Optimizing Retrieval-Augmented Generation through Adaptive Rewrite Selection

Anonymous ACL submission

Abstract

With the advancement of Retrieval-Augmented Generation (RAG) in open-domain question answering (ODQA), query rewriting has gained increasing attention as a means to better handle complex queries. By generating alternative formulations of a question, query rewrites can help bridge the gap between user intent and the structure of retrieved knowledge, thereby enhancing multi-hop reasoning. However, existing approaches often produce static rewrites that lack adaptability and fail to capture the evolving intent behind complex queries. To overcome this challenge, we propose **ARS-RAG**, an adaptive rewrite selection approach to dynamically determine the optimal number of rewrites for each query. ARS-RAG generates multiple rewrites for a given query and dynamically selects the effective ones. We train a self-supervised ranker to assess the relevance of each rewrite, as well as a contextual bandit selector that dynamically selects the optimal top-K rewrites. This enables query-specific adaptation and efficient retrieval. Experimental results on four ODQA datasets confirm the effectiveness of ARS-RAG. Importantly, our adaptive selection strategy introduces negligible overhead and requires no additional fine-tuning of the rewriter.

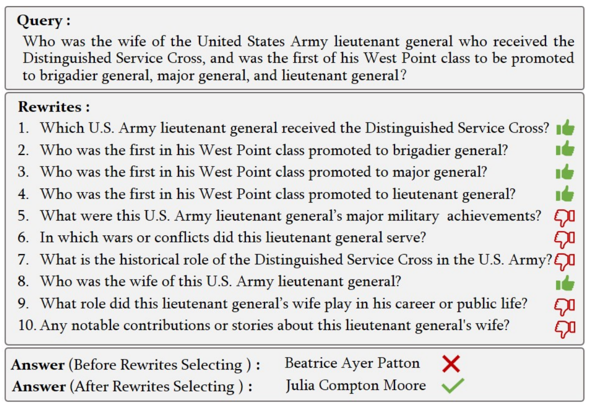
1 Introduction

Large language models (LLMs) have shown strong performance in natural language understanding and generation tasks (Achiam et al., 2023; Brown, 2020). However, in open-domain question answering (ODQA), LLMs often suffer from hallucinations (Ji et al., 2023) due to limitations in the timeliness, accuracy, and coverage of their internal parameterized knowledge. Integrating retrieved external knowledge has proven effective in mitigating hallucinations (Jiang et al., 2023).

Retrieval-augmented generation (RAG) integrates a retriever and a generator as its core components (Ma et al., 2023; Lewis et al., 2020). The retriever identifies relevant knowledge documents from a corpus, and the generator uses this information to enhance the LLMs' output reliability. RAG shows significant potential in knowledge-intensive ODQA tasks (Gao et al., 2023b; Guan et al., 2025).

Standard RAG often struggles with multi-hop reasoning or commonsense reasoning, as single-vector matching often fails to capture nuanced dependencies, creating a gap between the input and the corpus. To address this, Query2doc (Wang et al., 2023) expands queries by generating pseudo-documents using few-shot prompting, HyDE (Gao et al., 2023a) constructs hypothetical documents for effective vector matching, and the Rewrite-Retrieve-Read framework (Ma et al., 2023) introduces a rewriter before retrieval to better align the query with the corpus.

Existing approaches face two major limitations: they struggle to determine how many rewrites are needed to solve a complex query, and they lack effective mechanisms to assess the quality of these



Query :
Who was the wife of the United States Army lieutenant general who received the Distinguished Service Cross, and was the first of his West Point class to be promoted to brigadier general, major general, and lieutenant general?

Rewrites :
1. Which U.S. Army lieutenant general received the Distinguished Service Cross? 👍
2. Who was the first in his West Point class promoted to brigadier general? 👍
3. Who was the first in his West Point class promoted to major general? 👍
4. Who was the first in his West Point class promoted to lieutenant general? 👍
5. What were this U.S. Army lieutenant general's major military achievements? 👎
6. In which wars or conflicts did this lieutenant general serve? 👎
7. What is the historical role of the Distinguished Service Cross in the U.S. Army? 👎
8. Who was the wife of this U.S. Army lieutenant general? 👍
9. What role did this lieutenant general's wife play in his career or public life? 👎
10. Any notable contributions or stories about this lieutenant general's wife? 👎

Answer (Before Rewrites Selecting) : Beatrice Ayer Patton ✗
Answer (After Rewrites Selecting) : Julia Compton Moore ✓

Figure 1: **Impact of Irrelevant Rewrites on Answer Accuracy.** Expanding a query into 10 rewrites yields both relevant (green: Q1–Q4, Q8) and irrelevant (red: Q5–Q7, Q9–Q10) variants. Irrelevant rewrites introduce semantic noise, while selecting top-K relevant ones leads to the correct answer ("Julia Compton Moore"), highlighting the importance of precise rewrite selection.

rewrites in order to select the best ones. As a result, they often rely on a fixed parameter to determine the number of rewrites and ignore their quality, which can lead to the retrieval of less relevant content and the introduction of semantic noise. Since LLMs cannot inherently distinguish relevant from irrelevant information, injecting noisy context can degrade answer quality and increase the risk of hallucination. Figure 1 illustrates how inappropriate rewrite selection can lead to the retrieval of irrelevant information, ultimately resulting in incorrect answers—for example, predicting "Beatrice Ayer Patton" instead of the correct "Julia Compton Moore".

These findings highlight the need to identify high-quality rewrites that align with the original query intent. Our research aims to improve retrieval accuracy by effectively filtering out irrelevant rewrites and dynamically determining the optimal number of rewrites. Evaluating rewrite relevance typically relies on either manual annotation, which is tedious and inefficient, or LLM-based evaluations, which are computationally expensive. To address this, we propose a fast and automatic method to filter and prioritize rewrites, enhancing retrieval quality with minimal overhead.

