality of the downstream responses they induce. By incorporating response quality as an additional training signal, Rank4Gen aligns ranking preferences with the generator's evidence usage behavior, optimizing downstream RAG performance beyond relevance alone.

Generator-Specific Preference Modeling. As shown in Section 3.1.1, different generators exhibit distinct preferences over document subsets. Rank4Gen explicitly accounts for this variability by conditioning a single ranker on the downstream generator. This enables a single ranker to learn generator-specific preferences while sharing parameters, adapting document selection and ordering to the conditioned generator.

3.2.2 Data Construction Process

PRISM is a large-scale preference dataset designed to provide generator-conditioned supervision for ordered document set selection and ranking in RAG. As shown in the upper part of Figure 2, PRISM is built through a three-stage pipeline spanning data collection, response quality alignment, and generator-conditioned preference alignment. It captures the impact of document composition and ordering on generation quality, as well as systematic preference differences across generators.

Data Collection. We collect data from a set of publicly available datasets that are widely used in RAG and multi-document question answering, including HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MUSIQUE (Trivedi et al., 2022), MS MARCO (Nguyen et al., 2016), and CRUD-RAG (Lyu et al., 2025). These datasets span a broad range of domains, query types, and reasoning requirements, from factoid retrieval to multi-hop and compositional reasoning. We filter and preprocess the raw data based on data quality, document availability, and language, resulting in a bilingual query–document corpus with queries in both English and Chinese. In total, this stage yields a unified corpus containing 141k queries. Based on this corpus, we apply two preference-modeling strategies to construct generator-aligned supervision: response quality alignment and generator preference alignment.

Response Quality Alignment. The first step is response quality alignment, which aims to identify suitable ordered document subsets for a query based on downstream response quality under a fixed generator G. Given a query q, we construct multiple ordered document subsets by enumerating all permutations of its positive documents, resulting in subsets that contain only relevant evidence.

To further increase subset diversity and better reflect realistic retrieval noise, we additionally construct ordered subsets by mixing all positive documents with a small number of negative documents. To select negative documents, we cluster candidate documents along three dimensions: document length, dense semantic similarity computed using cosine similarity of bge-m3 embeddings, and sparse semantic similarity based on TF-IDF representations. We then sample documents from different clusters to form diverse ordered subsets, with the cluster distribution illustrated in Figure B1.

For each ordered document subset $\mathcal{S}$, we provide $\mathcal{S}$ to the generator G as contextual evidence to produce an answer, and evaluate the generated response against the ground-truth using an automatic response quality metric implemented via a Listwise LLM-as-a-judge, which assesses both the reasoning process and the final answer. The specific prompt is shown in Figure B8 and B9. By comparing response quality across subsets, we derive preference labels and obtain preferred ordered sets $\mathcal{S}^+$ and dispreferred ordered sets $\mathcal{S}^-$.

Generator Preference Alignment. The second step is generator preference alignment, whose goal is to enable the ranker to recognize and adapt to the specific downstream generator it serves. To

Ranker	Generator	BrowseComp+ EM	BrowseComp+ F1	KG-MHQA EM	KG-MHQA F1	ChronoQA EM	ChronoQA F1	SimpleQA EM	SimpleQA F1	CN-SimpleQA EM	CN-SimpleQA F1	Avg. EM	Avg. F1
Without RAG													
-	Qwen3-8B	0.12	3.53	0.65	3.36	2.84	24.18	2.80	9.73	14.67	46.27	4.22	17.41
	gemma-3-12b-it	0.00	2.45	0.65	3.21	1.70	13.66	4.28	11.54	9.07	34.28	3.14	13.03
	Llama-3.1-8B-Instruct	0.00	1.50	0.65	2.76	2.01	20.77	4.05	9.45	6.30	30.61	2.60	13.02
	DeepSeek-R1-Distill-Qwen-7B	0.12	1.62	0.33	2.48	1.45	21.80	1.16	5.16	2.50	26.69	1.11	11.55
Pointwise$^{\tau}$													
Pointwise-Vanilla	Qwen3-8B	46.02	55.20	6.19	11.40	9.39	36.83	60.56	75.75	30.83	48.09	30.60	45.45
	gemma-3-12b-it	39.44	46.64	6.19	11.23	3.15	16.77	60.61	76.30	24.83	42.94	26.84	38.78
	Llama-3.1-8B-Instruct	33.38	42.47	5.86	9.08	5.39	33.52	58.34	73.66	15.80	36.60	23.75	39.07
	DeepSeek-R1-Distill-Qwen-7B	8.31	15.47	5.54	9.30	0.81	21.14	14.24	35.55	17.07	34.10	9.19	23.11
Listwise$^{\tau}$													
Listwise-Vanilla	Qwen3-8B	29.88	37.77	7.82	11.52	8.25	34.13	59.69	74.79	32.60	49.40	27.65	41.52
	gemma-3-12b-it	25.42	31.08	7.82	12.20	3.17	16.12	60.33	75.65	27.03	45.11	24.75	36.03
	Llama-3.1-8B-Instruct	23.01	30.25	6.51	9.76	5.20	32.11	58.07	73.56	17.03	38.80	21.96	36.90
	DeepSeek-R1-Distill-Qwen-7B	4.46	8.92	6.19	9.80	0.83	19.00	13.71	35.60	19.07	37.10	8.85	22.08
RankZephyr	Qwen3-8B	45.06	53.74	7.49	12.09	9.09	35.75	**61.35**	**76.38**	31.60	48.83	30.92	45.36
	gemma-3-12b-it	41.57	48.90	7.17	11.23	3.02	15.89	**61.65**	**76.95**	25.83	44.45	27.85	39.48
	Llama-3.1-8B-Instruct	34.58	43.71	6.51	9.30	5.06	33.26	58.00	73.98	14.53	36.74	23.74	39.40
	DeepSeek-R1-Distill-Qwen-7B	7.71	14.45	6.19	9.24	0.58	18.48	13.80	36.46	17.43	35.31	9.14	22.79
Set Selection													
SetSelection-Vanilla	Qwen3-8B	45.54	53.87	**8.14**	12.76	8.89	35.05	61.03	75.76	35.77	51.58	31.87	45.80
	gemma-3-12b-it	46.63	53.52	7.49	12.09	4.52	18.28	61.47	76.82	29.40	46.97	29.90	41.54
	Llama-3.1-8B-Instruct	39.76	48.77	6.84	10.46	4.68	35.68	59.96	75.34	33.10	56.69	28.87	45.39
	DeepSeek-R1-Distill-Qwen-7B	15.90	25.06	6.84	11.18	**8.14**	34.62	**36.04**	55.23	**33.27**	50.31	20.04	35.28
SETR	Qwen3-8B	29.64	33.71	5.86	9.10	**9.59**	**36.89**	53.95	67.42	32.83	49.58	26.37	39.34
	gemma-3-12b-it	28.67	32.41	5.86	9.55	4.62	19.55	54.14	68.36	26.67	44.97	23.99	34.97
	Llama-3.1-8B-Instruct	25.06	30.33	5.54	8.64	5.08	35.43	53.49	67.51	23.47	46.16	22.53	37.61
	DeepSeek-R1-Distill-Qwen-7B	11.57	17.83	4.56	8.23	6.19	31.70	31.16	48.83	27.07	43.65	16.11	30.05
Rank4Gen	Qwen3-8B	**46.51**	**55.91**	**8.14**	**14.12**	9.28	36.23	59.80	75.26	**35.83**	**52.84**	**31.91**	**46.87**
	gemma-3-12b-it	**46.87**	**54.88**	**8.47**	**14.01**	**5.43**	**20.64**	61.35	76.31	**31.70**	**50.64**	**30.76**	**43.30**
	Llama-3.1-8B-Instruct	**42.05**	**52.86**	**7.49**	**12.05**	**5.74**	**37.66**	**60.10**	**76.09**	**33.47**	**58.21**	**29.77**	**47.37**
	DeepSeek-R1-Distill-Qwen-7B	**18.43**	**31.14**	**8.14**	**13.06**	7.29	**35.62**	34.91	**55.80**	33.17	**51.16**	**20.39**	**37.36**

Table 2: Main results on five RAG benchmarks using different rankers and representative generators. Performance is reported in Exact Match (EM) and token-level F1. τ denotes that the top-10 retrieved documents are used. The best and second-best results are highlighted in **bold** and underline for each generator, respectively.

explicitly identify different downstream generators within the ranker, we condition preference supervision on generator-related information.

