R^{3}AG: Retriever Routing for Retrieval-Augmented Generation

Tong Zhao[1], Yutao Zhu[1]*, Yucheng Tian[2], Zhicheng Dou[1]
[1]Renmin University of China
[2]South China University of Technology
zhaotong7@ruc.edu.cn, yutaozhu94@gmail.com

Abstract

Retrieval-augmented generation (RAG) has become a cornerstone for knowledge-intensive tasks. However, the efficacy of RAG is often bottlenecked by the "one-size-fits-all" retrieval paradigm, as different queries exhibit distinct preferences for different retrievers. While recent routing techniques attempt to select the optimal retriever dynamically, they typically operate under a "single and static capability" assumption, selecting retrievers solely based on semantic relevance. This overlooks a critical distinction in RAG: a retrieved document must not only be relevant but also effectively support the generator in producing correct answers. To address this limitation, we propose R^{3}AG, a novel routing framework that explicitly models the dynamic alignment between queries and retriever capabilities. Unlike previous approaches, R^{3}AG decomposes retriever capability into two learnable dimensions: retrieval quality and generation utility. We employ a contrastive learning objective that leverages complementary supervision signals, *i.e.*, document assessments and downstream answer correctness, to capture query-specific preference shifts. Extensive experiments on several knowledge-intensive tasks show that R^{3}AG consistently outperforms both the best individual retrievers and state-of-the-art static routing methods.

1 Introduction

The rapid progress of large language models (LLMs) has substantially advanced the state-of-the-art in natural language processing(NLP), playing an increasingly important role in a wide range of real-world applications (Vaswani et al., 2017; Radford et al., 2018; Brown et al., 2020; Bubeck et al., 2023; Zhao et al., 2023). Despite these impressive capabilities, LLMs are inherently constrained by their static and finite training corpora. Consequently, they frequently suffer from critical limitations such as hallucinations and inadequate timeliness, particularly when confronted with knowledge-intensive tasks (Ji et al., 2023; He et al., 2023; OpenAI, 2024). In response to these challenges, retrieval-augmented generation (RAG) (Lewis et al., 2020; Guu et al., 2020) has emerged as the *de facto* gold standard. By incorporating external evidence during inference, RAG effectively bridges the gap between the static training corpus of LLMs and the dynamic, knowledge-intensive characteristics of real-world queries (Cai et al., 2022; Gao et al., 2023; Izacard and Grave, 2021; Singh et al., 2021; Hindi et al., 2025).

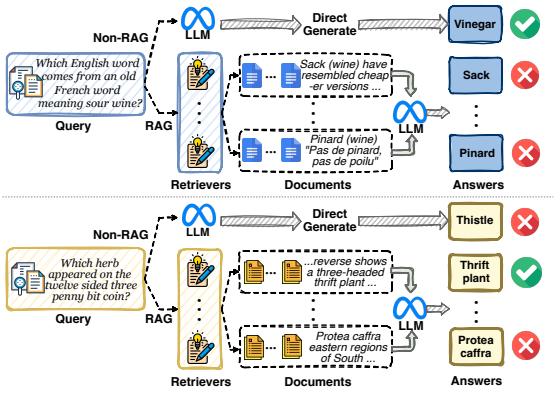

Figure 1: (Top) Retrieval is not universally beneficial. (Bottom) Different retrievers yield different generation results of varying quality.

However, the efficacy of RAG is heavily contingent on its retrieval component, which remains a significant bottleneck. In practice, most systems adopt a "one-size-fits-all" approach, utilizing a fixed retriever regardless of the input query. This strategy is suboptimal for two primary reasons. On the one hand, retrieval is not always necessary (Petroni et al., 2019): when the model's internal parametric knowledge is sufficient, retrieving additional evidence may introduce noise and degrade generation performance, as shown in Fig-

*Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 20506–20522

ure 1. On the other hand, extensive research reveals that no single retriever outperforms all others across all query types (Lee et al., 2025; Kim and Diaz, 2025). For example, sparse retrievers often excel at precise entity matching, whereas dense retrievers are superior at capturing semantic nuance. Therefore, the optimal retriever choice is highly query-dependent. This necessitates a move towards adaptive retrieval strategies and motivates our core research question: *How can we effectively route queries to the most suitable retriever under the RAG paradigm?*

Recently, several studies have attempted to design query-oriented routers to address this challenge (Peng et al., 2025; Hsu et al., 2025; Lee et al., 2025; Kim and Diaz, 2025). These approaches generally model retriever capability through a single lens, such as semantic relevance, and assume a fixed performance profile for each retriever. For example, some methods simply determine whether to invoke a text or table retriever based solely on the data type required by the query (Peng et al., 2025). While promising, such methods overlook the dual nature of the RAG objective: retrieved documents must not only be relevant (i.e., achieve high retrieval quality) but also utility-driven for the generator to ensure strong generation performance. In other words, a document deemed relevant by standard IR metrics may still lead to erroneous generation if it contains conflicting information or lacks the specific reasoning pattern required by the LLM. Consequently, routing strategies that neglect the complex interplay between retrieval quality and generation utility fail to adapt to the dynamic requirements of RAG scenarios.