This paper presents **ARS-RAG**, an **A**daptive **R**ewrite **S**election approach designed to enhance retrieval in open-domain QA. An overview of the approach is shown in Figure 2. ARS-RAG consists of two core components: a ranker and a selector, which collaboratively identify high-quality rewrites derived from the input query. The ranker, built on a BERT backbone, is trained via self-supervised learning to estimate the retrieval relevance of each rewrite. To generate training signals without manual annotation, we leverage LLM-based evaluation scores (e.g., from RAGAS (Es et al., 2023)) as pseudo-labels. Rather than relying on a fixed number of rewrites, ARS-RAG uses a contextual multi-armed bandit reinforcement learning method to dynamically select the optimal number of rewrites based on the query context. At inference time, ARS-RAG performs a single-pass selection using both the trained ranker and the selector, avoiding repeated retrieval or scoring operations. This design significantly improves retrieval efficiency and accuracy, while requiring no fine-tuning of the underlying rewriter.

To evaluate ARS-RAG, we conduct experiments on four ODQA benchmarks: BoolQ (Clark et al., 2019), HotpotQA (Pal et al., 2022), MedMCQA (Yang et al., 2018) and StrategyQA (Geva et al., 2021). ARS-RAG consistently outperforms strong baselines in both accuracy and efficiency.

The principal contributions of this paper are as follows:

1. We observed that query alignment inevitably introduces low-quality rewrites, leading to the retrieval of less relevant knowledge documents and degrading LLMs' response quality. This critical issue has been largely overlooked in previous research.

2. We propose ARS-RAG, which employs a contextual bandit to dynamically determine the best top-K rewrites for each query, selecting the most relevant rewrites from those scored by a self-supervised ranker.

3. We rigorously validated ARS-RAG's effectiveness against baseline methods on publicly available benchmark datasets.

2 Related Work

2.1 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) addresses the limitations of static language models by retrieving contextually relevant knowledge from external corpora, mitigating hallucinations and improving factual accuracy (Wu et al., 2023). RAG allows real-time access to updated information, making it well-suited for knowledge-intensive tasks (Ma et al., 2023; Gao et al., 2023b).

Recent extensions to RAG focus on adaptive and reflective retrieval. DeepRAG (Guan et al., 2025) decomposes queries to iteratively choose between retrieval and reasoning. MBA-RAG (Tang et al., 2024) adaptively selects retrieval strategies via bandits to balance accuracy and efficiency, while AQA (Hoveyda et al., 2024) formulates adaptive question answering as a contextual bandit problem, dynamically selecting multi-LLM communication strategies based on question complexity.

2.2 Ranker-Based RAG

Effective re-ranking is crucial for improving retrieval quality in RAG. FairRAG (Kim and Diaz, 2024) introduces a stochastic retriever for fairness, but lacks a self-supervised training scheme. Chain-of-Rank (CoR) (Lee et al., 2025) ranks document IDs using a fine-tuned LLM, simplifying inference. RankRAG (Yu et al., 2024) jointly tunes

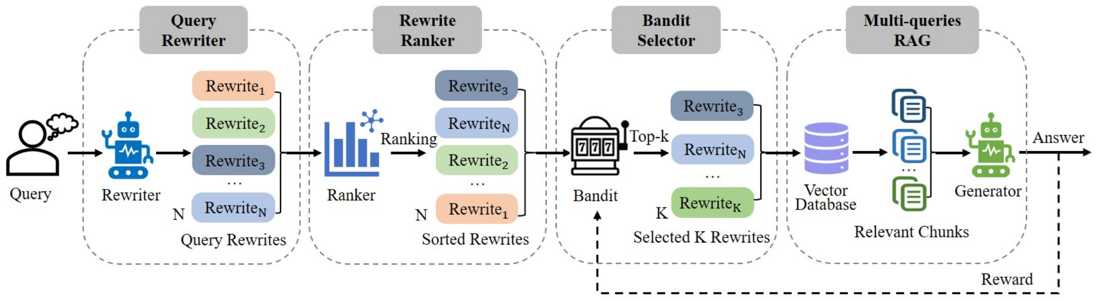


Figure 2: **ARS-RAG approach Overview.** A rewriter generates multiple rewrites, the ranker scores them based on their relevance to retrieval, and the bandit adaptively selects the optimal top-K rewrites. The selected rewrites guide retrieval to improve answer accuracy.

ranking and generation via instruction tuning. Extensive fine-tuning introduces potential biases, reduces model generality, and limits applicability in open-domain tasks. It also requires large-scale supervision and adds computational overhead, while the two-stage rerank-and-generate design further increases inference latency.

Unlike previous methods that require extensive LLMs fine-tuning, ARS-RAG employs a lightweight, self-supervised ranker and leverages a contextual bandit to dynamically select the optimal number of rewrites. This approach enables adaptive and efficient retrieval, reducing both inference latency and computational cost.

3 Method

3.1 Problem Formulation

RAG aims to enhance question answering by incorporating external knowledge from a large corpus $\mathcal{C}$. Given a query Q, we assume there exists a subset of relevant documents $R(Q)$ in the collection that provide supporting evidence for generating the correct answer:

$$R(Q) = \{d \mid d \text{ is relevant/useful to } Q\} \quad (1)$$

As this ideal set depends on many factors unknown to us, it is unrealistic to obtain it exactly. In practice, one typically approximate it with an alternative set $\hat{R}(Q)$ according to some hand-designed retrevial model $M(Q, d)$ and a threshold τ:

$$\hat{R}(Q) = \{d \mid M(Q, d) > \tau\} \quad (2)$$

In other words, the goal of RAG is to find $\hat{R}(Q)$ that approximates $R(Q)$ as close as possible. One of the major challenge for this lies in that determining the relevance of a given document could be difficult when the query Q is complex. To address this problem, a popular method (Ma et al., 2023; Mao et al., 2024) is to first rewrite the original query Q as a set of sub-queries, denoted as $\mathcal{Q} = \{q_1, \ldots, q_N\}$, where q_i is the i-th sub-query or rewrite, and then use them to approximate the needed RAG set $\hat{R}(Q)$, as follows,

$$\hat{R}(Q) = \bigcup_{i=1}^{N} \mathcal{R}(q_i), \quad q_i \in \mathcal{Q} \quad (3)$$

where $\mathcal{R}(q)$ denotes the set of retrieved documents for q.

However, this approach has two major limitations: 1) it is challenging to determine how many rewrites are sufficient for a given query - people have to set this hyperparameter value based on some heuristic ideas, and 2) there lack effective mechanisms for assessing the quality of each rewrite - importantly, not all rewrites are equally useful; many may be noisy, redundant, or even detrimental to answer quality. These challenges have been largely overlooked in existing research.