Concretely, for each downstream generator, we associate a unique generator identifier and a textual generator description that summarizes its profile and characteristics. These descriptions are constructed by collecting publicly available information about each generator and using an LLM with retrieval augmentation to synthesize concise generator profiles. During PRISM construction, we apply response quality alignment separately for each generator while incorporating the corresponding generator information. In total, PRISM covers seven open-source LLM generators, whose detailed configurations are listed in Table B1.

3.3 Rank4Gen Training

Rank4Gen is trained using a two-stage procedure that combines relevance-based supervised fine-tuning with preference-based optimization, providing a stable initialization and effective alignment with generator-specific ranking preferences. Meanwhile, motivated by intuitive considerations on document understanding for ranking, we introduce two complementary reasoning modes, `/index` and `/snapshot`, as shown in the right part of Figure 2. In the `/index` mode, the model directly outputs document IDs, while in our setting, the `/snapshot` mode additionally outputs the first 100 characters of each document after the ID. Rank4Gen supports both modes during training and inference, helping strengthen ID–content alignment for document set selection and ranking.

Relevance SFT and Preference Cold Start. We first perform supervised fine-tuning (SFT) to equip the ranker with relevance-aware document selection and structured output capabilities, stabilizing subsequent preference optimization. In this stage, Rank4Gen is trained primarily under a default setting without conditioning on a specific generator. Given a query and its candidate documents, the model learns to select and output an ordered document subset consisting of all positive documents.

To initialize generator-specific behaviors, we additionally introduce a small amount of preference-aware data as a cold start. For each downstream

Training Setting	Infer. Mode	Generator	BrowseComp+ EM	BrowseComp+ F1	KG-MHQA EM	KG-MHQA F1	ChronoQA EM	ChronoQA F1	SimpleQA EM	SimpleQA F1	CN-SimpleQA EM	CN-SimpleQA F1	Avg. EM	Avg. F1
Qwen3-8B *Zero-shot*	/index	Qwen3-8B	33.98	41.05	7.17	12.01	8.29	34.80	**61.78**	**76.60**	32.30	49.08	28.70	42.71
		gemma-3-12b-it	21.93	26.03	7.49	11.91	3.07	15.17	**62.00**	**77.49**	25.17	42.70	23.93	34.66
		Llama-3.1-8B-Instruct	17.71	23.70	5.86	9.05	5.24	32.81	58.37	73.94	15.07	36.65	20.45	35.23
		DeepSeek-R1-Distill-Qwen-7B	4.58	9.45	4.56	8.22	2.30	21.30	19.19	39.84	17.77	34.59	9.68	22.68
SFT+DPO	/snapshot	Qwen3-8B	**46.51**	**55.91**	**8.14**	**14.12**	**9.28**	**36.23**	59.80	75.26	**35.83**	**52.84**	**31.91**	**46.87**
		gemma-3-12b-it	**46.87**	**54.88**	8.47	14.01	**5.43**	20.64	61.35	76.31	**31.70**	**50.64**	**30.76**	**43.30**
		Llama-3.1-8B-Instruct	**42.05**	**52.86**	**7.49**	**12.05**	5.74	37.66	**60.10**	**76.09**	**33.47**	**58.21**	**29.77**	**47.37**
		DeepSeek-R1-Distill-Qwen-7B	18.43	**31.14**	**8.14**	**13.06**	**7.29**	**35.62**	**34.91**	**55.80**	33.17	51.16	**20.39**	**37.36**
	/index	Qwen3-8B	45.78	54.51	**8.14**	14.04	**9.28**	36.15	57.51	72.73	33.40	50.77	30.82	45.64
		gemma-3-12b-it	44.10	52.45	**8.79**	**14.19**	5.16	**22.20**	59.04	74.14	**31.70**	50.50	29.76	42.70
		Llama-3.1-8B-Instruct	38.67	48.83	6.84	11.22	**5.78**	**37.67**	58.88	74.61	26.87	53.13	27.41	45.09
		DeepSeek-R1-Distill-Qwen-7B	**18.67**	29.55	7.17	12.46	4.68	32.40	33.22	53.39	29.90	47.81	18.73	35.12
SFT	/snapshot	Qwen3-8B	44.34	52.91	**8.14**	13.23	8.87	35.55	56.93	71.89	34.10	50.52	30.48	44.82
		gemma-3-12b-it	45.42	53.70	7.49	13.08	4.37	18.30	57.63	72.99	29.77	48.40	28.94	41.29
		Llama-3.1-8B-Instruct	39.16	49.26	7.17	11.83	4.37	34.89	57.88	73.65	31.50	56.09	28.02	45.14
		DeepSeek-R1-Distill-Qwen-7B	17.23	28.81	6.84	12.04	7.85	34.30	33.47	53.37	**33.67**	**51.23**	19.81	35.95
SFT+DPO *Index-only*	/index	Qwen3-8B	46.02	54.47	7.82	13.33	8.83	34.60	59.45	74.56	34.07	50.83	31.24	45.56
		gemma-3-12b-it	44.22	51.97	7.82	12.96	4.72	20.36	61.44	76.35	31.60	50.45	29.96	42.42
		Llama-3.1-8B-Instruct	39.76	50.00	7.17	11.72	4.89	34.61	59.75	76.03	26.87	51.98	27.69	44.87
		DeepSeek-R1-Distill-Qwen-7B	14.10	26.38	7.17	11.76	4.85	31.42	33.77	54.22	30.20	47.20	18.02	34.20

Table 3: Ablation study results with different training settings and inference modes for Rank4Gen. The best and second-best results are highlighted in **bold** and underline for each generator, respectively.

generator, we include the best-performing ordered subsets identified from full permutations of positive documents that differ from the default ordering. As this generator-specific data is limited in scale, it serves as a lightweight cold start rather than full preference optimization. Together, relevance SFT and preference cold start provide a stable initialization for downstream preference learning.

DPO Training. After initialization, we further optimize Rank4Gen using Direct Preference Optimization (DPO). In this stage, we leverage the preference supervision provided by PRISM, consisting of preferred ordered sets $\mathcal{S}^+$ and dispreferred ordered sets $\mathcal{S}^-$ constructed under specific generators. For each preference pair, the ranker is trained to favor $\mathcal{S}^+$ over $\mathcal{S}^-$ when conditioned on the corresponding generator. This preference-based optimization directly aligns Rank4Gen with generator-specific document set preferences and downstream generation quality.

4 Experiment Setup

4.1 Datasets and Metrics

Training Dataset. We train Rank4Gen on PRISM. Due to computational constraints, we sample 12,994 queries to construct **PRISM_13K**. Detailed dataset statistics are reported in Table 1.

Evaluation Datasets. We evaluate Rank4Gen on two groups of RAG benchmarks. The first group focuses on complex reasoning, including multi-hop and temporal question answering, and consists of BrowseComp-Plus (Chen et al., 2025a), KG-MHQA (Wang et al., 2025), and ChronoQA (Chen et al., 2025b). The second group targets factuality evaluation and includes SimpleQA (Wei et al., 2024) and ChineseSimpleQA (He et al., 2025), for which we additionally augment candidate documents to enable RAG-based inference.

Metrics. Following prior work, we evaluate downstream response quality using Exact Match (EM) and F1 scores, computed between the generated responses and the ground-truth answers.

4.2 Baseline

We evaluate Rank4Gen against a broad set of baselines across seven downstream generators, including LLM-based ranking methods under Pointwise, Listwise, and Set Selection paradigms (see Appendix B.2 for details). We also include distillation-based ranking methods, including RankZephyr (Pradeep et al., 2023) and SETR (Lee et al., 2025), using Gemini-2.5 Flash Lite[1] as the teacher model. For fair comparison, all methods are trained on the same PRISM_13K training set.