To address this problem, we propose R^{3}AG, a Retriever Routing method for RAG. R^{3}AG is a contrastive learning-based framework explicitly designed to model the dynamic alignment between queries and retriever capabilities. Unlike prior methods that treat retrievers as black boxes with static scores, R^{3}AG decomposes retriever capability into two learnable dimensions: *retrieval quality*, representing the ability to find high-quality evidence, and *generation utility*, representing the ability to support accurate downstream answer generation. At the architectural level, R^{3}AG employs a multi-head attention mechanism to fuse these capability dimensions under the guidance of the input query, producing a hybrid representation that captures the query-specific preferences of different retrievers. For optimization, we devise a contrastive learning objective where positive and negative samples are constructed based on two complementary supervision signals: the quality of the retrieved documents and the correctness of downstream answer generation. This objective prompts the alignment between queries and retrievers that can simultaneously provide high-quality evidence and enable correct answer generation, thereby effectively guiding the router to prioritize end-to-end RAG performance over simple relevance matching.

We conduct a comprehensive suite of knowledge-intensive tasks (Joshi et al., 2017; Kwiatkowski et al., 2019; Yang et al., 2018) of R^{3}AG. Experimental results demonstrate that R^{3}AG consistently outperforms both the best individual retriever and state-of-the-art static routing methods. These results validate the superiority of dynamic routing and the critical role of generation utility in RAG.

Our main contributions are threefold:

(1) We identify the limitations of the single and static capability assumption in existing methods, highlighting the necessity of distinguishing between retrieval relevance and generation utility.

(2) We propose R^{3}AG, a novel routing framework that explicitly decomposes retriever capability into learnable retrieval quality and generation utility embeddings. By leveraging contrastive learning, R^{3}AG effectively captures dynamic and query-specific retriever preferences.

(3) Extensive experiments on knowledge-intensive tasks demonstrate that R^{3}AG successfully exploits retriever strengths, consistently outperforming both the best individual retrievers and state-of-the-art static routing methods.

2 Related Work

RAG RAG effectively mitigates the knowledge limitations of LLMs by incorporating external evidence at inference time (Lewis et al., 2020; Gao et al., 2023). To enhance RAG performance, prior research has focused on optimizing specific components, such as refining the retriever (Yu et al., 2023), or improving the generator's utilization of context through query rewriting (Ma et al., 2023; Peng et al., 2024), re-ranking (MacAvaney et al., 2020), and context compression (Li et al., 2025). More recently, the field has witnessed a shift towards diverse retrieval capabilities, including hybrid indexing strategies (Yan et al., 2025) and multi-perspective retrieval paradigms (Zhang et al., 2025c). This increasing heterogeneity in retrieval

methods reveals a critical opportunity: rather than relying on a single retriever, systems can benefit from ensemble approaches that strategically leverage multiple retrievers. This motivation underscores our focus on the retriever routing problem within RAG settings.

Routing Techniques Routing mechanisms have been extensively explored to improve model efficiency and performance. In the context of LLMs, routing primarily involves selecting domain-specific experts or models, utilizing either jointly trained routers (Sukhbaatar et al., 2024; Muqeeth et al., 2024) or post-hoc techniques based on domain signals (Feng et al., 2024; Belofsky, 2023; Shen et al., 2024). In the realm of information retrieval, routing is often framed as federated search, where sub-queries are assigned to expert retrievers or domain-specific indices (Lin et al., 2023; Lee et al., 2025). Within RAG specifically, recent efforts have investigated adaptive strategies, such as determining whether to retrieve (Mallen et al., 2023; Jeong et al., 2024) or routing between different LLMs (Zhang et al., 2025a). However, these approaches typically operate under a static assumption or assess retrievers solely based on retrieval relevance. They overlook a crucial factor in RAG: a retriever's value should be measured by its utility in supporting downstream generation. To bridge this gap, our proposed R^{3}AG explicitly models both retrieval quality and generation utility, enabling more effective query routing in RAG scenarios.