3.2 Adaptive Rewrite Selection

To address the above challenges, we propose to optimize the performance of RAG through an adaptive rewrite selection procedure. Our key idea is to first simply generate a sufficient number of rewrites (we set it to be 10 throughout our experiments) from the original query, and then select an appropriate subset from them adaptively for the generation of final relevant documents.

In particular, denote the selected set of subqueries as $\mathcal{Q}_s = \{q_{r_1}, \ldots, q_{r_K}\}$, where $q_{r_i} \in \mathcal{Q}$ and r_i is the index of the top i-th rank. In other words, $\mathcal{Q}_s$ is constructed such that it contains the K most relevant subqueries among the total rewrites of N, while K serves as a flexible threshold, dynamically adjusted per query Q. To this end, we define the final retrieval set as follows.

$$\hat{R}(Q) = \bigcup_{i=1}^{K} \mathcal{R}(q_{r_i}), \quad q_{r_i} \in \mathcal{Q}_s \tag{4}$$

As mentioned above, our objective is to construct a retrieval set $\hat{R}(Q)$ that closely approximates the ideal set $R(Q)$, thus improving the accuracy of the answer while minimizing the inclusion of irrelevant or noisy content. For this, we propose a two-stage adaptive rewrite selection approach, whose overall architecture is illustrated in Figure 2. In particular, the first stage ranks candidate rewrites according to their relevance, respectively; while the second stage simulates the perference of the user, which adaptively determines how many of the top-ranked rewrites should be retained.

More specifically, we first introduce a scoring function S, trained in a self-supervised learning manner, to estimate the relevance of each rewrite q to Q. We then formulate the selection of the optimal number K of rewrites as a contextual bandit problem, aiming to learn a policy π which automatically determines the optimal ranking threshold K based on the query embedding $\mathbf{x}$ of the original query Q, denoted as $K = \pi(\mathbf{x})$.

With this, the selected top-K rewrites form our adaptive subset: $\mathcal{Q}_s = \{q_{r_1}, \dots, q_{r_K}\}$, and the final retrieval set for query Q is constructed as:

$$\hat{R}(Q) = \bigcup_{q \in \mathcal{Q}_s} \mathcal{R}(q) \tag{5}$$

The retrieved knowledge is then passed to downstream tasks.

In what follows, we give the details of the above two-stage adaptive rewrite selection approach, which enables ARS-RAG to dynamically filter and select the most relevant rewrites for each query, yielding a flexible retrieval set $\hat{R}(Q)$ that closely approximates $R(Q)$ for better RAG.

3.3 Rewrite Ranker

A core challenge in adaptive rewrite selection is quantifying the relevance of each rewrite q to the original query Q. The motivation for our ranker is to score and rank sub-queries by their ability to retrieve knowledge that is truly useful for answering Q. We define rewrite relevance using an objective, automatic metric: the ability of a rewrite to retrieve documents that are helpful for answering Q. Intuitively, a rewrite is considered relevant if it leads to the retrieval of document that is itself relevant and informative for answering Q.

Collecting Training Data. For each input query Q, we generate a set of rewrites, denoted as $\mathcal{Q} = \{q_1, \dots, q_N\}$. For each rewrite q_i, we retrieve a set of knowledge documents $\mathcal{R}(q_i) = \{d_1, ..., d_m\}$. An automatic LLM-based evaluator $\mathcal{M}_E$, using the RAGAS (Es et al., 2023) context relevance metric, assigns a binary score $s_j = \mathcal{M}_E(Q, d_j)$ to each document $d_j \in \mathcal{R}(q_i)$ based on its relevance to the query Q. The scores for all documents $\mathcal{R}(q_i)$ are aggregated to produce a self-supervised label $y_i = \sum_{d_j \in \mathcal{R}(q_i)} s_j$, where a higher y_i indicates that q_i is more relevant to the original query Q. Each rewrite-score pair (q_i, y_i) is then paired with the input query Q to form the training data:

$$(Q, \{(q_i, y_i)\}_{i=1}^{N}) \tag{6}$$

Algorithm 1 details the procedure.

Algorithm 1 Collecting Training Data

```
1:  Require: Evaluator $\mathcal{M}_E$, Retriever $\mathcal{R}$
2:  Input: An input query $Q$
3:  Output: A tuple $(Q, \{(q_i, y_i)\}_{i=1}^{N})$
4:  Generate a set of rewrites: $\mathcal{Q} = \{q_1, ..., q_N\}$
5:  for each rewrite $q_i \in \mathcal{Q}$ do
6:      Set label: $y_i \leftarrow 0$
7:      for each document $d_j \in \mathcal{R}(q_i)$ do
8:          Evaluate score: $s_j \leftarrow \mathcal{M}_E(Q, d_j)$
9:          Update label: $y_i \leftarrow y_i + s_j$
10:     end for
11:     Save the pair $(q_i, y_i)$
12: end for
13: return $(Q, \{(q_i, y_i)\}_{i=1}^{N})$
```

Training Ranker. We train a BERT-based pairwise ranker to assess the relevance of rewrites. Given training data in the form $(Q, \{(q_i, y_i)\}_{i=1}^{N})$, we construct pairwise comparisons. For each pair (q_i, q_j) from the same query Q, we define the binary label $\bar{P}_{i,j}$:

$$\bar{P}_{i,j} = \frac{1}{2}(1 + \text{sign}(y_i - y_j)) \tag{7}$$

Each sample is represented as $(Q, q_i, q_j, \bar{P}_{i,j})$. A scoring function $\mathcal{S}(q, Q)$ computes a relevance score for each q. The probability that q_i is preferred over q_j is:

$$P_{i,j} = \sigma(\mathcal{S}(q_i, Q) - \mathcal{S}(q_j, Q)) \tag{8}$$

where σ is the sigmoid function. The ranker is trained using cross-entropy loss:

$$\mathcal{L} = -\frac{1}{N}\sum_{i,j}[\bar{P}_{i,j}\log P_{i,j}+(1-\bar{P}_{i,j})\log(1-P_{i,j})] \tag{9}$$

The trained model sorts the rewrites $\mathcal{Q}$ for each query Q, providing a relevance-based ordering for adaptive selection.

