OptiSet: Unified Optimizing Set Selection and Ranking for Retrieval-Augmented Generation

Yi Jiang, Sendong Zhao*, Jianbo Li, Bairui Hu, Yanrui Du, Haochun Wang, Bing Qin

Harbin Institute of Technology, China
{yjiang,sdzhao,jbli,brhu,yrdu,hcwang,qinb}@ir.hit.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) improves generation quality by incorporating evidence retrieved from large external corpora. However, most existing methods rely on statically selecting top-k passages based on individual relevance, which fails to exploit combinatorial gains among passages and often introduces substantial redundancy. To address this limitation, we propose **OptiSet**, a set-centric framework that unifies set selection and set-level ranking for RAG. OptiSet adopts an "Expand-then-Refine" paradigm: it first expands a query into multiple perspectives to enable a diverse candidate pool and then refines the candidate pool via re-selection to form a compact evidence set. We then devise a self-synthesis strategy without strong LLM supervision to derive preference labels from the set conditional utility changes of the generator, thereby identifying complementary and redundant evidence. Finally, we introduce a set-list wise training strategy that jointly optimizes set selection and set-level ranking, enabling the model to favor compact, high-gain evidence sets. Extensive experiments demonstrate that OptiSet improves performance on complex combinatorial problems and makes generation more efficient. Code is public[1].

1 Introduction

Retrieval-Augmented Generation (RAG) addresses the limitations of fixed knowledge in Large Language Models (LLMs) by integrating an external retrieval component that fetches relevant information from a large corpus (Lewis et al., 2020; Zhao et al., 2024; Fan et al., 2024; Li et al., 2025). This approach helps overcome challenges like difficulty in updating knowledge and the generation of hallucinations (Béchard and Ayala, 2024; Gade et al., 2025), leading to significant improvements in various NLP applications.

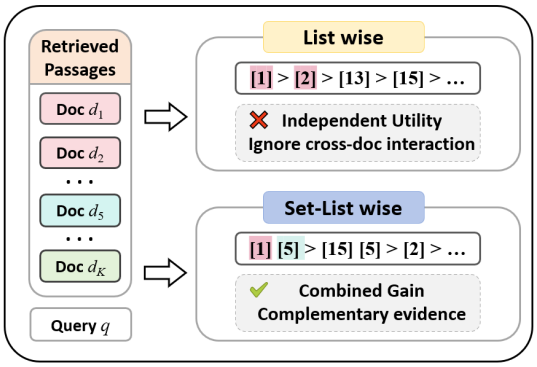

Figure 1: Illustration of the difference in objectives: Set-list wise modeling unifies the modeling of set selection to capture combinatorial gains and reduce redundancy.

Many studies have explored the integration of retrieval and generation. For instance, denoising (Wei et al., 2025) and compression (Xu et al., 2024a) methods aim to reduce redundant information but may compromise information traceability, which is crucial in practical applications. Rerankers (Glass et al., 2022), by re-evaluating passage relevance and selecting the most relevant top-k passages, can seamlessly integrate with and complement various other strategies, and have remained a mainstream research direction.

However, rerankers still face significant challenges. As shown in Fig. 1, while they select the best passages based on relevance, this process often leads to redundancy and fails to capture complementary gains across passages. High-ranking passages may be redundant, wasting context budget and inference cost, whereas moderately relevant passages, when combined, can cover complementary aspects and add value. In short, a static top-k reranker has two issues: (i) it fails to fully explore the space of passage combinations, missing potential benefits; (ii) it often includes near-duplicate evidence, introducing unnecessary redundancy.

To address these challenges, we propose **Op-**

*Corresponding author.
[1]https://github.com/liunian-Jay/OptiSet.git

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 7125–7139
October 24-29, 2026 ©2026 Association for Computational Linguistics

tiSet, a set-centric framework that unifies set selection and set-level ranking in training. Specifically, we first propose an "Expand-then-Refine" paradigm, which first expands the query into multiple subqueries and then samples candidate passages to capture diverse perspectives. The resulting candidate pool is then further refined through reselection, forming a complementary and compact set of evidence. Secondly, we use the perplexity variation of each set with respect to the generator as a utility signal, enabling the estimation of combinatorial gain. By combining the selection paradigm and utility estimation, we can construct high-quality training data without relying on strong LLM model supervision. Finally, we propose a set-list wise training strategy that jointly set selection and ranking, preserving the diversity of selection while avoiding the overfitting of surface patterns that is common in SFT training.

In summary, our contributions can be summarized as follows:

- We introduce "Expand-then-Refine" for compact and efficient evidence set selection, complemented by a utility-driven self-synthesis and labeling scheme for training data.
- We propose a set-list-wise training strategy that finely exploits positive and negative samples to jointly optimize set selection and set-level ranking in a unified framework.
- Extensive experiments demonstrate that our framework and training improve the performance of complex combinatorial problems and make the sets compact and efficient.

2 Related Works

2.1 Retrieval-Augmented Generation

RAG has emerged as a promising approach to mitigate limitations such as knowledge obsolescence and hallucinations (Béchard and Ayala, 2024; Gade et al., 2025) in LLMs. It leverages externally retrieved knowledge to augment LLMs (Lewis et al., 2020; Zhao et al., 2024; Fan et al., 2024; Li et al., 2025). Classical RAG typically follow a "retrieve-then-read" paradigm, first using a retriever and then employing a generator to produce responses (Lewis et al., 2020). However, this approach still faces numerous challenges that limit its effectiveness.

To address these challenges, recent research has focused on identifying retrieval intent to skip unnecessary searches (Jeong et al., 2024; Cheng et al., 2024; Wang et al., 2023) and optimizing queries to better align with retrievers (Ma et al., 2023; Xu et al., 2024b; Jiang et al., 2025a). To mitigate noise and irrelevant passages, some methods introduce denoising (Wei et al., 2025), reranking (Glass et al., 2022), selection (Jiang et al., 2025b; Lee et al., 2025), and compression (Xu et al., 2024a; Li et al., 2026) between retrieval and generation.

2.2 Reranker between Retrievers and LLMs.

To bridge the gap between retrieval and generation, some studies introduce trainable middleware (Jiang et al., 2025b; Dong et al., 2025), particularly rerankers (Glass et al., 2022). Recently, LLMs have been used as ranking models and have shown strong potential, becoming an active research direction (Pradeep et al., 2023; Weller et al., 2025; Zhuang et al., 2024; Liu et al., 2025). RankZephyr (Pradeep et al., 2023) uses LLMs as a list-wise reranker to capture global passage relevance. RankR1 (Zhuang et al., 2025) and ReasonRank (Liu et al., 2025) leverage diverse reward designs and reinforcement learning to elicit LLMs' reasoning capabilities.

However, these methods often overlook the holistic nature of set-level evidence: passages can be complementary, and combining them can yield larger gains. SetR (Lee et al., 2025) takes an initial step via prompting and distillation fine-tuning. Nevertheless, it still relies on strong LLM supervision, and straightforward fine-tuning may overfit to surface patterns, limiting generalization.