3 Problem Formulation

RAG improves LLMs by incorporating external knowledge via a external supplemental framework. Given a query q and an external corpus $\mathcal{D}$, a retriever R retrieves a set of relevant documents $d = R(\mathcal{D}, q)$ from the corpus. Subsequently, a generator M produces a response y conditioned on both the query and the retrieved documents, formulated as $y = M(q, d)$.

We focus on the problem of *retriever routing* within the RAG framework. Let $\mathcal{R} = \{R_1, \ldots, R_K\}$ denote a pool of K candidate retrievers. To account for scenarios where external knowledge is unnecessary or potentially detrimental, we introduce a special null retriever R_0, which returns an empty document set, *i.e.*, $R_0(\mathcal{D}, q) = \varnothing$. In this case, the generation reduces to the standard parametric setting $y = M(q)$, making the non-RAG setting a special case of our framework.

The goal of **retriever routing** is to select the optimal retriever index for a given query. We define a routing policy $\pi : \mathcal{Q} \to \{0, 1, \ldots, K\}$ that maps a query q to an index i. If $\pi(q) = i > 0$, the i-th retriever is invoked; if $\pi(q) = 0$, no retrieval is performed.

To evaluate the quality of a routing decision, we introduce a utility scoring function $s(y, y^*) \in [0, 1]$, which measures the quality of the generated response y against a reference answer y^*. Unlike standard IR metrics that measure retrieval relevance, this function explicitly quantifies the *generation utility* of a retriever. The objective of the routing policy is to maximize the expected generation performance across the query distribution:

$$\max_{\pi} \ \mathbb{E}_{q \sim \mathcal{Q}} \Big[s\big(M(q, R_{\pi(q)}(\mathcal{D}, q)), y^*\big) \Big]. \quad (1)$$

This formulation implies that the optimal router must dynamically adapt to query needs, selecting a specific retriever or bypassing retrieval entirely to maximize the final answer quality.

4 Methodology

We introduce R^{3}AG, a query-adaptive routing framework for RAG scenarios. As introduced in Section 1, effective routing requires modeling not just retrieval **relevance** but also the **utility** of retrieved documents for downstream generation. To capture this duality, R^{3}AG disentangles retriever capability into two distinct dimensions: *Retrieval Quality* and *generation utility*.

The overall framework is illustrated in Figure 2. We first describe the architecture of R^{3}AG, which consists of dual-encoder capability modeling, query-conditioned capability fusion, and similarity-based routing (Section 4.1). Subsequently, we detail our two-stage optimization strategy, which employs contrastive learning to align these capability representations with supervision signals derived from both document assessments and generation outcomes (Section 4.2).

4.1 Architecture

The architecture of R^{3}AG is designed to transform raw inputs into actionable routing decisions through a structured pipeline. Specifically, the model first maps the query and candidates into a disentangled capability space, then fuses these representations based on the query's specific needs, and finally computes a similarity score for routing.

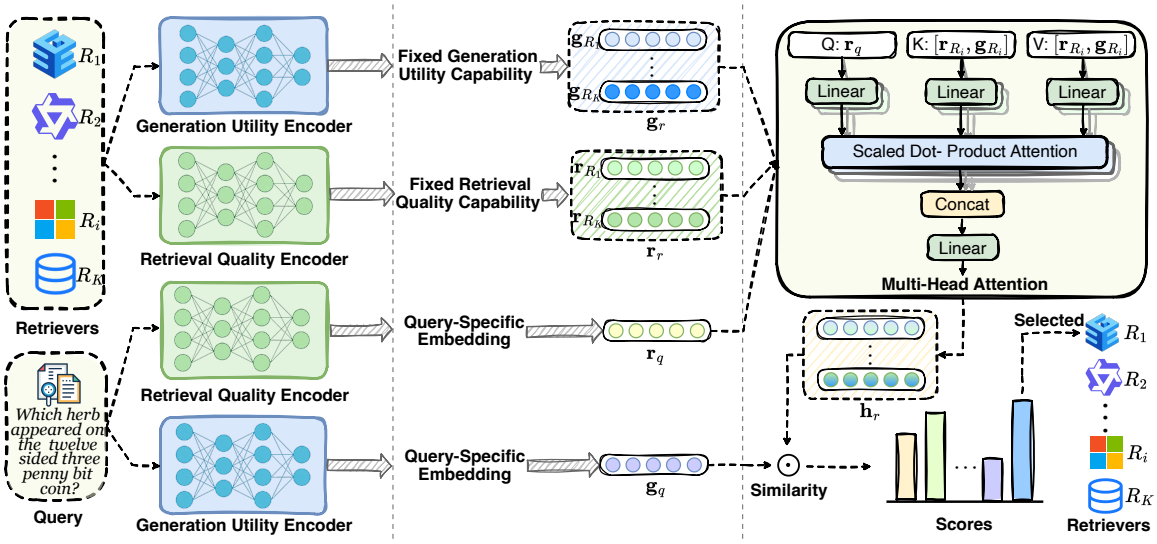

Figure 2: R^{3}AG models retriever capability from retrieval quality and generation utility, and applies query-adaptive multi-head attention to select the most suitable retriever for each query.