3.4 Bandit Selector

To dynamically determine the optimal top-K rewrites per query Q, we formulate K-selection as a contextual multi-armed bandit problem. This enables query-specific adaptation, balancing rewrite diversity with efficiency. We adopt NeuralUCB (Zhou et al., 2020), using neural networks to estimate the reward of each arm (a candidate K) based on the query context. As depicted in Figure 3, the bandit interacts with the RAG module through online learning. For the t-th query Q_t, the contextual bandit is defined as follows:

Context ($\mathbf{x}_t$): The BERT-encoded embedding of the query Q_t is $\mathbf{x}_t \in \mathbb{R}^d$. **Arms ($k$):** Each arm corresponds to a candidate value $k \in \{1,\dots,N\}$, where N is the maximum number of rewrites. **Action (a_t):** The selected arm $a_t \in \{1,\dots,N\}$ for Q_t at step t determines the chosen K. **Reward (r_t):** $r_t = \text{reward} \times w_{a_t}$, where reward $\in \{0,1\}$ indicates whether the answer is correct, and w_{a_t} is a monotonically decreasing function of a_t (e.g., $w_{a_t} = 3.0 - 0.2 \times a_t$). This reward formulation encourages the selection of smaller K values when possible, thus promoting both efficiency and answer quality.

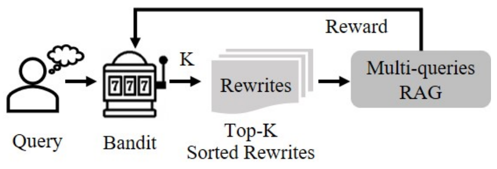


Figure 3: Bandit Selector.

Each arm k is associated with a reward prediction function $f_k(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_k)$ parameterized by a neural network, which estimates the expected reward $\hat{r}_k(\mathbf{x}_t)$ given context $\mathbf{x}_t$. Uncertainty is quantified via the gradient $g_k = \nabla_{\boldsymbol{\theta}_k} f_k(\mathbf{x}_k;\boldsymbol{\theta}_k)$ and a positive-definite matrix $\mathbf{Z}_k \in \mathbb{R}^{|\boldsymbol{\theta}_k|\times|\boldsymbol{\theta}_k|}$ capturing parameter confidence.

To select an arm, the bandit computes an upper confidence bound (UCB) score for each k:

$$\sigma_k^2(\mathbf{x}_t) = g_k(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_k)^\top \mathbf{Z}_k^{-1} g_k(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_k) \tag{10}$$

$$a_t = \underset{k\in\{1,\dots,K\}}{\arg\max}\left(f_k(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_k) + \beta_t\sqrt{\sigma_k^2(\mathbf{x}_t)}\right) \tag{11}$$

Here, $\hat{\boldsymbol{\theta}}_k$ are the estimated parameters for arm k, and $\beta_t > 0$ is the exploration parameter. $g_k(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_k)$ is the gradient of f_k with respect to $\boldsymbol{\theta}_k$ at $\mathbf{x}_t$. The term $\sqrt{g_k^\top \mathbf{Z}_k^{-1} g_k}$ quantifies the predictive uncertainty for context $\mathbf{x}_t$ according to the parameter covariance matrix $\mathbf{Z}_k$. Upon observing the reward r_t for the selected arm a_t, the bandit updates $\mathbf{Z}_{k_t}$ for a_t as follows:

$$\mathbf{Z}_{a_t} \leftarrow \mathbf{Z}_{a_t} + g_{a_t}(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_{a_t}) g_{a_t}(\mathbf{x}_t;\hat{\boldsymbol{\theta}}_{a_t})^\top \tag{12}$$

This online process allows the bandit to adaptively learn a query-specific policy $\pi(K \mid \mathbf{x})$ balancing accuracy (maximizing reward) and efficiency (minimizing K) based solely on the query context.

4 Experiments

4.1 Datasets and Metrics

To evaluate ARS-RAG across diverse domains, we conduct experiments on four ODQA datasets.

BoolQ (Clark et al., 2019) is a reading comprehension dataset with naturally occurring yes/no questions. **HotpotQA** (Yang et al., 2018) requires multi-hop reasoning over Wikipedia passages. **MedMCQA** (Pal et al., 2022) is a large-scale multiple-choice dataset for medical entrance exams. **StrategyQA** (Geva et al., 2021) targets commonsense reasoning through yes/no questions that integrate world knowledge.

Performance on BoolQ, MedMCQA, and StrategyQA is measured using Accuracy and Precision metrics, while HotpotQA is assessed with Exact Match (EM) and F1 score.

4.2 Baselines

We compare ARS-RAG with a broad range of baselines, including non-retrieval models, retrieval-augmented frameworks, and ranker-based RAG .

Non-Retrieval Baselines. These baselines rely solely on the language model's internal knowledge. We evaluate a few-shot prompting LLM and a Chain-of-Thought (CoT) (Trivedi et al., 2022) prompting method.