In contrast, we propose a self-synthesized paradigm that unifies set selection and ranking. Our approach achieves strong performance without added inference overhead, while substantially improving document selection efficiency.

3 Method

Our framework comprises two main components: the "Expand-then-Refine" paradigm and a set–list wise training framework. The paradigm generates diverse yet efficient candidate sets, while the training framework leverages these candidates to unify the set-level selection and ranking. This section details the core methods, as shown in Fig 2.

3.1 Expand-then-Refine for Set Selection

The combinatorial selection space is extremely large, as the set size is not fixed. For an input

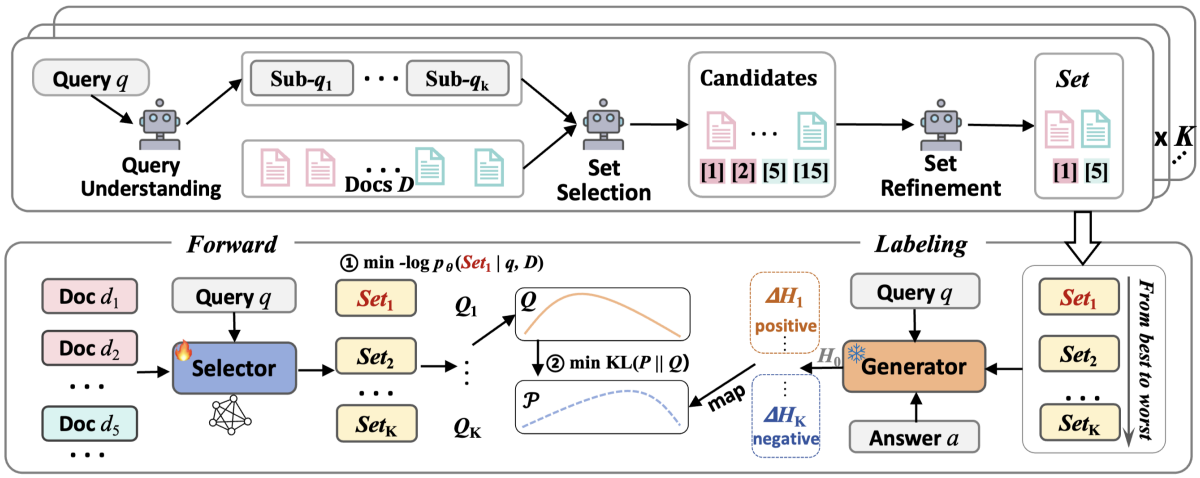

Figure 2: The framework diagram of OptiSet is shown. The upper part represents the "Expand-then-Refine" paradigm, which can serve as both a training-free framework and a training data synthesis pipeline. The lower part represents set-list wise training. It utilizes multiple sets generated by the framework, performs partial ordering and signal labeling, and trains the selector.

of 20 documents, the space contains 2^{20} possible subsets. To find a globally better yet compact evidence set, we propose an "Expand-then-Refine" paradigm that leverages LLM reasoning: expand the query for broader exploration, then refine after an initial selection for targeted re-selection. This is training-free and also supports our data synthesis.

3.1.1 Expansion: Query Understanding

Complex problems often require multi-hop reasoning, and simply identifying the original query is insufficient to obtain comprehensive information. To alleviate this limitation, we introduce an expansion phase before set selection to achieve a thorough understanding of the original query. Specifically, in this phase, we prompt the LLM to reinterpret the original query q, generating multiple subqueries or variations, each capturing a different aspect of the problem. This process can be formalized as follows:

$$Q_s = \text{Expand}(q) = \{q_1, q_2, \ldots, q_k\}, \quad (1)$$

where $q_1, q_2, \ldots, q_k$ are generated by applying query decomposition (or rewriting) to the original query q. This provides multi-perspective views of the query, enabling subsequent stages to select candidate sets with more comprehensive coverage.

3.1.2 Passage Set Selection

After the expansion stage, inspired by CoT (Wei et al., 2022), we leverage step-by-step LLM reasoning for set selection, avoiding random sampling over the large combinatorial space. Concretely, following SetR (Lee et al., 2025), we prompt the LLM to perform three reasoning steps: (1) list the key information required to answer the question and its sub-questions; (2) identify passages that contain each required item; (3) form a passage subset that provides the most comprehensive coverage.

Putting the above steps together, the LLM selects multiple candidate passages for the original query q and each sub-query $q_i \in Q_s$. For any query $q' \in \{q\} \cup Q_s$, we denote its candidate passage set as $\mathcal{C}(q') = \{s_1, s_2, \ldots, s_n\}$. Our objective is to construct an evidence set $\mathcal{S} \subseteq \mathcal{D}$ by aggregating candidates across all queries: $\mathcal{S} = \bigcup_{q' \in \{q\} \cup Q_s} \mathcal{C}(q')$. We summarize this procedure as

$$\mathcal{S}_{\text{raw}} = \text{Select}(q, Q_s, \mathcal{D}) \quad (2)$$

This produces a diverse, high-coverage evidence pool by leveraging decomposition in the expansion stage and stepwise selection, and provides a strong starting point for refinement in the next stage.

3.1.3 Refinement: Set Re-selection

After generating a diverse set of candidate passages, $\mathcal{S}_{\text{raw}}$ there may still contain redundant or irrelevant passages. To address this, we therefore apply a refinement step to filter such candidates, producing a smaller and more efficient set.

Concretely, we repeat the selection procedure in §3.1.2, but replace the original candidate pool $\mathcal{D}$ with $\mathcal{S}_{\text{raw}}$ and drop the decomposed queries Q_s, using only the original query q as a global constraint:

$$\mathcal{S}_{\text{refined}} = \text{Refine}(q, \mathcal{S}_{\text{raw}}). \quad (3)$$

After refinement, we obtain $\mathcal{S}_{\text{refined}}$, which contains fewer passages while preserving set-level relevance and complementary gains through LLM reasoning.

3.2 Self-Synthesized Training Data

We next describe how we self-synthesize training data for **OptiSet**. As summarized in Algorithm 1, we run "Expand-then-Refine" multiple times to generate diverse evidence sets and label them with set-level utility.

3.2.1 Training Data Contruction

By running the framework multiple times with high-temperature sampling, we can obtain multiple candidate sets for the same query. Formally,

$$\mathcal{S} = [s_1, \ldots, s_k] = \text{ESR}(q, \mathcal{D}, [a]). \quad (4)$$

ESR$(\cdot)$ is a shorthand for the full "Expand-then-Refine" pipeline (i.e., Eq 1,2 and 3) and $[\cdot]$ represents optional input. Note that, unlike the training-free framework, training data construction uses the gold answer a as an additional input, and its effect is detailed in § 4.5.