This process comprises three key components: capability representation, capability fusion, and the final routing decision.

Capability Representation We construct a representational space where both queries and retrievers are characterized by two complementary factors: (1) **Retrieval Quality (r)**, which captures intrinsic retrieval properties such as relevance, coverage, and redundancy; and (2) **generation utility (g)**, which reflects the potential to support correct downstream answers.

To implement this, R^{3}AG employs two shared encoders: a retrieval quality encoder ϕ_r and a generation utility encoder ϕ_g. Given a query q and a retriever R_i, we obtain their respective capability embeddings as:

$$\begin{aligned} \mathbf{r}_q &= \phi_r(q), \qquad \mathbf{g}_q = \phi_g(q), \\ \mathbf{r}_{R_i} &= \phi_r(R_i), \quad \mathbf{g}_{R_i} = \phi_g(R_i). \end{aligned} \tag{2}$$

Here, the pair $\{\mathbf{r}_{R_i}, \mathbf{g}_{R_i}\}$ constitutes the comprehensive capability representation for the retriever R_i. Crucially, encoders of the same type share parameters across queries and retrievers to ensure a consistent metric space.

Capability Fusion Since different queries prioritize different retriever capabilities, a static combination of $\mathbf{r}_{R_i}$ and $\mathbf{g}_{R_i}$ is insufficient. R^{3}AG employs a multi-head attention mechanism to dynamically fuse these capabilities conditioned on the query's retrieval intent $\mathbf{r}_q$. Formally, the fused retriever representation $\mathbf{h}_{R_i}$ is computed as:

$$\mathbf{h}_{R_i} = \mathrm{MHA}(\mathbf{r}_q,\ [\mathbf{r}_{R_i}, \mathbf{g}_{R_i}],\ [\mathbf{r}_{R_i}, \mathbf{g}_{R_i}]), \tag{3}$$

where $\mathrm{MHA}(\cdot)$ denotes the multi-head attention operation and $[\cdot,\cdot]$ indicates vector concatenation. This mechanism allows the router to weigh retrieval quality and generation utility differently depending on the specific retrieval needs of the query q.

Similarity-Based Retriever Routing The final routing result is based on the alignment between the query's generation requirement $\mathbf{g}_q$ and the fused retriever capability $\mathbf{h}_{R_i}$. We select the optimal retriever $R_{\pi(q)}$ by maximizing the cosine similarity:

$$\pi(q) = \arg\max_{R_i \in \mathcal{R}} \mathrm{sim}(\mathbf{g}_q, \mathbf{h}_{R_i}), \tag{4}$$

where $\mathrm{sim}(\cdot,\cdot)$ denotes cosine similarity. This formulation ensures that the selected retriever is the one whose fused capability best matches the generation utility required by the query. This design choice reflects our emphasis on the final generation quality, as the retriever is selected based on how well it supports the downstream generation rather than retrieval performance alone.

4.2 Optimization

The effectiveness of a retriever in RAG is dynamic, exhibiting query-specific shifts in both retrieval quality and generation utility. To effectively capture these shifts, we employ a two-stage optimization strategy. In the first stage, we train the capability encoders (ϕ_r, ϕ_g) using contrastive learning

with disentangled supervision signals. In the second stage, we freeze these encoders and optimize the fusion mechanism to learn the optimal combination of capabilities.

4.2.1 Optimizing Capability Encoders

We design two complementary contrastive objectives to train ϕ_r and ϕ_g separately. This isolation ensures that each encoder captures its specific aspect of capability without interference from the other.