Models	Methods	BoolQ		HotpotQA		MedMCQA		StrategyQA		Avg.
		Acc.	Prec.	EM	F1	Acc.	Prec.	Acc.	Prec.	
	LLM	55.25	66.59	18.05	27.44	65.25	68.76	63.75	71.45	54.57
	COT	57.25	72.08	20.04	29.45	65.75	67.70	67.50	73.63	56.68
	FLARE	74.82	76.43	18.56	30.13	70.72	73.46	74.34	75.16	61.70
	SELF-RAG	77.25	80.04	18.25	29.35	71.25	74.06	81.54	82.13	64.23
Gemma2:9B	RaFe	62.86	75.45	17.28	25.94	64.32	65.26	78.02	79.25	58.55
	CoR	76.78	78.92	20.57	29.34	70.50	73.68	77.62	80.75	63.52
	FairRAG	73.64	75.58	19.63	31.75	72.88	73.42	81.00	83.37	63.91
	RankRAG	78.12	79.84	21.40	33.18	71.78	72.38	82.25	85.07	65.50
	ARS-RAG	**80.25**	**81.25**	**29.37**	**40.43**	**75.75**	**77.12**	**84.75**	**84.85**	**69.22**
	LLM	65.25	76.67	16.03	26.50	71.50	71.88	73.62	78.24	59.96
	COT	70.25	80.09	24.05	34.98	70.54	71.89	78.06	79.68	63.69
	FLARE	79.41	82.25	27.46	38.85	79.62	83.12	77.37	78.53	68.33
	SELF-RAG	82.25	85.33	26.54	40.09	83.52	84.08	87.58	87.86	72.16
Qwen2.5:32B	RaFe	64.62	70.26	29.47	40.89	82.46	84.52	83.56	84.26	67.51
	CoR	81.50	85.25	28.00	40.86	83.75	84.07	89.50	90.31	72.91
	FairRAG	78.64	83.52	26.75	40.46	81.02	81.66	85.75	86.85	70.58
	RankRAG	81.58	84.82	27.73	41.28	82.57	82.84	90.00	90.51	72.67
	ARS-RAG	**85.04**	**87.53**	**35.08**	**45.23**	**85.75**	**86.06**	**92.25**	**92.83**	**76.22**
	LLM	72.56	76.99	28.25	42.06	85.50	86.04	75.56	77.91	68.11
	COT	78.08	81.17	32.75	42.62	84.54	85.04	83.06	83.07	71.29
	FLARE	84.92	85.74	34.62	43.82	85.60	87.23	79.25	81.61	72.85
	SELF-RAG	85.00	87.32	35.75	48.89	86.75	86.96	89.46	92.60	76.59
Llama3.3:70B	RaFe	82.46	86.25	31.54	43.75	87.43	88.16	85.74	86.17	73.94
	CoR	84.48	85.49	35.78	50.34	88.02	88.15	91.85	91.93	77.01
	FairRAG	82.62	85.25	31.42	46.25	86.83	86.96	87.52	87.86	74.34
	RankRAG	84.72	86.14	34.78	53.68	87.11	87.75	92.26	92.12	77.32
	ARS-RAG	**88.75**	**89.82**	**43.50**	**55.96**	**89.75**	**89.78**	**94.50**	**94.58**	**80.83**

Table 1: **Performance comparison.** The Avg. column indicates overall performance and the Time column reflects response time. The **best** and second-best results are **bolded** and underlined.

Retrieval-Augmented Baselines. These methods retrieve external knowledge to assist generation. FLARE (Jiang et al., 2023) dynamically determines when and what to retrieve during generation. SELF-RAG (Asai et al., 2023) combines on-demand retrieval with self-reflection to refine outputs and improve coherence.

Ranker-Based RAG Baselines. These approaches use ranking to select the most relevant contexts or rewrites. RaFe (Mao et al., 2024) improves query rewriting for RAG by leveraging reranker feedback to train models without annotations. CoR (Lee et al., 2025) simplifies the ranking process by prioritizing document reliability. FairRAG (Kim and Diaz, 2024) introduces a stochastic retriever to promote equal exposure of relevant contexts. RankRAG (Yu et al., 2024) facilitates context ranking and answer generation through the instruction-tuning of LLMs.

4.3 Main Results

Table 1 reports the performance of ARS-RAG and various baselines across four ODQA benchmarks and three model scales. ARS-RAG consistently achieves the best average performance (80.83) on `Llama3.3:70B`, outperforming strong ranker-based methods such as RankRAG (77.32) and CoR (77.01). Notably, ARS-RAG attains the highest score on StrategyQA (94.50), underscoring its ability to select and utilize high-quality rewrites for challenging reasoning tasks.

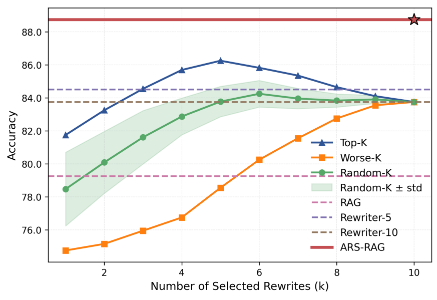
Across all benchmarks and model sizes, performance improves as model parameters increase, with `Llama3.3:70B` obtaining the highest results on each dataset. The effectiveness of ARS-RAG is consistent across scales, with clear gains observed not only for the largest model but also for `Gemma2:9B` and `Qwen2.5:32B`, demonstrating strong scalability and generalization.

In summary, ARS-RAG outperforms both non-retrieval and retrieval-augmented baselines, while also reducing computational overhead compared to RAG-based ranker pipelines. These results highlight the effectiveness of combining adaptive rewrite selection and relevance-based ranking in RAG pipelines, eliminating the need for LLM fine-tuning. This combination enhances relevance and reduces computational cost.

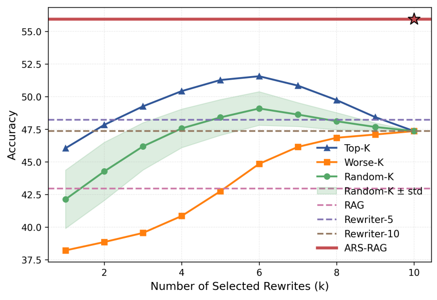
4.4 Effect of Rewrite Selection Strategies

We assess the impact of different rewrite selection strategies on BoolQ and HotpotQA. The compared strategies are as follows:

RAG: Utilizes the input query for retrieval. **Rewriter-5/10:** Generates 5 or 10 rewrites per query for targeted knowledge retrieval. **Top-K:** Employs the K most relevant rewrites as ranked by the ranker. **Worse-K:** Uses the K least relevant rewrites as ranked by the ranker. **Random-K:** Samples K rewrites randomly (averaged over 5 runs). **ARS-RAG (Adaptive-K):** Adaptively selects the best K rewrites per query.



(a) BoolQ



(b) HotpotQA

Figure 4: Comparison of rewrite selection strategies. ARS-RAG achieves the best overall performance.

Figure 4a and Figure 4b illustrate the comparative performance of these strategies. The substantial gap between Top-K and Worse-/Random-K validates the effectiveness of the learned ranker: prioritizing rewrites based on relevance significantly improves performance. However, Top-K also reveals a limitation: its accuracy is sensitive to the choice of K, often peaking around $K = 5$ (on BoolQ) or $K = 6$ (on HotpotQA) and declining as K increases due to noisy or redundant rewrites.