3.2.2 Signals Labeling

Perplexity-based utility. To score each selected set, we use perplexity (PPL) (Li et al., 2024) as a dense utility signal; lower PPL indicates better evidence for generating the gold answer:

$$\text{PPL} = \exp\left(-\frac{1}{N}\sum_{j=1}^{N}\log p(a_j \mid q, S, a_1, \ldots, a_{j-1})\right). \quad (5)$$

To stabilize the scale of this signal, we convert PPL to entropy (i.e., log-perplexity), mathematically:

$$H = \log(\text{PPL}). \quad (6)$$

Thus, different selected sets can be compared via their utility under the generator.

Adjusting Positive and Negative Samples As shown in prior work, a "preference gap" exists between retrievers and generators (Ke et al., 2024; Jiang et al., 2025b; Dong et al., 2025): low-entropy samples may still yield limited gains, since low entropy may reflect the generator's internal knowledge and retrieved evidence does not necessarily help. We therefore compute the baseline entropy (w/o passages) H_o and define

$$\Delta H = H_p - H_o, \quad (7)$$

where H_p is the entropy conditioned on the selected set. We treat sets with $\Delta H \leq 0$ as positive and those with $\Delta H > 0$ as negative.

To increase separability, we map ΔH to a signed preference score:

$$P = \begin{cases} 1 - \text{sigmoid}(\alpha \Delta H), & \Delta H \leq 0, \\ -\text{sigmoid}(\beta \Delta H), & \Delta H > 0. \end{cases} \quad (8)$$

Here, α and β are scaling coefficients estimated from the data distribution (Appendix C). After transformation, positive map to $P \in (0.5, 1]$, while negative map to $P \in [-1, -0.5)$.

3.3 Set-List Wise Training

To better leverage the sampled sets, we adopt a set-list wise training approach that jointly learns set selection and set-level ranking. The algorithm flow is summarized in Algorithm 2.

Supervised Learning To learn the optimal selection of the set, we take the highest gain score in the sampling group as an approximation of the optimal set selection and optimize it through loss on standard cross-entropy.

$$s^* = \arg\max_{s \in \mathcal{S}} P(a \mid q, s) \quad (9)$$

$$\mathcal{L}_{\text{CE}} = -\log p_\theta(s^* \mid q, \mathcal{D}) \quad (10)$$

Through optimal set fitting, the model will further enhance its set selection ability and make the output conform to the expected format.

List-Wise Ranking In our training framework, we forward propagate multiple sets sampled from the same data, calculate their PPL values, and perform a softmax operation over the set scores. The softmax function normalizes the PPL values, assigning higher probabilities to better sets while retaining diversity by allowing for the possibility of selecting suboptimal sets. We first define the target distribution over sets based on their quality scores:

$$\mathcal{P} = \text{softmax}(\text{P}), \quad \mathcal{P}_i = \frac{\exp(\text{P}_i)}{\sum_{j=1}^{m}\exp(\text{P}_j)} \quad (11)$$

Next, we define the model-predicted distribution over the same set candidates. For each set s_i, the model assigns a sequence-level log-probability:

$$\ell_i = \log p_\theta(s_i \mid q, \mathcal{D}), \quad (12)$$

and the predicted distribution is obtained via softmax normalization:

$$\mathcal{Q} = \text{softmax}(\boldsymbol{\ell}), \quad \mathcal{Q}i = \frac{\exp(\ell_i)}{\sum_{j=1}^{m} \exp(\ell_j)} \quad (13)$$

This step is followed by the calculation of the KL-divergence between the predicted distribution and the true distribution of the sets:

$$\mathcal{L}_{\text{KL}} = KL(\mathcal{P}||\mathcal{Q}) = \sum_{i=1}^{m} \mathcal{P}_i \log \left(\frac{\mathcal{P}_i}{\mathcal{Q}_i} \right) \quad (14)$$

This loss function allows the model to learn to rank sets by their quality while preserving the diversity within the sampled sets.

Objective Our final training involves combining the two losses together, formally,

$$\mathcal{L}_{\text{all}} = \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{KL}}, \quad (15)$$

where λ is the balance coefficient.

Algorithm 1 Training Data Construction

Input: Query q, gold answer a, corpus $\mathcal{D}$, sampling times K
Output: Candidate set list $\mathcal{S} = \{s_1, \ldots, s_K\}$ with quality signals $\mathbf{P}$.

Compute baseline PPL_0 with empty set
$H_0 \leftarrow \log(\text{PPL}_0)$ ▷ Baseline entropy
for $k = 1$ to K **do**
  $Q_s^{(k)} \leftarrow \text{Expand}(q, a)$
  $s_{raw}^{(k)} \leftarrow \text{Select}(q, Q_s^{(k)}, \mathcal{D}, a)$
  $s^{(k)} \leftarrow \text{Refine}(q, s_{raw}^{(k)}, a)$
  Compute PPL_k conditioned on (q, s_k, a)
  $H_k \leftarrow \log(\text{PPL}_k)$
  $\Delta H_k \leftarrow H_k - H_0$ ▷ Entropy difference
  $P_k \leftarrow \text{Transform}(\Delta H_k)$ ▷ Eq. (8)
end for
return $\mathcal{S} = \{s_1, \ldots, s_K\}$, $\mathbf{P} = \{P_1, \ldots, P_K\}$

4 Experiments

4.1 Implementation Details

Training Data. We sample a subset of the training split from HotpotQA datasets, running our framework 10 times per question. After collecting 20K samples, we apply an initial filter to remove examples that are entirely incorrect or lack discriminative signal. To reduce **bias from passage IDs** in the training distribution, we shuffle the the order of the input passage. **three** times per query. Finally, we retain 5 sets per sample and randomly subsample 8K samples for training.

Algorithm 2 Set-List Wise Training

Input: Training data $(q, \mathcal{D}, \mathcal{S}, \mathbf{P})$, model p_θ
Output: Updated model parameters θ

for each training instance $(q, \mathcal{D}, \mathcal{S})$ **do**
  $s^* \leftarrow \arg\max_{s_i \in \mathcal{S}} P_i$ ▷ Eq. (9)
  $\mathcal{L}_{\text{CE}} \leftarrow -\log p_\theta(s^* \mid q, \mathcal{D})$ ▷ Eq. (11)
  for each set $s_i \in \mathcal{S}$ **do**
    $\ell_i \leftarrow \log p_\theta(s_i \mid q, \mathcal{D})$
  end for
  $\mathcal{Q} \leftarrow \text{softmax}(\boldsymbol{\ell})$ ▷ Model distribution
  $\mathcal{P} \leftarrow \text{softmax}(\mathbf{P})$ ▷ Target distribution
  $\mathcal{L}_{\text{KL}} \leftarrow \text{KL}(\mathcal{P} \| \mathcal{Q})$ ▷ Eq. (14)
  $\mathcal{L} \leftarrow \mathcal{L}_{\text{CE}} + \lambda \mathcal{L}_{\text{KL}}$ ▷ Eq. (15)
  Update θ using $\nabla_\theta \mathcal{L}$
end for

Training Details. Training Details We finetune LLaMA3.1-8B with LoRA (r=128, α=32, dropout=0.05). During SFT, we train for 1 epoch with a learning rate of 3e-5. The loss balancing factor λ is set to 0.1. All experiments are conducted on 2× A100 GPUs.