4.3 Implementation Details

We use Qwen3-8B as the backbone. Baselines use two epochs of SFT, while Rank4Gen uses one epoch of SFT followed by one epoch of DPO. All models are trained using ms-swift (Zhao et al., 2025), with a learning rate of 1×10^{-5} and a warm-up ratio of 0.05, on 8 NVIDIA H20 GPUs. During inference, we adopt a dynamic temperature strategy with up to three retries, using temperatures of 0.0, 0.7, and 1.0, respectively.

[1] https://docs.cloud.google.com/vertex-ai/generative-ai/docs/models/gemini/2-5-flash-lite

5 Experiment Analysis

5.1 Main Results

Table 2 reports the main results on five RAG benchmarks across four representative downstream generators, with full results provided in Table B2. Overall, Rank4Gen improves downstream response quality on most datasets and generators.

Pointwise and listwise ranking methods achieve competitive performance in certain settings, but their performance is less stable across benchmarks and downstream generators. This instability is likely due to their reliance on top-k selection from a global ranking, which limits their ability to adapt document subsets to different queries and generator behaviors. In contrast, set selection methods are generally better suited for RAG, as training on all-positive document sets yields contexts with higher concentrations of positive evidence, which is beneficial for answer generation.

Distillation-based methods show mixed performance across datasets and generators. Their effectiveness appears sensitive to the capability and preference biases of the teacher model, which may not generalize consistently across different RAG benchmarks or generator configurations.

Compared to these baselines, the full Rank4Gen model, trained with relevance SFT and preference DPO and leveraging both `/index` and `/snapshot` reasoning modes, achieves more consistent improvements across downstream generators. These results highlight the benefits of jointly optimizing ranking with respect to response quality and generator-specific preferences.

5.2 Ablation

Table 3 presents ablation results for key components of Rank4Gen. By comparing variants trained with different configurations, we analyze the effects of relevance-based initialization, preference optimization, and reasoning modes.

While the backbone model exhibits strong instruction-following ability and can handle simple factual queries in a zero-shot manner, its document selection does not generalize well across generators. Through training, Relevance SFT provides a strong foundation for relevance-aware document selection, while its performance still varies noticeably across downstream generators, reflecting limited generalization under generator-agnostic objectives. Introducing DPO improves overall performance and leads to more stable behavior across generators, highlighting the benefit of answer-quality-based preference optimization.

When combined with the `/index` mode, the `/snapshot` reasoning mode provides additional gains in some settings, though its effect is not consistently observed across all datasets.

Generator	Setting	BrowseComp+		ChronoQA	
		EM	F1	EM	F1
Ministral-3-14B	**w/o RAG**	0.24	2.26	0.65	2.17
	Rank4Gen *default*	35.78	49.04	0.85	21.78
	Rank4Gen	36.27	50.89	0.93	22.23
DeepSeek-V3.2	**w/o RAG**	2.41	5.41	1.30	3.79
	Rank4Gen *default*	52.53	63.04	7.60	33.37
	Rank4Gen	53.73	63.04	7.97	33.56

Table 4: Generalization results on BrowseComp+ and ChronoQA with two OOD generators (EM / F1).

5.3 Generalization

We evaluate the generalization ability of Rank4Gen on unseen generators that are not included during PRISM construction. Specifically, we consider an additional open-source LLM and a large-scale proprietary LLM, with details provided in Table B1.

As shown in Table 4, Rank4Gen maintains strong RAG performance on unseen generators under the default mode, where no generator information is specified. This demonstrates that Rank4Gen can generalize beyond the generators observed during training. When generator identifiers and descriptions are provided, performance further improves, indicating that Rank4Gen can effectively leverage generator information to better adapt document selection and ranking to new generators.

6 Conclusion

In this work, we study ranking for generators in RAG and propose **Rank4Gen**, a generator-aware ranker that selects and orders document subsets by modeling response-quality-based and generator-specific preferences. To support this formulation, we construct **PRISM**, a bilingual preference-aligned dataset covering 141k queries for generator-conditioned document set selection and ranking. Rank4Gen is trained on **PRISM_13K** and evaluated on five challenging RAG benchmarks. Experimental results show that Rank4Gen consistently improves downstream response quality and robustness across generators. Overall, Rank4Gen provides a principled approach to aligning document ranking with downstream generation objectives and highlights the importance of modeling generator preferences in RAG systems.

Limitations

This work has a few limitations, primarily related to data scale, preference optimization behavior, and data requirements.

Due to computational constraints, Rank4Gen is trained on a sampled subset of PRISM, namely **PRISM_13K**. While our experiments show consistent improvements under this setting, training on the full PRISM dataset may further enhance model capacity and robustness. We plan to release the full PRISM dataset to support future research.

From the evaluation results after DPO training, we observe that downstream F1 scores tend to improve, while Exact Match (EM) may decrease for some generators, as shown in Table 2. This suggests a trade-off introduced by preference optimization with mixed positive and negative document subsets, which can encourage the ranker to select larger or more diverse document sets. As a result, generators may produce longer reasoning processes and more diverse responses, improving partial correctness without always achieving exact matching.

In addition, PRISM construction relies on datasets with annotated positive and negative documents. Applying our data construction pipeline to fully unlabeled corpora would require additional large language model–based annotation modules, which may introduce additional annotation cost and limit scalability in such settings.

Finally, we represent each generator using a fixed identifier and a textual description generated through a RAG-based pipeline. While manual correction mitigates potential hallucination artifacts, this static representation may not fully capture the generator's context-dependent evidence preferences. Future work could explore more dynamic generator representations, for example, by allowing the ranker to interact with the generator before ranking to infer its characteristics from contextual signals and adapt document selection accordingly.

Ethical Statement

This work uses only publicly available datasets and models that can be accessed through official sources, and all such resources are properly cited. The construction of PRISM does not involve human annotation, and all preference signals are derived automatically through model-based evaluation. During the course of this project, AI assistants were used solely to support auxiliary tasks such as coding assistance and text polishing. These tools were used in a responsible manner and did not replace human judgment in research design, experimentation, or result interpretation.

References

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*.

Zijian Chen, Xueguang Ma, Shengyao Zhuang, Ping Nie, Kai Zou, Andrew Liu, Joshua Green, Kshama Patel, Ruoxi Meng, Mingyi Su, and 1 others. 2025a. Browsecomp-plus: A more fair and transparent evaluation benchmark of deep-research agent. *arXiv preprint arXiv:2508.06600*.

Ziyang Chen, Erxue Min, Xiang Zhao, Yunxin Li, Xin Jia, Jinzhi Liao, Jichao Li, Shuaiqiang Wang, Baotian Hu, and Dawei Yin. 2025b. a question answering dataset for temporal-sensitive retrieval-augmented generation. *Scientific Data*, 12(1):1855.

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

Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, Zhuoran Lin, Dekai Sun, Zhicheng Zheng, Wenbo Su, and Bo Zheng. 2025. Chinese SimpleQA: A Chinese factuality evaluation for large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 19182–19208, Vienna, Austria. Association for Computational Linguistics.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and 1 others. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems*, 43(2):1–55.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th conference of the european chapter of the association for computational linguistics: main volume*, pages 874–880.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7029–7043.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

Philippe Laban, Alexander Richard Fabbri, Caiming Xiong, and Chien-Sheng Wu. 2024. Summary of a haystack: A challenge to long-context llms and rag systems. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9885–9903.

Dahyun Lee, Yongrae Jo, Haeju Park, and Moontae Lee. 2025. Shifting from ranking to set selection for retrieval augmented generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 17606–17619.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

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. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Qi Liu, Bo Wang, Nan Wang, and Jiaxin Mao. 2025a. Leveraging passage embeddings for efficient listwise reranking with large language models. In *Proceedings of the ACM on Web Conference 2025*, pages 4274–4283.

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

Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, and Enhong Chen. 2025. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *ACM Transactions on Information Systems*, 43(2):1–32.

Siyuan Meng, Junming Liu, Yirong Chen, Song Mao, Pinlong Cai, Guohang Yan, Botian Shi, and Ding Wang. 2025. From ranking to selection: A simple but efficient dynamic passage selector for retrieval augmented generation. *arXiv preprint arXiv:2508.09497*.