In contrast, ARS-RAG surpasses the peak performance of Top-K without requiring manual tuning. By leveraging a contextual bandit, ARS-RAG adaptively determines the optimal number of rewrites for each query. This flexibility allows the model to dynamically balance the trade-off between information coverage and noise, leading to robust and high performance.

Overall, these results highlight two key findings: 1) a trained ranker is crucial for high-quality rewrite selection, and 2) adaptive determination of K via bandit learning further improves effectiveness. The combination of these approaches, as realized in ARS-RAG, leads to superior retrieval-augmented QA performance across datasets.

4.5 Ablation Study

To assess the contribution of each component in ARS-RAG, we conduct an ablation study across four ODQA datasets, as shown in Table 2. The ablation settings are as follows:

LLM: Directly answers the question using LLMs. **RAG:** Retrieves external knowledge using the original query. **Rewriter:** Generates 10 rewrites per query, each used for retrieval. **Rewriter+Ranker:** Generates 10 rewrites, ranks them by relevance, and selects the top-5 most relevant rewrites for retrieval. **Rewriter+Bandit:** Uses a bandit to adaptively determine K, then generates K rewrites directly for retrieval, without ranking. **ARR-RAG:** Combines all components; generates 10 rewrites, ranks them, and uses a bandit to adaptively select the optimal K top-ranked rewrites for retrieval.

Variant	BoolQ	HotpotQA	MedMCQA	StrategyQA
LLM	72.56	42.06	85.50	75.56
RAG	79.23	43.67	86.52	83.74
Rewriter	83.75	47.36	87.50	90.75
Rewriter+Ranker	86.25	50.57	88.75	93.50
Rewriter+Bandit	85.42	48.45	88.23	91.88
ARS-RAG	**88.75**	**55.96**	**89.75**	**94.50**

Table 2: Ablation Study Results.

Performance improves steadily from the LLM baseline to RAG and Rewriter, confirming the value of external knowledge and query reformulation. Introducing the ranker (Rewriter+Ranker) further improves accuracy by selecting the most relevant rewrites, but is constrained by a fixed K, which can limit performance on diverse queries due to either redundancy or insufficient coverage.

The Rewriter+Bandit variant adaptively determines K for each query. While this reduces inference cost and adapts to query complexity, it lacks quality control: rewrites are directly used without ranking, making it susceptible to semantic drift and retrieval noise, especially on complex datasets like HotpotQA.

By integrating both ranking and adaptive selection, ARS-RAG achieves the best overall performance. The ranker ensures high-quality rewrites, while the bandit dynamically adjusts K, leading to robust and consistent gains across all datasets. This ablation study underscores the necessity of jointly optimizing rewrite quality and quantity for retrieval-augmented QA.

4.6 Effectiveness of the Bandit-based Rewrite Selection

To assess the effectiveness of the bandit technique in adaptive rewrite selection, we conduct 500 training epochs with a batch size of 32, evaluating policy performance on the test set after each epoch, and recording the weighted reward. All results are averaged over five independent runs, and both mean and standard deviation (std) of weighted reward are reported for train and test sets.

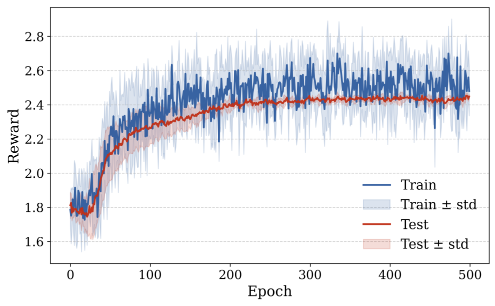


Figure 5: Mean and standard deviation of weighted reward on training and test sets in the bandit selector.

Figure 5 illustrates the reward curves throughout training. The steady increase in test reward, accompanied by a gradual decrease in standard deviation, indicates that the bandit selector effectively learns a stable and robust policy over time. Notably, the narrowing std suggests the learned policy generalizes well and converges reliably across different runs.

To further analyze the learned policy, we examine the distribution of selected K values on the test set using the bandit policy from the final training epoch (Figure 6). The distribution shows a clear preference for smaller K values, suggesting that the bandit selector learns to minimize retrieval redundancy and suppress irrelevant information, while still preserving answer quality. The preference for lower K values demonstrates that the reward formulation effectively encourages efficient and adaptive retrieval.

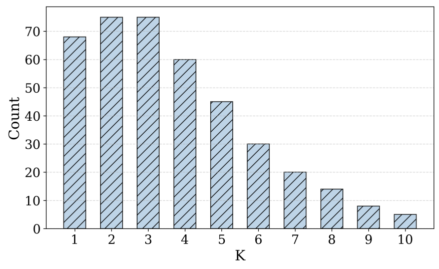


Figure 6: Distribution of K for last training epoch on the test set.

In summary, these results demonstrate that the bandit selector can efficiently learn a query-adaptive K selection policy, achieving a favorable trade-off between answer quality and resource consumption.

5 Conclusion

In this paper, we presented ARS-RAG, an adaptive rewrite selection approach for open-domain QA. ARS-RAG generates multiple rewrites and dynamically selects the most effective ones. A self-supervised ranker evaluates the relevance of each rewrite, while a contextual bandit selector adaptively chooses the optimal subset for retrieval. This design enables query-specific adaptation and precise knowledge retrieval, all without requiring additional fine-tuning of the rewriter. Extensive experiments on four ODQA benchmarks show that ARS-RAG consistently outperforms both non-retrieval and RAG-based baselines across multiple model sizes. With its single-pass design and low computational overhead, ARS-RAG offers a scalable and efficient solution for knowledge-intensive applications.

6 Limitations

While ARS-RAG shows strong empirical performance, several limitations remain. First, our ranker relies on context relevance scores generated by the RAGAS framework, which in turn depends on an LLM-based evaluator. As a result, biases or limitations inherent in the evaluator's training distribution may affect data quality and overall system performance. Second, the pairwise ranking objective focuses on relative comparisons. This may penalize novel but valid rewrites that deviate from the original query and may overlook interdependencies among multiple rewrites. Finally, our experiments are limited to general-domain ODQA benchmarks. The approach's robustness under domain shift, adversarial inputs, or larger retrieval corpora has yet to be fully explored.