Inference Details. During inference, we use Contriever (Izacard et al., 2021) as the retriever and set k to 20. For all datasets, we use 21M English Wikipedia dump as the source passages for the retrieval (Karpukhin et al., 2020). Prompts for the experiments can be found in Appendix F.

4.2 Datasets and Evaluation

Datasets. We evaluate our OptiSet on four open-domain QA datasets covering both multi-hop and single-hop. The multi-hop QA benchmarks include HotpotQA (Yang et al., 2018), Bamboogle (Press et al., 2023) and MuSiQue (Trivedi et al., 2022b). For general QA, we use TriviaQA (Joshi et al., 2017). We adopt 500-sample subsets of HotpotQA and TriviaQA for efficiency, following (Trivedi et al., 2022a; Kim et al., 2024).

Evaluation. We report two metrics: EM and F1. Following prior work Asai et al. (2023); Mallen et al. (2022); Jiang et al. (2025c), we use a non-strict EM where a prediction is correct if it contains the gold answer. F1 measures token-level overlap between the predicted and gold answers. Since

Method	HotpotQA			Bamboogle			MuSiQue			TriviaQA		
	EM	F1	doc	EM	F1	doc	EM	F1	doc	EM	F1	doc
Zero-shot Baselines & Ours Training-Free												
Vanilla	25.20	26.88	0.00	13.60	18.18	0.00	6.54	10.14	0.00	63.60	64.40	0.00
NaiveRAG	33.20	35.13	3.00	17.60	23.83	3.00	8.44	13.55	3.00	69.00	71.14	3.00
Ours†	38.20	41.72	2.20	22.40	29.08	2.30	10.84	15.45	2.13	72.40	72.58	2.93
$SetR_O$	*37.00*	*41.11*	*3.45*	*20.00*	*27.58*	*4.05*	*10.76*	*16.16*	*3.91*	*74.20*	*73.93*	*4.44*
Fine-tuned Baselines & Ours												
ReasonIR	37.40	40.22	3.00	24.80	29.85	3.00	10.47	14.94	3.00	72.60	73.43	3.00
RankZephyr	33.60	39.19	3.00	20.80	28.69	3.00	10.76	15.36	3.00	72.80	73.05	3.00
SetWise	35.40	40.41	3.00	21.60	28.01	3.00	9.64	14.86	3.00	74.20	74.41	3.00
RankR1	38.00	42.23	3.00	23.20	29.72	3.00	10.30	15.33	3.00	73.00	73.40	3.00
Rank1	35.20	40.09	3.00	22.40	30.60	3.00	10.51	**15.84**	3.00	74.20	74.61	3.00
Ours	**38.80**	**43.30**	2.34	**25.60**	**32.67**	2.47	**10.84**	15.49	2.47	**75.40**	**75.83**	2.70

Table 1: EM/F1(%) of different methods experimented on four datasets. The best and second best scores are highlighted in **bold** and underlined, respectively. *Gray* is not used for comparison. Ours† donate Training-free.

longer response may improve EM via coverage but introduce noise that lowers F1, evaluating both metrics allows for a more balanced evaluation.

4.3 Baselines

We select several of the most representative methods for comparison. (1) Vanilla: direct answering without retrieval. (2) Naive RAG, which is the classic "retrieve-then-read" paradigm. (3) $SetR_O$ (Lee et al., 2025): a zero-shot version of set selection based on LLM reasoning. (4) ReasonIR (Shao et al., 2025): a powerful 7B retriever with strong inferential retrieval capabilities. (5) RankZephyr (Pradeep et al., 2023): The classic approach is to use LLM for list-wise ranking. (6) SetWise (Zhuang et al., 2024): a setwise algorithm that uses heap sort to produce a full ranking. (7) RankR1 (Zhuang et al., 2025): a reasoning reranker trained with RL. (8) Rank1 (Weller et al., 2025): reranking with increased test-time computation. All retrieval-based methods use top-20 passages. All reranker only selected the top-3 evidence. Details experimental settings are shown in the Appendix A and B.

4.4 Main Results

In our experiments as shown in Table 1, we summarize several key findings:

(1) The role of re-ranking or selection. Adding retrieval with re-ranking consistently improves over direct answering and naive retrieval. Compared with *NaiveRAG*, our method improves EM/F1 by +5.00/+6.59 on HotpotQA and by +4.80/+5.25 on Bamboogle in the zero-shot setting, demonstrating that the improvement comes from effectively selecting high-quality evidence.

(2) Reasoning vs. Non-reasoning. We observe that rerankers with explicit reasoning capabilities generally outperform non-reasoning approaches. Methods such as *RankR1* achieve higher EM/F1 than non-reasoning ranking baselines (e.g., *RankZephyr*) on most datasets. We attribute this advantage to cognitive chaining, enabling the model to identify the intermediate steps required for multi-hop reasoning and better assess passage relevance. Notably, our method remains competitive without explicit reasoning, indicating the effectiveness of our selection mechanism.

(3) Expand-then-Refine vs. SetR. $SetR_O$ is competitive, suggesting reasoning-based selection is effective, but it tends to over-select passages, increasing cost. Our expand-then-refine paradigm achieves higher accuracy with fewer documents (e.g., Bamboogle: +2.40 EM with 2.30 vs. 4.05 doc). Overall, refinement is crucial for mitigating over-selection and improving evidence efficiency.

(4) Efficiency. Our approach improves both generation and training efficiency. It selects fewer passages than prior rerankers, reducing context length and generation inference cost, and it avoids reinforcement learning, simplifying training and improving stability without sacrificing performance.