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.

Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*.

Ronak Pradeep and 1 others. 2023. Rankzephyr: Effective and robust zero-shot listwise reranking. *arXiv preprint arXiv:2312.02724*.

Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, and 1 others. 2024. Large language models are effective text rankers with pairwise ranking prompting. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518.

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. *Advances in neural information processing systems*, 36:53728–53741.

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. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 8642–8652.

Zhengliang Shi, Lingyong Yan, Weiwei Sun, Yue Feng, Pengjie Ren, Xinyu Ma, Shuaiqiang Wang, Dawei Yin, Maarten de Rijke, and Zhaochun Ren. 2025. Direct retrieval-augmented optimization: Synergizing knowledge selection and language models. *arXiv preprint arXiv:2505.03075*.

Duolin Sun, Meixiu Long, Dan Yang, Yihan Jiao, Zhehao Tan, Jie Feng, Junjie Wang, Yue Shen, Peng Wei, Jian Wang, and 1 others. 2025a. Grouprank: A groupwise reranking paradigm driven by reinforcement learning. *arXiv preprint arXiv:2511.11653*.

Jiashuo Sun, Xianrui Zhong, Sizhe Zhou, and Jiawei Han. 2025b. Dynamicrag: Leveraging outputs of large language model as feedback for dynamic reranking in retrieval-augmented generation. *arXiv preprint arXiv:2505.07233*.

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*, pages 14918–14937.

Yubao Tang, Ruqing Zhang, Jiafeng Guo, Maarten De Rijke, Wei Chen, and Xueqi Cheng. 2024. List-wise generative retrieval models via a sequential learning process. *ACM Transactions on Information Systems*, 42(5):1–31.

Fangzheng Tian, Debasis Ganguly, and Craig Macdonald. 2025. Is relevance propagated from retriever to generator in rag? In *European Conference on Information Retrieval*, pages 32–48. Springer.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022. ♫ MuSiQue: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554.

Nan Wang, Yongqi Fan, ZongYu Wang, Xuezhi Cao, Xinyan He, Haiyun Jiang, Tong Ruan, Jingping Liu, and 1 others. 2025. Kg-o1: Enhancing multi-hop question answering in large language models via knowledge graph integration. *arXiv preprint arXiv:2508.15790*.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. 2024. Measuring short-form factuality in large language models. *arXiv preprint arXiv:2411.04368*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. 2024. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, 37:121156–121184.

Hengran Zhang, Keping Bi, Jiafeng Guo, Jiaming Zhang, Shuaiqiang Wang, Dawei Yin, and Xueqi Cheng. 2025a. Distilling a small utility-based passage selector to enhance retrieval-augmented generation. In *Proceedings of the 2025 Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*, pages 22–30.

Le Zhang, Bo Wang, Xipeng Qiu, Siva Reddy, and Aishwarya Agrawal. 2025b. Rearank: Reasoning re-ranking agent via reinforcement learning. *arXiv preprint arXiv:2505.20046*.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, and 1 others. 2025. Swift: a scalable lightweight infrastructure for fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 29733–29735.

A Supplementary material for benchmark access approach

We provide an anonymous link to a GitHub repository containing the PRISM dataset, training and evaluation codes, and Rank4Gen models. Meanwhile, we have added an open-source license to the project.

URL: https://anonymous.4open.science/r/Rank4Gen-2267

B Supplementary Materials for Dataset and Experiment

B.1 Generator Configuration

Table B1 lists the generators considered in this work, including those within PRISM, which are used during PRISM construction and Rank4Gen training, and those without PRISM, which are excluded and only used for evaluating generalization to unseen generators. Figures B2 and B3 present example generator descriptions in English and Chinese, respectively.

Generators within PRISM
Qwen3-8B
gpt-oss-20b
gemma-3-12b-it
Qwen3-8B-thinking
Qwen2.5-7B-Instruct
Llama-3.1-8B-Instruct
DeepSeek-R1-Distill-Qwen-7B
Generators without PRISM
Ministral-3-14B-Instruct-2512
DeepSeek-V3.2

Table B1: Generators within PRISM and without PRISM.

B.2 Baseline Description

We provide detailed training setups on PRISM_13K for three basic ranking paradigms: pointwise, listwise, and set selection.

Pointwise Ranking. The pointwise paradigm treats document ranking as a binary relevance classification problem. Each candidate document is independently scored with respect to the query using a binary relevance label (e.g., yes/no), where documents annotated as positive in PRISM_13K are treated as relevant. The ranker is trained to predict the relevance of each document individually, and documents are ranked by their predicted scores.

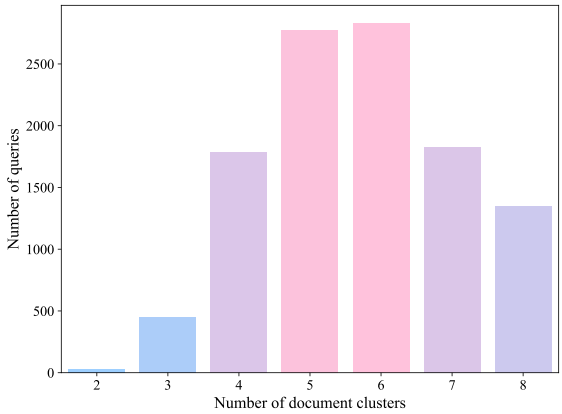

Figure B1: Distribution of the number of clusters derived from the fused three-dimensional coordinate representation.

Listwise Ranking. The listwise paradigm takes the full set of candidate documents as input and directly outputs an ordering over all candidates. During training, the ranker is optimized to promote positive documents to higher positions in the output ranking, while documents are otherwise ordered according to their original retrieval order when no preference is specified.

Set Selection. The set selection paradigm focuses on selecting and ordering only the positive documents. Given the full candidate document set as input, the ranker outputs an ordered subset consisting of all positive documents, with their relative order learned during training and their original retrieval order preserved when ties occur.

B.3 Three-Dimensional Fusion Clustering

For each query, we partition its associated passages along three independent and complementary dimensions: semantics, lexical form, and length. The semantic dimension captures similarity in the embedding space, the lexical dimension reflects similarity at the word and phrase level, and the length dimension distinguishes passages with different textual lengths. In each dimension, passages are clustered into two groups ($k = 2$).

Each passage is assigned a discrete label along each dimension. The combination of the three labels forms a three-dimensional discrete coordinate, which is used to bucket passages. Each bucket thus corresponds to a group of passages that are relatively consistent in semantic content, lexical expression, and length, while being clearly distinct from passages in other buckets.

These buckets are treated as clusters in subsequent sampling and composition. This design allows us to preserve relevance while explicitly encouraging diversity in expression and structural characteristics.

B.4 Prompt Templates

This section presents the prompts used in different stages of our method, as shown in Figures B4 to B9. Figure B4 illustrates the prompt template used for downstream question answering in the RAG setting. Figures B5 and B6 present the system prompts used for Rank4Gen training in English and Chinese, respectively, while Figure B7 shows the corresponding user prompt. Figures B8 and B9 show the system and user prompts used for LLM-as-a-Judge during PRISM construction.

C Detailed Dataset Description

Table 1 presents a high-level overview of the datasets used for training and evaluation in this work. For clarity and space efficiency, the main table reports only the dataset names, sample sizes, and languages, while additional details regarding data sources, preprocessing, and document augmentation are summarized here.

Training datasets. The PRISM_13K collection consists of multiple widely used open-source question answering benchmarks, including HotpotQA, 2WikiMultiHopQA, MUSIQUE, MS MARCO, and CRUD-RAG. These datasets originate from heterogeneous sources such as Wikipedia, web search results, and news articles. We apply the dataset- specific filtering and subsampling strategies to control data quality and reasoning depth, while preserving the original question–answer pairs and supervision signals.

Evaluation datasets. The evaluation benchmarks cover a diverse set of settings, including web- based question answering (BrowseComp-Plus), knowledge-graph reasoning (KG-MHQA), and news-driven temporal reasoning (ChronoQA), spanning both English and Chinese.