7 Ethical Considerations

All datasets used in this work are publicly available and widely adopted in the ODQA community. We do not collect or annotate any private or sensitive data. As our method builds on pretrained LLMs and LLM-based evaluators, it may reflect certain biases present in these models. While ARS-RAG improves retrieval precision and efficiency, it does not explicitly filter harmful content in the retrieved documents. The adaptive rewrite mechanism may also amplify factual inconsistencies if used with unreliable retrievers or generators.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

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*.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *arXiv preprint arXiv:1905.10044*.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2023. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023a. Precise zero-shot dense retrieval without relevance labels. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023b. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

Xinyan Guan, Jiali Zeng, Fandong Meng, Chunlei Xin, Yaojie Lu, Hongyu Lin, Xianpei Han, Le Sun, and Jie Zhou. 2025. Deeprag: Thinking to retrieval step by step for large language models. *arXiv preprint arXiv:2502.01142*.

Mohanna Hoveyda, Arjen P de Vries, Maarten de Rijke, Harrie Oosterhuis, and Faegheh Hasibi. 2024. Aqa: Adaptive question answering in a society of llms via contextual multi-armed bandit. *arXiv preprint arXiv:2409.13447*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.

To Eun Kim and Fernando Diaz. 2024. Towards fair rag: On the impact of fair ranking in retrieval-augmented generation.

Juntae Lee, Jihwan Bang, Seunghan Yang, Kyuhong Shim, and Simyung Chang. 2025. Chain-of-rank: Enhancing large language models for domain-specific rag in edge device. *arXiv preprint arXiv:2502.15134*.

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.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *arXiv preprint arXiv:2305.14283*.

Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Huajun Chen, and Ningyu Zhang. 2024. Rafe: ranking feedback improves query rewriting for rag. *arXiv preprint arXiv:2405.14431*.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Conference on health, inference, and learning*, pages 248–260. PMLR.

Xiaqiang Tang, Qiang Gao, Jian Li, Nan Du, Qi Li, and Sihong Xie. 2024. Mba-rag: a bandit approach for adaptive retrieval-augmented generation through question complexity. *arXiv preprint arXiv:2412.01572*.

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

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. *arXiv preprint arXiv:2303.07678*.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *arXiv preprint arXiv:1809.09600*.

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.

Dongruo Zhou, Lihong Li, and Quanquan Gu. 2020. Neural contextual bandits with ucb-based exploration. *Preprint*, arXiv:1911.04462.

A Dataset Details

We conduct experiments on four widely used ODQA benchmarks:

BoolQ (Clark et al., 2019) is a reading comprehension dataset consisting of naturally occurring yes/no questions. These questions are non-factoid and often require entailment-like inference over passages, making them challenging for models trained on traditional QA data.

HotpotQA (Yang et al., 2018) focuses on multi-hop reasoning, where each question requires aggregating information from multiple Wikipedia passages. The dataset also provides supporting sentence-level evidence, enabling supervision for both answer prediction and explainability.

MedMCQA (Pal et al., 2022) is a large-scale multiple-choice QA dataset based on real medical entrance exam questions. It covers 21 medical subjects and demands fine-grained reasoning across diverse medical topics.

StrategyQA (Geva et al., 2021) is a common-sense reasoning benchmark where questions embed implicit multi-step reasoning strategies. Each instance includes decompositions into reasoning steps and evidence paragraphs, testing a system's ability to infer hidden logical structure.

For each dataset, we randomly sample 1,600 examples for training and 400 for testing.

B Implementation Details

LLM Selection. We use `LLaMA3.3:70B` as both the query rewriter and the context relevance evaluator, leveraging its strong instruction-following and text generation capabilities to ensure high-quality rewrites and accurate relevance scoring. For answer generation, we experiment with three models: `Gemma2:9B`, `Qwen2.5:32B`, and `LLaMA3.3:70B`. For ranker training, we use `BERT-base-uncased` as the encoder for BoolQ, HotpotQA, and StrategyQA, and `ClinicalBERT` for MedMCQA to better align with the medical domain.

Retriever. We use the `nomic-embed-text` embedding model for dense retrieval, with embeddings indexed in Chroma. Retrieval is based on vector similarity, with top-5 documents selected by default.

Retrieval Contexts. For each query, the selected rewrites are independently used to retrieve supporting documents. All retrieved contexts are then merged and deduplicated to eliminate overlapping or redundant content. This ensures that the generator receives a diverse yet non-repetitive set of evidence, which helps reduce semantic noise and improves the quality and factuality of the generated answers.

Ranker Training Details. The model is optimized with AdamW (learning rate 5e-5, weight decay 0.01), using BCEWithLogits loss. We train for 100 epochs with a batch size of 64, using cosine learning rate scheduling. Dropout (rate 0.1) is applied in the classification head for regularization.

Bandit Selector Reward Setting. We adopt different reward schemes for different datasets. For BoolQ, MedMCQA, and StrategyQA, we use binary rewards: the reward is 1 if the answer is correct and 0 otherwise. For HotpotQA, we apply a mixed standard: if EM equals 1, the reward is set to 1; if EM is 0, the F1 score is used as the reward. As F1 scores are within [0, 1], the overall reward for HotpotQA is also bounded in [0, 1]. To encourage the bandit to select smaller k for efficiency, we adopt a weighted reward scheme in all datasets: the observed reward is multiplied by a monotonically decreasing weight w_k (e.g., $w_k = 3.0 - 0.2k$), i.e., weighted_reward $=$ reward $\times$ w_k. This formulation penalizes the use of larger k values, guiding the bandit to prefer more efficient selections without sacrificing answer quality.

C Penalty Function in Bandit Selector

Weighted Reward vs. Penalty-based Schemes. In the main paper, we adopt a weighted reward function that encourages the bandit to select smaller k values, promoting both efficiency and answer quality. To further investigate the impact of alternative penalty designs, we also experimented with an explicit penalty term, $\lambda \cdot k$, added to the reward.

λ	0	0.01	0.02	0.03	0.04
Accuracy	88.2	88.5	88.63	88.1	87.3
Mean-k	5.7	4.8	4.2	2.9	2.5

Table 3: Effect of penalty parameter λ on accuracy and average k (evaluated on BoolQ).