4.5 Ablation Study I: Paradigm

We ablate the expand-then-refine paradigm by removing each module while keeping all other set-

Method	HotpotQA				Bamboogle				MuSiQue				Avg.
	EM	F1	Avg	doc	EM	F1	Avg	doc	EM	F1	Avg	doc	
Pipeline Ablation: Effectiveness of Different Modules													
Selection-only	37.00	41.11	39.06	3.45	20.00	27.58	23.79	4.05	10.76	**16.16**	**13.46**	3.91	25.44
w/o Expand	36.60	40.84	38.72	**1.99**	20.80	28.64	24.72	**2.07**	10.63	15.89	13.26	**2.02**	25.57
w/o Refine	**39.00**	**42.13**	**40.57**	4.45	19.20	26.56	22.88	5.22	10.47	15.76	13.12	4.51	25.52
$w/$ All	38.20	41.72	39.96	2.20	**22.40**	**29.08**	**25.74**	2.30	**10.84**	15.45	13.15	2.13	**26.28**
Ours+Answer	*39.00*	*43.64*	*41.32*	*2.09*	*23.20*	*30.67*	*26.94*	*1.78*	*13.20*	*17.91*	*15.55*	*1.95*	*27.94*
Training Ablation: Comparison of Training Strategies													
SFT	32.80	36.76	34.78	**1.95**	20.80	26.52	23.66	**1.94**	7.57	12.18	9.88	**1.94**	22.77
DPO	37.40	41.76	39.58	2.87	20.80	28.73	24.77	3.19	10.67	15.18	12.93	3.13	25.76
Ours	**38.80**	**43.30**	**41.05**	2.34	**25.60**	**32.67**	**29.13**	2.47	**10.84**	**15.49**	**13.16**	2.47	**27.78**

Table 2: Ablation studies on both pipeline modules and training strategies. **Bold** and underline denote the best and second-best results in each group respectively. *Gray* is not used for comparison.

tings fixed (top part of Table 2).

Effect of *Refinement*. *Refine* is the main driver of efficiency. Removing *Expand* (*w/o Expand*) yields the most compact evidence sets (about 2doc) with comparable performance to other variants, indicating that refinement alone can perform strong selection under a tight budget.

Effect of *Expansion*. Using *Expand* alone is not consistently beneficial: *w/o Refine* improves HotpotQA but degrades other datasets while selecting substantially more passages. In contrast, *Expand* complements *Refine*: compared with *w/o Expand*, the full paradigm (*All*) yields higher performance (e.g., 25.57→26.28) with only a marginal increase in evidence size. These results suggest that expansion broadens candidate coverage, allowing *Refine* to select better evidence than refinement alone.

Effect of *Answer*. During data synthesis, we additionally condition on the gold answer as guidance. This consistently resulted in higher performance, further improving the quality of the training data.

4.6 Ablation Study II: Training

We further compare training strategies for the selector under the same evaluation protocol (Table 2).

SFT vs. DPO. SFT produces the smallest evidence sets but substantially hurts accuracy, suggesting it may learn overly simple selection patterns. DPO improves accuracy over SFT but selects more passages and remains suboptimal.

Set-List wise training. Our training achieves the best overall performance with a compact evidence size. It consistently outperforms DPO across datasets, with particularly large gains on Bamboogle (Avg 29.13 vs. 24.77) while using fewer passages (2.47 vs. 3.19 doc). These results suggest that set–list-wise supervision improves both accuracy and evidence efficiency compared to point-wise SFT and pair-wise DPO.

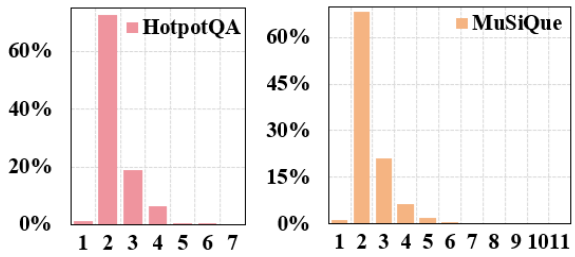

Figure 3: Illustration of the distribution of the number of documents dynamically selected from a set.

4.7 Dynamic Set Distribution and Efficiency

We analyze the distribution of selected set sizes to characterize the behavior of dynamic evidence selection. As shown in Fig. 3, most questions are answered with compact evidence sets of 2-3 passages, indicating strong efficiency. Meanwhile, the method occasionally selects larger sets, suggesting it can adaptively increase coverage when additional information is required. This automatic trade-off, however, reduces user control over the evidence budget, which is a limitation.

4.8 Redundancy Analysis

To quantify redundancy reduction, we follow Clarke et al. (2008); Tsai et al. (2010) and report $Novel$ (see Appendix D for details), a novelty-aware metric that penalizes repetition after accounting for correlation.

Metric	Model	HotpotQA	Bamboogle	MuSiQue
Jaccard				
Novel@all	RankZep	79.94	84.49	84.43
	Ours	**84.28**	**87.21**	**87.48**
Novel@2	RankZep	81.80	87.90	86.55
	Ours	**85.56**	**88.92**	**88.56**
Novel@3	RankZep	79.17	**85.28**	84.51
	Ours	**80.58**	85.03	**84.97**
BERTScore F1				
Novel@all	RankZep	56.34	58.96	59.08
	Ours	**64.76**	**65.60**	**66.41**
Novel@2	RankZep	65.57	68.90	68.40
	Ours	**67.68**	**69.71**	**69.72**
Novel@3	RankZep	55.94	59.40	59.27
	Ours	**56.74**	**59.53**	**59.62**

Table 3: Passage novelty comparison between RankZephyr and our method.

Table 3 shows that our method achieves high overall novelty when averaged across all examples, partly because it selects fewer passages on average; importantly, selecting fewer passages is itself a desirable form of redundancy reduction. To control for set size, we further compare the most frequent cases with 2- and 3-passage sets against *RankZephyr*. Under matched sizes, our novelty remains higher than the classic list-wise ranker, indicating room for improving redundancy removal beyond compact selection.

4.9 What Gains does Self-Labeling bring?

To verify whether our gains mainly come from higher passage recall, we compare the change in hit rate between OptiSet and other reranking baselines.

As shown in Figure 4, compared to *RankZephyr* and *Rank1*, we rarely observed significant increases in OptiSet's hit rate, except for Hotpot; most showed only minor changes, or even a decrease in hit rate. Nevertheless, answer quality still improved in these cases. Since the generator was not trained, these improvements cannot be explained by generator adaptation, suggesting that the improvements stem from selecting higher-quality (but not necessarily higher recall) evidence sets. We hypothesize that the performance improvement is due to the combined gain from different passages selected by OptiSet, rather than from individual gold passages. However, different generators may have different document combination preferences, and improving generalization is a future direction.

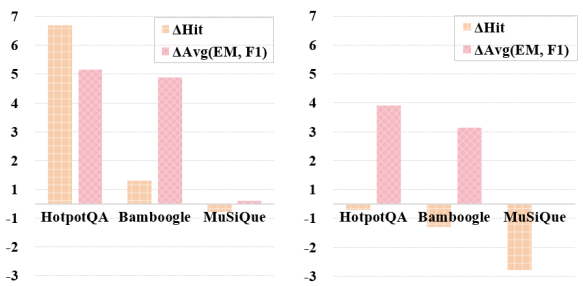

Figure 4: Illustration of gain. Changes in hit of the gold answer and QA performance using OptiSet. The left side compares to RankZephyr, and the right side compares to Rank1. Avg(EM, F1) is the metric.