Document augmentation. For datasets that are not originally released with explicit retrieval contexts, we construct additional retrieval-augmented generation (RAG) documents to enable document-grounded evaluation. In particular, for SimpleQA and ChineseSimpleQA, we collect relevant documents using an internal search engine pipeline and provide them as external context during inference. This augmentation does not modify the original questions or answers and introduces no additional supervision, but enables a unified RAG-based evaluation setting across all document-augmented benchmarks in this work.

Ranker	Generator	BrowseComp+ EM	BrowseComp+ F1	KG-MHQA EM	KG-MHQA F1	ChronoQA EM	ChronoQA F1	SimpleQA EM	SimpleQA F1	CN-SimpleQA EM	CN-SimpleQA F1	Avg. EM	Avg. F1
Without RAG													
-	Qwen3-8B	0.12	3.53	0.65	3.36	2.84	24.18	2.80	9.73	14.67	46.27	4.22	17.41
	gpt-oss-20b	0.24	0.83	0.98	2.79	1.72	27.78	3.26	11.11	6.87	37.48	2.61	16.00
	gemma-3-12b-it	0.00	2.45	0.65	3.21	1.70	13.66	4.28	11.54	9.07	34.28	3.14	13.03
	Qwen3-8B-thinking	0.00	1.14	0.65	2.96	2.07	22.21	3.17	9.99	14.47	44.33	4.07	16.13
	Qwen2.5-7B-Instruct	0.12	2.54	0.65	2.82	0.56	13.43	1.04	4.34	0.90	16.58	0.65	7.94
	Llama-3.1-8B-Instruct	0.00	1.50	0.65	2.76	2.01	20.77	4.05	9.45	6.30	30.61	2.60	13.02
	DeepSeek-R1-Distill-Qwen-7B	0.12	1.62	0.33	2.48	1.45	21.80	1.16	5.16	2.50	26.69	1.11	11.55
Pointwise$^{\tau}$													
Pointwise-Vanilla	Qwen3-8B	46.02	55.20	6.19	11.40	9.39	36.83	60.56	75.75	30.83	48.09	30.60	45.45
	gpt-oss-20b	33.13	43.39	6.19	8.66	2.26	18.38	41.22	59.89	19.20	43.23	20.40	34.71
	gemma-3-12b-it	39.44	46.64	6.19	11.23	3.15	16.77	60.61	76.30	24.83	42.94	26.84	38.78
	Qwen3-8B-thinking	**49.28**	56.80	7.82	12.57	8.58	34.02	58.88	74.22	27.40	45.13	30.39	44.55
	Qwen2.5-7B-Instruct	12.53	30.37	4.89	9.30	4.21	**30.95**	23.16	42.80	9.90	27.36	10.94	28.16
	Llama-3.1-8B-Instruct	33.38	42.47	5.86	9.08	5.39	33.52	58.34	73.66	15.80	36.60	23.75	39.07
	DeepSeek-R1-Distill-Qwen-7B	8.31	15.47	5.54	9.30	0.81	21.14	14.24	35.55	17.07	34.10	9.19	23.11
Listwise$^{\tau}$													
Listwise-Vanilla	Qwen3-8B	29.88	37.77	7.82	11.52	8.25	34.13	59.69	74.79	32.60	49.40	27.65	41.52
	gpt-oss-20b	23.98	31.95	7.82	10.98	4.16	26.55	49.75	67.53	28.07	56.00	22.76	38.60
	gemma-3-12b-it	25.42	31.08	7.82	12.20	3.17	16.12	60.33	75.65	27.03	45.11	24.75	36.03
	Qwen3-8B-thinking	31.57	38.78	8.14	12.22	7.50	33.01	59.92	75.34	30.87	47.47	27.60	41.36
	Qwen2.5-7B-Instruct	9.52	21.29	5.54	9.37	4.33	29.41	23.23	42.51	10.10	27.92	10.54	26.10
	Llama-3.1-8B-Instruct	23.01	30.25	6.51	9.76	5.20	32.11	58.07	73.56	17.03	38.80	21.96	36.90
	DeepSeek-R1-Distill-Qwen-7B	4.46	8.92	6.19	9.80	0.83	19.00	13.71	35.60	19.07	37.10	8.85	22.08
RankZephyr	Qwen3-8B	45.06	53.74	7.49	12.09	9.09	35.75	**61.35**	**76.38**	31.60	48.83	30.92	45.36
	gpt-oss-20b	32.17	43.00	6.84	9.07	1.99	17.19	41.31	60.11	21.23	46.03	20.71	35.08
	gemma-3-12b-it	41.57	48.90	7.17	11.23	3.02	15.89	**61.65**	**76.95**	25.83	44.45	27.85	39.48
	Qwen3-8B-thinking	43.61	51.33	8.14	11.61	8.14	33.25	59.27	74.64	28.03	45.88	29.44	43.34
	Qwen2.5-7B-Instruct	16.99	31.87	5.86	9.40	4.10	30.57	23.35	43.12	8.83	26.93	11.83	28.38
	Llama-3.1-8B-Instruct	34.58	43.71	6.51	9.30	5.06	33.26	58.00	73.98	14.53	36.74	23.74	39.40
	DeepSeek-R1-Distill-Qwen-7B	7.71	14.45	6.19	9.24	0.58	18.48	13.80	36.46	17.43	35.31	9.14	22.79
Set Selection													
SetSelection-Vanilla	Qwen3-8B	45.54	53.87	**8.14**	12.76	8.89	35.05	61.03	75.76	35.77	51.58	31.87	45.80
	gpt-oss-20b	**42.41**	**51.61**	7.82	11.87	**9.14**	**37.57**	**54.37**	**72.64**	**39.10**	**62.83**	**30.57**	**47.30**
	gemma-3-12b-it	46.63	53.52	7.49	12.09	4.52	18.28	61.47	76.82	29.40	46.97	29.90	41.54
	Qwen3-8B-thinking	48.80	55.87	8.14	12.52	8.76	**34.71**	**61.21**	**76.13**	**34.40**	50.19	**32.26**	45.88
	Qwen2.5-7B-Instruct	**20.48**	34.91	**6.84**	10.83	4.50	28.82	29.43	48.13	10.10	27.47	14.27	30.03
	Llama-3.1-8B-Instruct	39.76	48.77	6.84	10.46	4.68	35.68	59.96	75.34	33.10	56.69	28.87	45.39
	DeepSeek-R1-Distill-Qwen-7B	15.90	25.06	6.84	11.18	**8.14**	34.62	**36.04**	55.23	**33.27**	50.31	20.04	35.28
SETR	Qwen3-8B	29.64	33.71	5.86	9.10	**9.59**	**36.89**	53.95	67.42	32.83	49.58	26.37	39.34
	gpt-oss-20b	20.60	26.38	5.21	7.91	3.11	22.03	38.03	54.57	21.73	46.18	17.74	31.41
	gemma-3-12b-it	28.67	32.41	5.86	9.55	4.62	19.55	54.14	68.36	26.67	44.97	23.99	34.97
	Qwen3-8B-thinking	28.92	33.15	5.86	9.17	8.41	34.51	52.75	66.29	29.80	47.22	25.15	38.07
	Qwen2.5-7B-Instruct	13.98	23.17	4.56	8.29	**4.77**	30.80	25.68	42.85	10.10	27.68	11.82	26.56
	Llama-3.1-8B-Instruct	25.06	30.33	5.54	8.64	5.08	35.43	53.49	67.51	23.47	46.16	22.53	37.61
	DeepSeek-R1-Distill-Qwen-7B	11.57	17.83	4.56	8.23	6.19	31.70	31.16	48.83	27.07	43.65	16.11	30.05
Rank4Gen	Qwen3-8B	**46.51**	**55.91**	**8.14**	**14.12**	9.28	36.23	59.80	75.26	**35.83**	**52.84**	**31.91**	**46.87**
	gpt-oss-20b	38.31	50.42	**8.47**	**12.19**	5.05	28.69	48.96	66.79	27.27	53.44	25.61	42.31
	gemma-3-12b-it	**46.87**	**54.88**	**8.47**	**14.01**	**5.43**	**20.64**	61.35	76.31	**31.70**	**50.64**	**30.76**	**43.30**
	Qwen3-8B-thinking	47.71	**57.66**	**9.12**	**14.21**	**8.83**	34.54	57.77	73.49	32.93	**50.61**	31.27	**46.10**
	Qwen2.5-7B-Instruct	19.40	**36.18**	**6.84**	**11.76**	4.52	28.83	**32.85**	**50.78**	**11.23**	**28.80**	**14.97**	**31.27**
	Llama-3.1-8B-Instruct	**42.05**	**52.86**	**7.49**	**12.05**	**5.74**	**37.66**	**60.10**	**76.09**	**33.47**	**58.21**	**29.77**	**47.37**
	DeepSeek-R1-Distill-Qwen-7B	**18.43**	**31.14**	**8.14**	**13.06**	7.29	**35.62**	34.91	**55.80**	33.17	**51.16**	**20.39**	**37.36**