As shown in Table 3, varying the penalty parameter λ demonstrates a trade-off between efficiency and accuracy. Without penalty ($\lambda = 0$), the bandit often selects larger k, resulting in moderate accuracy. Introducing a moderate penalty ($\lambda = 0.01$ or 0.02) reduces average k and improves accuracy, as the penalty discourages redundant or irrelevant rewrites and suppresses retrieval noise. However, overly large λ leads to very small k values, reducing coverage of effective rewrites and overall accuracy. These results highlight that a properly tuned penalty promotes both efficiency and answer quality.

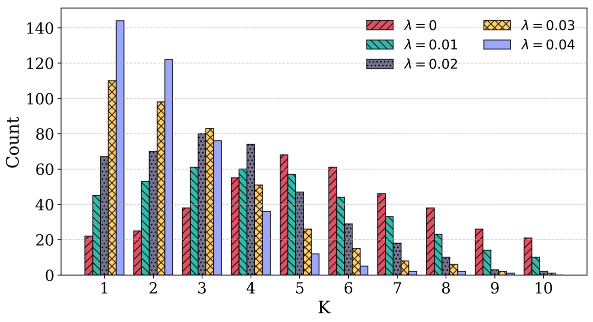


Figure 7: Distribution of k for Different λ.

Figure 7 further visualizes the distribution of selected k values across different λ settings. As λ increases, the bandit is increasingly incentivized to choose smaller k, resulting in a leftward shift in the distribution. However, overly large penalties may result in under-exploration and suboptimal answer quality. Based on these findings, we use the weighted reward design in the main text, as it yields higher accuracy while still maintaining efficiency.

D Time consumption

We evaluate the inference latency of two RAG pipelines: the Original pipeline and the ARS-RAG pipeline.

The Original pipeline generates 10 rewrites per query, retrieves 5 documents per rewrite, and uses a binary LLM-based evaluator to score them. The top-5 rewrites are then selected and passed to the Multi-queries RAG module. This multi-step process incurs substantial latency due to repeated retrieval and evaluation.

In contrast, ARS-RAG generates 10 rewrites but leverages a lightweight ranker to score them. A bandit selector then adaptively determines the optimal number of rewrites to use based on the query. This adaptive design eliminates redundant per-rewrite retrieval and scoring.

As shown in Figure 8, ARS-RAG consistently achieves lower latency across all datasets. Its one-

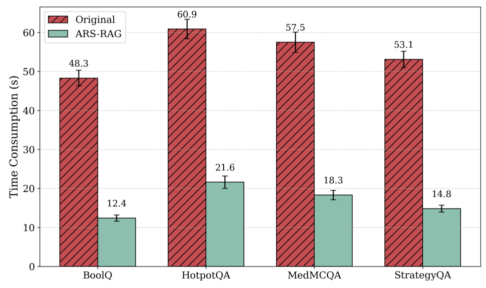


Figure 8: Comparison of Time Consumption.

pass ranking and adaptive selection strategy enables faster inference and easier deployment.

E Prompt Templates

We list below all prompt templates used throughout our approach. Each prompt is tailored to guide the LLMs in a specific subtask, including query rewriting, context relevance evaluation, and answer generation under different settings. All prompts follow a modular structure with a role declaration, behavioral rules, and clearly defined inputs.

Query Rewriting. This prompt guides the LLM to produce ten diverse, semantically faithful rewrites for a given input question, enhancing retrieval coverage without sacrificing precision. See Table 4.

Context Relevance Evaluation. This prompt is used to judge whether a retrieved document is semantically relevant to the original question. It provides binary supervision (“relevant” or “irrelevant”) for training and evaluating the ranker. See Table 5.

Zero-Context Answer Generation. This prompt asks the LLM to determine whether a given question is true or false without any context. It serves as a baseline to isolate model performance from retrieved evidence. See Table 6.

RAG Answer Generation. This prompt is used in retrieval-augmented settings. Given a question and supporting context, the LLM is instructed to answer “true” or “false” and optionally produce reasoning steps. See Table 7.

Query Rewriting Prompt
Role: As an AI assistant, I am responsible for rewriting the original question to enhance its clarity, diversity, and contextual relevance. **# Rules:** 1. I will generate ten diverse rewrites by systematically reformulating the input question. 2. Each rewrite will maintain semantic fidelity, adhere to formal language standards, and avoid ambiguity or redundancy. 3. The rewrites will be clearly numbered, grammatically correct, and contextually meaningful. 4. Variations may include paraphrasing, structural reformulation, or expansion/compression where appropriate. 5. All rewrites will be separated by line breaks to ensure readability. **# Input:** Original question: {question}

Table 4: Query Rewriting Prompt.

Context Relevance Evaluation Prompt
Role: You are an AI assistant responsible for evaluating whether the provided context is relevant to the given question. **# Rules:** 1. If the context directly answers the question or includes key information or data that can reasonably help answer it, respond with "relevant". 2. If the context is unrelated to the question, does not provide sufficient information to answer it, or lacks key details, respond with "irrelevant". 3. Respond using only the word "relevant" or "irrelevant". Do not include any additional text or characters. This ensures clarity and precision. **# Input:** Question: {question} Context : {context}

Table 5: Context Relevance Evaluation Prompt.

Generation without Retrieval Prompt
Role: You are an AI Q&A assistant. I will prepare questions for you, and your role is to help me think through the question and arrive at the correct answer. **# Rules:** 1. Determine whether the following question is true or false. 2. Your response must be formatted as either "true" or "false". 3. Do not include any additional text or characters in your response. **# Input:** Question: {question}

Table 6: Generation without Retrieval Prompt.

Generation with Retrieval Prompt
Role: You are an AI Q&A assistant. I will prepare questions for you, and your role is to help me think through the question and arrive at the correct answer. **# Rules:** 1. Based on the provided question and context, please provide a clear, step-by-step reasoning process leading to your answer, answering the following question is true or false. 2. Ensure that your explanation is logical, accurate, and directly relevant to the question. 2. Your response must be formatted as either "true" or "false". 3. Do not include any additional text or characters in your response. **# Input:** Question: {question} Context : {context}

Table 7: Generation with Retrieval Prompt.