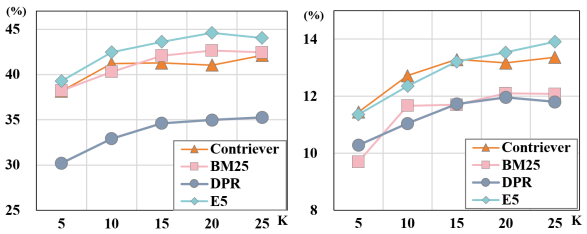

Figure 5: Illustration showing how generation performance changes as the candidate sizes changes or different retrievers are replaced. The left part is HotpotQA, and the right part is MuSiQue. For fairness, Avg(EM, F1) is uesd as the evaluation metric.

4.10 Robustness to Different Retrievers and K

To evaluate the generalization ability of our **optiSet** to different candidate sizes and its robustness to different retrieval systems, we observed the following:

Retriever robustness. OptiSet performs consistently across retrieval systems, achieving strong results with both dense (e.g., E5) and sparse (BM25) retrievers. This suggests that our selector is largely retriever-agnostic and does not overfit to a specific candidate distribution induced by the training setup.

Sensitivity to K. Varying K follows the expected information-coverage trend: larger candidate pools generally improve accuracy by providing more relevant evidence to choose from. Importantly, increasing K beyond the training setting (e.g., K=25) still yields gains, indicating that **optiSet** generalizes beyond the training-time candidate-size bias.

5 Conclusion

In this work, we investigate the combinatorial selection problem and propose OptiSet, a unified framework for set selection and set-level ranking in retrieval-augmented generation. OptiSet uses an *expand-then-refine* paradigm to construct compact

evidence sets and a self-synthesis scheme to label training data. We further introduce set-listwise training that jointly optimizes set selection and set-level ranking. Experiments show that OptiSet improves both efficiency and performance.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China [62576126]; and the Heilongjiang Provincial Natural Science Foundation of China [2023ZX01A11].

Limitations

Despite achieving impressive performance, our approach still has some limitations in certain aspects:

- LLM-based methods are limited by context length; how to accept more candidates is a key research direction for us in the future.
- Many RL baselines have achieved better performance. Although we achieved excellent performance without RL, how to stimulate inference ability through RL is still worth exploring.
- While the selection is dynamic top-k, which increases intelligence but sacrifices flexibility, incorporating K-budget into the instructions is a very worthwhile area of research.

References

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

Patrice Béchard and Orlando Marquez Ayala. 2024. Reducing hallucination in structured outputs via retrieval-augmented generation. *arXiv preprint arXiv:2404.08189*.

Qinyuan Cheng, Xiaonan Li, Shimin Li, Qin Zhu, Zhangyue Yin, Yunfan Shao, Linyang Li, Tianxiang Sun, Hang Yan, and Xipeng Qiu. 2024. Unified active retrieval for retrieval augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 17153–17166.

Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 659–666.

Guanting Dong, Yutao Zhu, Chenghao Zhang, Zechen Wang, Ji-Rong Wen, and Zhicheng Dou. 2025. Understand what llm needs: Dual preference alignment for retrieval-augmented generation. In *Proceedings of the ACM on Web Conference 2025*, pages 4206–4225.

Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 6491–6501.

Anoushka Gade, Jorjeta G Jetcheva, and Hardi Trivedi. 2025. It's about time: Incorporating temporality in retrieval augmented language models. In *2025 IEEE Conference on Artificial Intelligence (CAI)*, pages 75–82. IEEE.

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2701–2715.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning. *arXiv preprint arXiv:2112.09118*.

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.

Yi Jiang, Lei Shen, Lujie Niu, Sendong Zhao, Wenbo Su, and Bo Zheng. 2025a. Qagent: A modular search agent with interactive query understanding. *arXiv preprint arXiv:2510.08383*.

Yi Jiang, Sendong Zhao, Jianbo Li, Haochun Wang, and Bing Qin. 2025b. GainRAG: Preference alignment in retrieval-augmented generation through gain signal synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10746–10757. Association for Computational Linguistics.

Yi Jiang, Sendong Zhao, Jianbo Li, Haochun Wang, Lizhe Zhang, Yan Liu, and Bing Qin. 2025c. Cocoa: Collaborative chain-of-agents for parametric-retrieved knowledge synergy. *arXiv preprint arXiv:2508.01696*.

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly

supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *EMNLP (1)*, pages 6769–6781.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10438–10451.

Jaehyung Kim, Jaehyun Nam, Sangwoo Mo, Jongjin Park, Sang Woo Lee, Minjoon Seo, Jung Woo Ha, and Jinwoo Shin. 2024. Sure: Summarizing retrievals using answer candidates for open-domain qa of llms. In *12th International Conference on Learning Representations, ICLR 2024*.

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.

Jianbo Li, Yi Jiang, Sendong Zhao, Bairui Hu, Haochun Wang, and Bing Qin. 2026. Arcaligner: Adaptive recursive aligner for compressed context embeddings in rag. *arXiv preprint arXiv:2601.05038*.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. 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 7602–7635.

Xiaoxi Li, Jiajie Jin, Yujia Zhou, Yuyao Zhang, Peitian Zhang, Yutao Zhu, and Zhicheng Dou. 2025. From matching to generation: A survey on generative information retrieval. *ACM Transactions on Information Systems*, 43(3):1–62.

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

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness and limitations of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*, 7.

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

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 5687–5711.

Stephen E Robertson and Steve Walker. 1994. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR'94: Proceedings of the Seventeenth Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval, organised by Dublin City University*, pages 232–241. Springer.

Rulin Shao, Rui Qiao, Varsha Kishore, Niklas Muennighoff, Xi Victoria Lin, Daniela Rus, Bryan Kian Hsiang Low, Sewon Min, Wen-tau Yih, Pang Wei Koh, and 1 others. 2025. Reasonir: Training retrievers for reasoning tasks. *arXiv preprint arXiv:2504.20595*.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2022a. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. *arXiv preprint arXiv:2212.10509*.

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

Flora S Tsai, Wenyin Tang, and Kap Luk Chan. 2010. Evaluation of novelty metrics for sentence-level novelty mining. *Information Sciences*, 180(12):2359–2374.

Jingbo Wang, Sendong Zhao, Haochun Wang, Bing Qin, and Ting Liu. 2026. Analyzing and mitigating cross-lingual degradation in multilingual medical vqa. *Preprint*, arXiv:2608.22363.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10303–10315.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. 2025. InstructRAG: Instructing retrieval-augmented generation via self-synthesized rationales. In *The Thirteenth International Conference on Learning Representations*.

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

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024a. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. In *12th International Conference on Learning Representations, ICLR 2024*.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024b. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM Web Conference 2024*, pages 1362–1373.

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.

Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Jie Jiang, and Bin Cui. 2024. Retrieval-augmented generation for ai-generated content: A survey. *arXiv preprint arXiv:2402.19473*.

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

Shengyao Zhuang, Honglei Zhuang, Bevan Koopman, and Guido Zuccon. 2024. A setwise approach for effective and highly efficient zero-shot ranking with large language models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 38–47.