Table B2: Main results on five RAG benchmarks using different rankers and generators. Performance is reported in Exact Match (EM) and token-level F1. τ denotes that the top-10 retrieved documents are used. The best and second-best results are highlighted in **bold** and underline for each generator, respectively.

Qwen3-8B:
Qwen3-8B is an open-source large language model developed by Alibaba's Qwen team. It possesses approximately 8 billion parameters and supports a context length of up to 32,768 tokens. The model demonstrates outstanding performance in domains such as multilingual translation, logical reasoning, mathematical computation, and code generation, with a particular strength in hybrid reasoning and efficient task processing. Its unique characteristic is the ability to seamlessly switch between a "Thinking Mode" and a "Non-Thinking Mode". In Non-Thinking Mode, the model directly outputs the answer, offering faster and more efficient responses, which is ideal for rapid, everyday conversations or simple queries. This approach enhances the user experience by avoiding unnecessary step-by-step thinking. Conversely, when in-depth analysis is needed, the model can be switched to Thinking Mode to engage in step-by-step reasoning, ensuring the accurate resolution of complex problems.

gpt-oss-20b:
gpt-oss-20B is an open-source language model based on the Mixture-of-Experts (MoE) architecture, released by OpenAI in 2025. It features approximately 21 billion total parameters, with about 3.6 billion being active parameters. The model supports a context window of up to 128k tokens and possesses capabilities such as Chain-of-Thought (CoT) reasoning, adjustable reasoning effort, function calling, and tool use. Furthermore, its performance in instruction following and structured output has been reinforced through Supervised Fine-Tuning (SFT) and Reinforcement Learning. Benefiting from the MoE architecture's characteristic of selectively activating only a subset of experts, it significantly reduces computational and video memory requirements while maintaining reasoning performance. This allows for local execution on devices with approximately 16GB of memory, making it suitable for edge deployment and resource-constrained scenarios. In evaluations, the model demonstrates robust performance in reasoning and coding tasks, although there remains room for improvement in its multilingual capabilities. Overall, it is positioned as an efficient, flexible, and locally deployable open-source large language model.

gemma-3-12b-it:
Gemma-3-12b-it is a lightweight, instruction-tuned multimodal language model developed by Google DeepMind, featuring approximately 12 billion parameters. It is specifically designed for efficient deployment on single GPUs, TPUs, or resource-constrained devices such as laptops and smartphones. The model supports a context window of up to 128k tokens, enabling it to effectively handle long inputs, and boasts multilingual capabilities spanning over 140 languages. As a multimodal model, it utilizes a custom SigLIP vision encoder to process text and image inputs and generate text outputs; it is suitable for tasks such as question answering, summarization, reasoning, and image understanding, excelling in benchmarks relative to models of comparable scale. Gemma-3-12b-it represents an advancement in the Gemma family, emphasizing optimized performance, ease of deployment, and enhanced safety features through the introduction of vision-language integration and expanded size options.

Qwen3-8B-thinking:
The Thinking Mode of Qwen3-8B represents a core innovation, significantly enhancing the capability to solve complex tasks by simulating the human deep thinking process. In this mode, the model generates detailed intermediate reasoning steps for intricate problems such as mathematical reasoning and code generation; these thinking processes are encapsulated within specific <think> tags, rendering the logical chain clearly visible. Users can actively trigger this mode by inputting the /think command or by setting the enable_thinking=True parameter. This design not only yields a substantial boost in accuracy across professional mathematics and coding benchmarks but also enhances the credibility and interpretability of the model's outputs through transparent reasoning processes, thereby achieving an intelligent balance between "Slow Thinking, High Precision" and "Fast Response, High Efficiency."

Qwen2.5-7B-Instruct:
Qwen2.5-7B-Instruct is an instruction-tuned language model within the Qwen2.5 series, featuring approximately 7 billion parameters and built upon the Transformer architecture. It is optimized for general-purpose natural language understanding and generation tasks. By aligning the pre-trained foundation model with high-quality instruction data, it demonstrates enhanced instruction-following capabilities and more stable performance across diverse scenarios, including dialogue, reasoning, code generation, and multilingual processing. The model supports a long context window and leverages efficient tokenization schemes and training strategies to improve inference efficiency and broaden its knowledge coverage. Furthermore, it is compatible with common deployment methods, such as local inference and various quantization formats, striking an excellent balance between performance, inference speed, and resource consumption. This makes it highly suitable for developers to integrate and customize rapidly at the application layer.

Llama-3.1-8B-Instruct:
Llama-3.1-8B-Instruct is an open-source, instruction-tuned language model with approximately 8 billion parameters, designed for general-purpose dialogue, task execution, and text generation scenarios. Building upon the language understanding and generation capabilities of the base model, it is optimized using large-scale instruction data, enabling it to exhibit greater stability in adhering to user intent, providing structured responses, articulating reasoning, and generalizing across multiple tasks. The model features high reasoning consistency and lower hallucination rates, achieving effective computational efficiency and deployment flexibility while maintaining a compact parameter scale, making it suitable for diverse usage scenarios such as local deployment, API services, and edge inference.

DeepSeek-R1-Distill-Qwen-7B:
DeepSeek-R1-Distill-Qwen-7B is a lightweight distilled version of DeepSeek-R1, the first-generation reasoning model launched by DeepSeek AI. Utilizing knowledge distillation technology, it condenses the superior capabilities demonstrated by the massive 685-billion-parameter teacher model, DeepSeek-R1, in mathematics, coding, and reasoning tasks into a 7-billion-parameter architecture. Built upon the specialized mathematical reasoning base model, Qwen2.5-Math-7B, this model undergoes supervised fine-tuning using 800,000 high-quality reasoning samples generated by DeepSeek-R1, enabling it to internalize the complex reasoning patterns and Chain-of-Thought (CoT) capabilities of the larger model.

Ministral-3-14B-Instruct-2512:
Ministral-3-14B-Instruct-2512 is the largest model in the Ministral 3 family, offering frontier capabilities and performance comparable to larger models. It is a powerful and efficient multimodal language model with vision capabilities, and this instruct-post-trained version in FP8 precision has been fine-tuned for instruction tasks, making it ideal for chat, instruction following, and assistant-style use cases. The FP8 quantization enables deployment with reduced memory requirements, capable of fitting in 24 GB of VRAM and even less when further quantized. The model supports dozens of major languages including English, French, Spanish, German, Chinese, Japanese, Korean, Arabic and more, and features a large 256 k context window for handling long contexts. It also provides strong adherence to system prompts, native function calling, and structured JSON output, and is licensed under the Apache 2.0 open-source license suitable for both commercial and non-commercial use.