A Dataset

Here, we introduce in detail the datasets we used, which are four datasets on two tasks

HotpotQA (Yang et al., 2018): It is a multi-hop question answering dataset based on Wikipedia. Given the cost constraints of the experiment, we used a subsample of the Trivedi et al. (2022a) dataset, which was obtained by extracting 500 questions from the validation set of each dataset.

Bamboogle (Press et al., 2023): Bamboogle is a small multi-hop question-answering benchmark suite developed to study the "combinatorial reasoning gap" in language models. It contains 125 questions. Each article is manually written out, requiring two hops of information to answer. It deliberately retains questions where "evidence can be found on Wikipedia, but searching for the question directly on mainstream search engines often fails to yield the correct answer/the summary will be incorrect."

MuSiQue (Trivedi et al., 2022b): By iteratively selecting composable single-hop question pairs, we created questions spanning 2-4 hops, which are generally more difficult.

TriviaQA (Joshi et al., 2017): A compilation of trivia questions paired with answers, both originally pulled from online sources. For the sake of inference efficiency, we randomly sample 500 entries here.

Task Type	Dataset	# Samples
Multi-HopQA	HotpotQA	500
	Bamboogle	125
	MuSiQue	2,417
One-HopQA	TriviaQA	500

Table 4: Description of tasks and evaluation datasets.

B Baselines

We selected several of the most representative methods for comparison.

(1) **Vanilla**: Direct answer generation without any external retrieval. The model relies solely on its parametric knowledge (and the given prompt) to produce an answer, serving as a closed-book baseline that isolates the effect of adding retrieval.

(2) **Naive RAG**: The standard "retrieve-then-read" pipeline. A retriever first fetches top-k passages, and the generator then conditions on the concatenated retrieved evidence to produce the final answer.

(3) **SetR**: Leveraging the model's powerful reasoning capabilities, it performs set selection through long-chain cognition. Its fine-tuned version requires strong model supervision; since weights are not open, we compare it with a zero-shot version.

(4) **ReasonIR** (Shao et al., 2025): ReasonIR trains a strong retriever with a synthetic data pipeline that generates reasoning-required queries and "plausible-but-useless" hard negatives for each document. Due to its large capacity and strong performance, we use it as a representative strong retriever and employ it as the reranker used to supply candidates for reranking.

(5) **RankZephyr** (Pradeep et al., 2023): A representative listwise LLM reranker built on Zephyr-7B. It is instruction-distilled from GPT-4/RankGPT-style teacher signals to directly output an ordered list of document indices for a candidate set.

(6) **SetWise** (Zhuang et al., 2024): A cost-efficient zero-shot reranking strategy combining setwise prompting with a sorting procedure. Instead of pairwise comparisons, the LLM selects the most relevant item from a small set of candidates each time, and a heap-sort-style process aggregates these selections into a full ranking with fewer LLM calls.

(7) **Rank-R1** (Zhuang et al., 2025): A reasoning-enhanced reranker that uses setwise prompting plus reinforcement learning (e.g., GRPO). It improves "think-then-rank" behavior using only relevance supervision, rather than relying solely on SFT or prompting.

(8) **Rank1** (Weller et al., 2025): A reranking approach that brings test-time compute to ranking via reasoning-trace distillation. It curates and releases 600K+ reasoning traces from strong reasoning models (e.g., DeepSeek-R1) on MS MARCO and distills them into a smaller reranker that generates a reasoning trace before producing relevance judgments/rankings.

All retrieval-based methods use top-20 passages. And all reranker only selected the top-3 evidence.

C Training & Inference Details

Training Data. We primarily sampled 20k data poinsts from training dataset of HotpotQA (Yang et al., 2018), running the framework 10 times per question. After obtaining 20k data points, we performed initial filtering, removing those that were completely erroneous or lacked significant discriminative power. During training, each data point was kept with 5 ranking candidates. Additionally, to reduce the bias in paragraph IDs within the training data distribution, we randomly shuffled the input data three times (equivalent to 3 epochs). Finally, our preliminary experiments showed that convergence could be achieved with only partial training, so we randomly sampled 8k samples for analysis.

Training Details. Training Details We fine-tune LLaMA3.1-8B with LoRA (r=128, α=32, dropout=0.05). During SFT, we train for 1 epoch (he data has been shuffled 3 times during construction) with a learning rate of 3e-5. All experiments are conducted on 2$\times$ A100 GPU.

Inference Details. During inference, we use Contriever (Izacard et al., 2021) as the retriever and set k to 20. For all datasets, we use 21M English Wikipedia dump as the source passages for the retrieval (Karpukhin et al., 2020). In addition, we also used a series of other retrievers in the analysis section, including BM25 (Robertson and Walker, 1994), DPR (Karpukhin et al., 2020), and E5 (Wang et al., 2022). Furthermore, we use the vllm[2] library to speed up inference.

Evaluation We report two metrics: EM and F1. Following prior work Asai et al. (2023); Mallen et al. (2022); Wang et al. (2026), we use a non-strict EM where a prediction is correct if it contains the gold answer. F1 measures token-level overlap between the predicted and gold answers. Since longer response may improve EM via coverage but introduce noise that lowers F1, evaluating both metrics allows for a more balanced evaluation. In addition, to measure efficiency, we reported the documents used after the reranking.

[2]https://docs.vllm.ai/en/latest/

Fitting α and β. We estimate (α, β) by minimizing a goodness-of-fit loss on the resulting score distributions, using the sum of Kolmogorov–Smirnov distances (negative scores vs. $\mathrm{Unif}[-1, -0.5]$ and positive scores vs. $\mathrm{Unif}[0.5, 1]$), with $\alpha, \beta \in [0.01, 10]$ optimized via `scipy.optimize.minimize`.

D Novelty Metrics

We evaluate redundancy removal using a novelty indicator adapted from novelty-based evaluation (Clarke et al., 2008; Tsai et al., 2010). Given the selected passages for a query in ranked order $\mathcal{P} = \langle p_1, \ldots, p_n \rangle$, we compute the marginal novelty of each passage as

$$g_1 = 1, \qquad g_i = 1 - \max_{j<i} \mathrm{Sim}(p_i, p_j) \ (i > 1), \tag{16}$$

where $\mathrm{Sim}(\cdot, \cdot) \in [0, 1]$ measures passage similarity (we use Jaccard or BERTScore-F1). The novelty of the set is the average marginal gain:

$$\mathrm{Novel}(\mathcal{P}) = \frac{1}{n} \sum_{i=1}^{n} g_i. \tag{17}$$

This matches our implementation: the first passage is treated as fully new, and each subsequent passage contributes 1 minus its maximum similarity to earlier selections.

Novel@all / Novel@2 / Novel@3. We report dataset-level novelty by averaging $\mathrm{Novel}(\mathcal{P})$ over queries. **Novel@all** averages over all queries. Since novelty is affected by the number of selected passages (smaller sets tend to have higher novelty), we additionally compute size-controlled scores: **Novel@2** and **Novel@3** average only over queries where the selected set size is exactly 2 or 3, respectively. This allows fair comparison of redundancy removal independent of selection size.