DeepSeek-V3.2:
DeepSeek-V3.2 is an open-source large language model developed by the Chinese AI company DeepSeek and was officially released in December 2025 as the latest version in the DeepSeek-V3 series. The model achieves a strong balance among computational efficiency, reasoning capability, and agent performance through three major technological breakthroughs. First, it introduces the DeepSeek Sparse Attention (DSA) mechanism, an efficient sparse attention method that significantly reduces computational complexity in long-context scenarios while preserving model performance. Second, through a scalable reinforcement learning framework and large-scale post-training computation, the model's performance reaches the level of GPT-5. Notably, the high-compute variant DeepSeek-V3.2-Speciale even surpasses GPT-5 and matches Gemini-3.0-Pro in reasoning capability, achieving gold-medal-level performance in competitions such as the International Mathematical Olympiad (IMO) and the International Olympiad in Informatics (IOI) in 2025. Third, DeepSeek developed a large-scale agent task synthesis pipeline that enables reasoning to be directly integrated into tool usage, allowing tools to be invoked in both thinking and non-thinking modes. This substantially enhances the model's generalization ability and instruction-following robustness in complex interactive environments. DeepSeek-V3.2 is well suited for advanced reasoning, agent-based AI applications, tool calling, as well as mathematics and programming tasks. It has been open-sourced on the Hugging Face platform and is widely available through DeepSeek's App, Web, and API offerings.

default:
The Default model is a versatile Large Language Model(LLM) trained on massive datasets. It possesses cross-task comprehension and generation capabilities, enabling it to reason, learn, and make decisions across diverse complex scenarios.

Figure B2: Examples of descriptions for each generator (en).

Qwen3-8B:
Qwen3-8B是由阿里巴巴Qwen团队开发的开源大语言模型，拥有约80亿参数，支持长达32,768个token的上下文处理，在多语言翻译、逻辑推理、数学计算和编码生成等领域表现出色，尤其擅长混合推理和高效任务处理。它独特之处在于可以无缝切换思考模式和no-thinking模式：在no-thinking模式下，模型直接输出答案，响应更快、更高效，适合日常快速对话或简单查询，避免不必要的逐步思考，从而提升用户体验；而在需要深入分析时，可切换到思考模式，进行步步推理，确保复杂问题的准确解决。

gpt-oss-20b:
gpt-oss-20B 是 OpenAI 于 2025 年开源发布的约 210 亿参数（其中约 36 亿为激活参数）的专家混合架构语言模型，支持最长 128k 上下文，具备链式思维推理、可调节推理强度、函数调用与工具使用等能力，并经过监督微调与强化学习强化了指令遵从与结构化输出表现。得益于 MoE 架构仅激活部分专家的特性，它在保持推理性能的同时显著降低计算与显存需求，可在约 16GB 内存的设备上本地运行，适合边缘部署与资源受限场景。在评测中，该模型在推理与代码任务上表现稳健，多语言能力仍有提升空间，整体定位为高效、灵活、可本地部署的开源大语言模型。

gemma-3-12b-it:
Gemma-3-12b-it是由Google DeepMind开发的轻量级、指令调优的多模态语言模型，拥有约120亿参数，专为在单个GPU、TPU或资源受限设备（如笔记本电脑和智能手机）上高效部署而设计。 它支持高达128k个token的上下文窗口，能够有效处理长输入，并具备超过140种语言的多语言能力。 作为多模态模型，它使用自定义SigLIP视觉编码器处理文本和图像输入，生成文本输出，适用于问答、摘要、推理和图像理解等任务，并在同类规模模型的基准测试中表现出色。 Gemma-3-12b-it代表了Gemma家族的进步，通过引入视觉-语言整合和更多尺寸选项，强调优化性能、易部署性和增强的安全特性。

Qwen3-8B-thinking:
Qwen3-8B的Thinking模式是其核心创新，它通过模拟人类的深度思考过程来显著提升复杂任务的解决能力。在该模式下，模型会为数学推理、代码生成等复杂问题生成详细的中间推理步骤，这些思考过程被封装在特定的<think>标签内，使得逻辑链条清晰可见。用户可通过输入/think指令或设置enable_thinking=True参数来主动触发此模式。这种设计不仅大幅提升了在专业数学和代码评测中的准确率，更通过透明的推理过程增强了模型结果的可信度和可解释性，实现了“慢思考、高精度”与“快响应、高效率”的智能平衡。

Qwen2.5-7B-Instruct:
Qwen/Qwen2.5-7B-Instruct 是一款基于 Transformer 架构的约 70 亿参数指令微调语言模型，属于 Qwen2.5 系列，面向通用自然语言理解与生成任务进行了优化。它在基础大模型的预训练能力之上，通过高质量指令数据进行对齐，使其在对话、推理、代码生成、多语言处理等场景中具备更强的遵循指令能力和更稳定的输出表现。模型通常支持较长上下文输入，并采用高效的分词方案与训练策略来提升推理效率与知识覆盖广度。此外，它兼容常见部署方式（如本地推理和多种量化格式），在性能、推理速度和资源占用之间取得了较好的平衡，适合开发者在应用层快速集成与定制化使用。

Llama-3.1-8B-Instruct:
Llama-3.1-8B-Instruct 是一款约 80 亿参数规模的开源指令微调语言模型，面向通用对话、任务执行与文本生成场景。它在基础模型的语言理解与生成能力之上，通过大规模指令数据进行优化，使其在遵循用户意图、结构化回答、推理表达以及多任务泛化方面表现更稳定。该模型具有较高的推理一致性、较低的幻觉率，并在保持较小参数规模的同时实现了较好的计算效率与部署灵活性，适用于本地部署、API 服务及边缘推理等多种使用场景。

DeepSeek-R1-Distill-Qwen-7B:
DeepSeek-R1-Distill-Qwen-7B 是 DeepSeek AI 推出的第一代推理模型 DeepSeek-R1 的一个轻量化蒸馏版本，它通过知识蒸馏技术，将庞大的 6850 亿参数教师模型 DeepSeek-R1 在数学、代码和推理任务上表现出的卓越能力浓缩至 70 亿参数的架构中。该模型基于专门的数学推理基座模型 Qwen2.5-Math-7B 构建，并利用了从 DeepSeek-R1 生成的 80 万个高质量推理样本进行有监督微调，使其能够内化大模型的复杂推理模式与思维链（Chain-of-Thought）能力。

Ministral-3-14B-Instruct-2512:
Ministral-3-14B-Instruct-2512 是 Mistral AI 推出的大型多模态指令型 AI 模型，是 Ministral 3 系列中参数规模最大的一款，具有约 140 亿参数的语言与视觉融合能力。这个模型在设计时兼顾了高性能和高效率，其整体表现可以与更大规模的同类模型相媲美，同时支持在各种硬件上部署，包括边缘设备和本地服务器等。该模型的权重采用 FP8 量化格式，有助于减少显存占用，使得在单个 24GB GPU 上即可部署运行，进一步量化后对资源的要求更低。Ministral-3-14B-Instruct-2512 是指令微调版本，专门针对聊天、指令执行和助手类工作负载进行了优化，并且具备图像理解能力，可以处理文本和图像输入，实现统一的多模态推理。它支持几十种语言，包括英语、中文、法语、西班牙语、德语、日语、韩语、阿拉伯语等多种主要语言，适合全球应用。模型还拥有大约 256k 的上下文窗口，有利于处理长上下文任务，并原生支持系统提示遵循、函数调用和结构化 JSON 输出等先进特性。此外，该模型采用 Apache-2.0 开源许可，可用于商业和非商业用途。

DeepSeek-V3.2:
DeepSeek-V3.2是由中国AI公司DeepSeek开发的开源大型语言模型，于2025年12月正式发布，是DeepSeek-V3系列的最新版本。该模型在计算效率、推理能力和代理性能方面实现了显著平衡，主要通过三项关键技术突破：首先，引入DeepSeek Sparse Attention（DSA）机制，这是一种高效的稀疏注意力方法，大幅降低了长上下文场景下的计算复杂度，同时保持了模型性能。其次，通过可扩展的强化学习框架和大规模后训练计算，使模型性能达到GPT-5水平。其中，高计算变体DeepSeek-V3.2-Speciale甚至超越GPT-5，并在推理能力上媲美Gemini-3.0-Pro，并在2025年国际数学奥林匹克（IMO）和国际信息学奥林匹克（IOI）等竞赛中取得金牌级表现。第三，开发了大规模代理任务合成管道，支持将思考直接融入工具使用，并在思考模式与非思考模式下均可调用工具，大幅提升了模型在复杂交互环境中的泛化能力和指令遵循鲁棒性。DeepSeek-V3.2适用于高级推理、代理AI应用、工具调用以及数学、编程等领域，已在Hugging Face平台开源，并通过DeepSeek的App、Web和API广泛可用。

default:
Default模型是一个通用大语言模型（LLM），是由海量数据训练、具备跨任务理解与生成能力，能够在多种复杂场景中进行推理、学习和决策的人工智能模型。

Figure B3: Examples of descriptions for each generator (zh).