E Case Study

In this section, we conduct two case studies and a quantitative analysis to reduce the impact of randomness.

Fig. 6 compares **OptiSet** with **RankZephyr** quantitatively. For clarity, tied cases are omitted, although they account for over 90% of all comparisons. The consistently low loss rate shows that OptiSet's gains are stable rather than random.

Fig. 7 and Fig. 8 provide two representative case studies. These examples illustrate the synergistic improvements achieved by OptiSet compared with a listwise reranker.

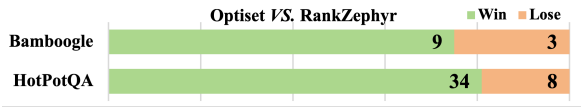

Figure 6: OptiSet *Vs.* RankZephyr. For clarity, tied cases are omitted, although they account for over 90% of all comparisons.

E.1 Transferability to Different Tasks

Although our work primarily focuses on question answering, we further evaluate OptiSet on PubHealth and ARC-Challenge to examine its transferability to other task formats. The results are reported in Table 5.

Method	PubHealth		ARC-Challenge	
	ACC	#Docs	ACC	#Docs
NaiveRAG	62.01	3.00	77.05	3.00
OptiSet	64.03	3.10	76.88	3.05

Table 5: Results on non-QA tasks.

OptiSet improves performance on PubHealth with a comparable number of retrieved documents, and remains on par with NaiveRAG on ARC-Challenge. Together with the QA transferability shown in the main experiments, these results indicate that OptiSet is robust across task formats, while stronger gains on non-QA tasks may require task-specific synthesized training data.

F Prompts

All the prompts we used are presented as follows:

Task:Prompt used by “Expanding” stage
Generate all questions needed to find every piece of information required to answer the final question. Search Query: $\{question\}$ Answers: $\{answers\}$ Rules: - Each query must stand alone without referencing previous queries or using pronouns like “it,” “they,” or “that.” Always repeat the necessary entities. Output format exactly: "### Queries: <Query 1>\n<Query 2>... \n<Query n>\n"

Table 6: The prompt used by “Expansion” stage.

Task:Prompt used by “Sampling” stage
I will provide you with num passages, each indicated by a numerical identifier []. Select the passages based on their relevance to the search query: $\{question\}$. $\{context\}$\n Search Query: $\{question\}$ Sub-Queries:\n$\{queries\}$ Answers: $\{answers\}$ Please follow the steps below: Step 1. Please list up the information requirements to answer the query and sub-queries. Step 2. For each requirement in Step 1, find the passages that has the information of the requirement. Step 3. Choose the passages that mostly covers clear and diverse information to answer the query. Number of passages is unlimited. The format of final output should be ’### Final Selection: [] [].\n’, e.g., ’### Final Selection: [2] [1].\n’.

Table 7: The prompt used by “Selection” stage.

Task:Prompt used by “Refining” stage
I will provide you with num passages, each indicated by a numerical identifier []. Select the passages based on their relevance to the search query: $\{question\}$. $\{context\}$\n Search Query: $\{question\}$\n Answers: $\{answers\}$ Step 1. Identify whether the above set of paragraphs contains irrelevant or repetitive paragraphs. Step 2. Exclude a small amount of irrelevant or repetitive information. The format of final output should be ’### Final Selection: [] [].\n’, e.g., ’### Final Selection: [2] [1].\n’.

Table 8: The prompt used by “Refinement” stage.

Task:Prompt used by “Answer”
$\{context\}$\n Answer the question below concisely in a few words. Question: $\{question\}$\n

Table 9: The prompt used for “Question Answering”.

Task:Prompt used in "Training"
I will provide you with $\{num\}$ passages, each indicated by a numerical identifier []. Select the passages based on their relevance to the search query: $\{question\}$. $\{context\}$\n Search Query: $\{question\}$\n Please follow the steps below: Step 1. Please list up the information requirements to answer the query and sub-queries. Step 2. For each requirement in Step 1, find the passages that has the information of the requirement. Step 3. Choose the passages that mostly covers clear and diverse information to answer the query. Number of passages is unlimited. The format of final output should be '### Final Selection: [] [].\n', e.g., '### Final Selection: [2] [1].\n'. \n Output only the required format. No explanations, no headings, no bullets, no markdown.

Table 10: The prompt used in Selector "Training" and "inference".

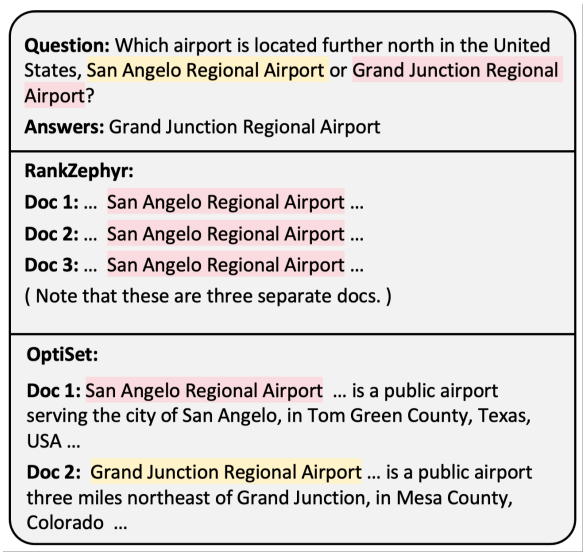

Figure 7: Case 1: This presents a question requiring two distinct pieces of information. Our **OptiSet** achieves superior combinatorial gain, whereas **RankZephyr**—despite ranking documents that are individually relevant—exhibits information redundancy.

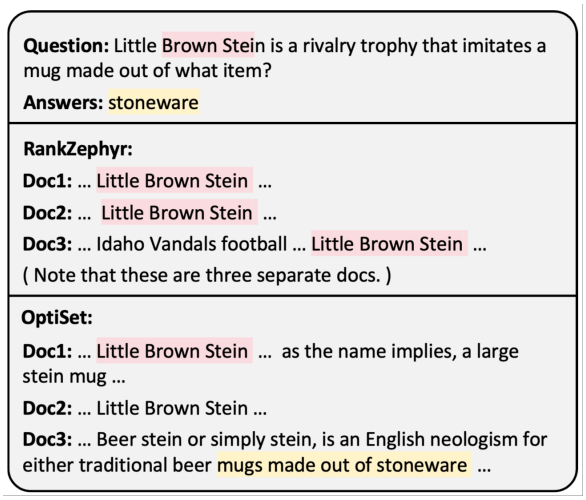

Figure 8: Case 2: This query necessitates multi-hop reasoning; our **OptiSet** provides a more comprehensive set of supporting evidence, whereas **RankZephyr** is only able to furnish partial evidence.