Downstream QA Answering Prompt (EN)

You are given a question and several reference documents. Please answer the question based on them.

Instructions:
1) Analyze the question first and present a clear line of reasoning.
2) Use the documents to find relevant evidence. Some content may be distracting—read carefully.
3) Follow the output format strictly: write the "Reasoning" section first, then provide the final answer prefixed with "Answer:".

Input
- Question: {query}
- Documents:
{documents}

Output Format
Reasoning: ...
Answer: ...

Downstream QA Answering Prompt (ZH)

你将会收到一个问题和若干参考文档。请根据这些文档回答问题。
指令：
1）先分析问题并呈现清晰的推理过程。
2）使用文档寻找相关证据。部分内容可能具有干扰性——请仔细阅读。
3）严格遵循输出格式：先写“Reasoning”部分，然后给出以“Answer:”开头的最终答案。

输入
- 问题: {query}
- 文档:
{documents}

输出格式
Reasoning: ...
Answer: ...

Figure B4: Prompt used for downstream question answering.

Rank4Gen Training System Prompt (EN)

You are **Rank4Gen**, a **Ranker** designed for retrieval-augmented generation tasks.
Given a **Query (<Query>)** and **Candidate Documents (<Documents>)**, you need to **select and rank** the documents from a set of candidate documents that are most suitable for the downstream generator to answer the query, based on the characteristics and preferences of **Downstream Generator Information**.

When the downstream generator is `default`, it indicates a default mode with no specific preferences. In this case, you should **select and rank** the candidate documents that are **most helpful for the query** and **most directly support answering it**.
Please **strictly follow** the **Instructions (<Instruct>)** below for document selection and ranking.

Downstream Generator Information

The downstream generator you serve is: `{model_name}`
Generator description: `{description}`

Output Mode
1. Index Mode
If the instruction contains **`/index`**, output only the **document index**, one per line, without additional text or explanation.

Example:
[<doc_index_1>]
[<doc_index_2>]
[<doc_index_3>]

2. Snapshot Mode
If the instruction contains **`/snapshot`**, output the selected documents **line by line** using *snapshot format*.
Each line must include:

- **Document index**
- **Preview of the first 100 characters** of the document content

Example:
[<doc_index_1>] <first_100_characters_of_document>...
[<doc_index_2>] <first_100_characters_of_document>...
[<doc_index_3>] <first_100_characters_of_document>...

Figure B5: System prompt used for Rank4Gen training (en).

Rank4Gen Training System Prompt (ZH)

你是**Rank4Gen**，一个检索增强生成任务的**Ranker**。
给定**查询 (<Query>)**与**候选文档 (<Documents>)**，你需要根据**下游生成器信息**的特点和偏好，从候选文档中**筛选并排序**出最适合该生成器回答的文档。

当下游生成器为`default`时，代表无偏好的默认模式，你需要从候选文档中**选择并排序**出**对该查询最有帮助**、**最能直接支持回答**的文档。

请**严格按照**下方的**指令 (<Instruct>)**进行文档选择与排序。

下游生成器信息

你所服务的下游生成器是：`{model_name}`
生成器描述：`{description}`

输出模式

1. Index 模式
如果指令中包含 ** `/index`**，则仅输出 **文档索引**，每行一个，不添加任何解释或额外文本。

示例:
[<doc_index_1>]
[<doc_index_2>]
[<doc_index_3>]

2. Snapshot 模式
如果指令中包含 **`/snapshot`**，请使用 *snapshot 格式* **逐行输出**所选文档。
每行必须包括：

- **文档索引**
- **文档内容前 100 个字符的预览**

示例：
[<doc_index_1>] <first_100_characters_of_document>...
[<doc_index_2>] <first_100_characters_of_document>...
[<doc_index_3>] <first_100_characters_of_document>...

Figure B6: System prompt used for Rank4Gen training (zh).

Rank4Gen Training User Prompt (EN)

<Instruct>: I will provide you with {num} documents, each indicated by a numerical identifier []. Select the documents based on their relevance to the search query "{question}".

<Query>: {question}

<Documents>:
{context}

Select the documents that mostly cover clear and diverse information to answer the query.

Please output the final document selection and sorting results according to the format constraints of the **"Output Mode"**.

<Output>:

Rank4Gen Training User Prompt (ZH)

<Instruct>: 我将向你提供 {num} 个文档，每个文档都有一个数字标识符 []。请根据它们与搜索查询“{question}”的相关性选择段落。

<Query>: {question}

<Documents>:
{context}

请选择那些能够提供清晰且多样信息、最能回答查询的文档。
请根据“输出模式”的格式要求输出最终的文档选择和排序结果。

<Output>:

Figure B7: User prompt used for Rank4Gen training.

LLM-as-a-Judge System Prompt (EN)

You are a careful and objective evaluation expert. You are good at comparing model responses and ranking them.

LLM-as-a-Judge User Prompt (EN)

Task Description
I will give you:
- One standard answer;
- Several candidate responses. Each candidate response includes:
  an id (unique number), a reasoning part, and an answer part.

Your task:
Rank all candidate responses based on how correct and relevant their answers are compared to the standard answer.
Ranking rules (in priority order):
- First, compare the answer part: Check whether the answer is correct, whether it fully covers the key points of the standard answer, and whether it has clear mistakes or is off topic.
- If the answer parts are the same or equal, then compare the reasoning part: Check whether the reasoning is clear and logical, and whether it has clear logic errors or unnecessary content.

Standard Answer
{standard_answer}

Candidate Responses
{candidate_responses}

Output Requirements
Directly output a list showing the ranking of the candidate responses. Responses that appear earlier in the list are more correct and more relevant. Do not output any explanations, descriptions, or any other irrelevant content.
Each item in the list must be an object: {"id": xxx, "correct": true or false}
- correct = true means the answer part is overall correct.
- correct = false means the answer is wrong, incomplete, or clearly does not match the standard answer.
- Once "false" appears, no later item can be marked as "true".
Example format:
[
  {"id": xxx, "correct": true or false},
  {"id": xxx, "correct": true or false},
  {"id": xxx, "correct": true or false}
]

Figure B8: Prompt used for LLM-as-a-Judge (en).

LLM-as-a-Judge System Prompt (ZN)

你是一名严谨、客观的评测专家，擅长对模型回复进行对比分析与排序。

LLM-as-a-Judge User Prompt (ZN)

任务说明
我将提供以下内容：
- 一个标准答案；
- 若干条候选回复，每条候选回复包含：id（唯一编号）、思考部分、答案部分。

你的任务是：
根据候选回复的答案与标准答案之间的正确性与相关性，对所有候选回复进行排序。
排序规则如下（按优先级）：
- 优先比较“答案部分”：答案是否正确，是否完整覆盖标准答案的核心要点，是否存在明显错误或偏离。
- 若答案部分相同或等价，再比较思考部分：推理是否清晰、合理，是否存在明显逻辑错误或冗余。

标准答案
{standard_answer}

候选回复
{candidate_responses}

输出要求
直接输出一个列表（List）表示不同候选回复的排序结果，越靠前的候选回复，表示其答案越正确、越相关，不得输出任何解释、说明或其他无关内容。
列表中每一项是一个对象: {"id": xxx, "correct": true 或 false}
- correct = true 表示该候选回复的“答案部分”整体正确。
- correct = false 表示答案错误、不完整或明显不符合标准答案。
- 出现 false 后不可以再出现 true。
格式示例如下：
```
[
  {"id": xxx, "correct": true 或 false},
  {"id": xxx, "correct": true 或 false},
  {"id": xxx, "correct": true 或 false},
  ...
]
```

Figure B9: Prompt used for LLM-as-a-Judge (zh).