Retrieval Quality Encoder (ϕ_r) For retrieval quality, supervision should be independent of the generator to avoid confounding errors. We employ a powerful LLM (*e.g.*, Qwen3-Next-80B-A3B-Instruct (Yang et al., 2025)) to assess retrieved documents along dimensions such as relevance, coverage, and redundancy; the details are provided in Appendix A.4. Let $s(q, R_i)$ be the aggregated retrieval quality score. We construct positive set $\mathcal{V}_r^+(q)$ and negative set $\mathcal{V}_r^-(q)$ based on the top-k ranking of $s(q, R_i)$:

$$\begin{cases} \mathcal{V}^+(q) = \{\, \mathbf{r}_{R_i} \mid R_i \in \text{Top-}k_R(s(q,R)) \,\}, \\ \mathcal{V}^-(q) = \{\, \mathbf{r}_{R_i} \mid R_i \in \mathcal{R} \setminus \text{Top-}k_R(s(q,R)) \,\}. \end{cases}$$

The corresponding contrastive loss is formulated as:

$$\mathcal{L}_{\text{qual}}(q) = \sum_{\mathbf{r}^+ \in \mathcal{U}^+(q)} \Big(-\tfrac{1}{\tau} \operatorname{sim}(\mathbf{r}_q, \mathbf{r}^+) + \log Z_{\text{r}}(q) \Big),$$

$$Z_{\text{r}}(q) = \sum_{\mathbf{r} \in \mathcal{V}^+(q) \cup \mathcal{V}^-(q)} \exp\big(\operatorname{sim}(\mathbf{r}_q, \mathbf{r})/\tau\big),$$

where $\operatorname{sim}(\cdot,\cdot)$ denotes cosine similarity, τ is a temperature hyperparameter, and we set $k = 2$ in all experiments.

Generation Utility Encoder (ϕ_g) For generation utility modeling, we derive supervision signals from downstream generation outcomes. However, relying solely on the correctness of a single generated answer can be noisy and insufficient to characterize overall generation utility. To obtain a more robust supervision signal, we consider both query-level and retriever-level generation performance. Specifically, for a given query q and retriever R_i, we compute a generation utility score $u(q, R_i)$ combining: (i) query-level exact match (EM), (ii) query-level F1 score, and (iii) the retriever's global average correctness $\sigma(R_i)$:

$$u(q, R_i) = \text{EM}(q, R_i) + \beta \cdot \text{F1}(q, R_i) + \gamma \cdot \sigma(R_i). \quad (5)$$

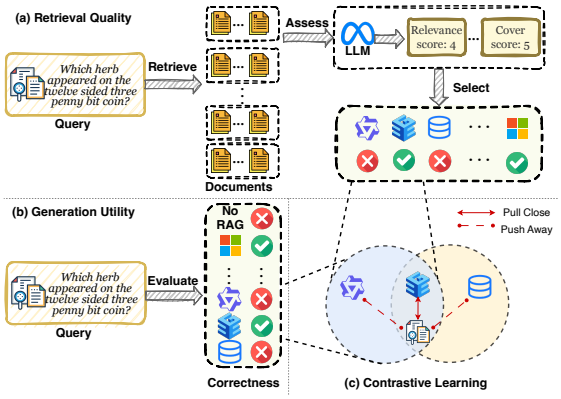

Figure 3: (a) Retrieval Quality supervision provides contrastive signals based on document-level quality assessments, decoupled from generation outcomes. (b) Generation Utility supervision derives labels from downstream answer quality, including both retrieval and no-retrieval cases. (c) Contrastive learning aims to align positive samples more closely with the query representation while distancing negative ones from it.

where β, and γ are weighting coefficients.

Similar to retrieval quality, we form positive set $\mathcal{V}_g^+(q)$ and negative set $\mathcal{V}_g^-(q)$ based on the top-k ranking of $u(q, R_i)$:

$$\begin{cases} \mathcal{V}^+(q) = \{\, \mathbf{g}_{R_i} \mid R_i \in \text{Top-}k_R(u(q,R)) \,\} \\ \mathcal{V}^-(q) = \{\, \mathbf{g}_{R_i} \mid R_i \in \mathcal{R} \setminus \text{Top-}k_R(u(q,R)) \,\}. \end{cases}$$

The corresponding contrastive loss is defined as:

$$\mathcal{L}_{\text{util}}(q) = \sum_{\mathbf{g}^+ \in \mathcal{V}^+(q)} \Big(-\tfrac{1}{\tau} \operatorname{sim}(\mathbf{g}_q, \mathbf{g}^+) + \log Z_{\text{g}}(q) \Big),$$

$$Z_{\text{g}}(q) = \sum_{\mathbf{g} \in \mathcal{V}^+(q) \cup \mathcal{V}^-(q)} \exp\big(\operatorname{sim}(\mathbf{g}_q, \mathbf{g})/\tau\big).$$

4.2.2 Optimizing Capability Fusion

Once the capability encoders are trained, we freeze their parameters and focus on optimizing the multi-head attention module. The goal of this stage is to learn how to weigh retrieval quality and generation utility dynamically to maximize the likelihood of selecting a useful retriever.

We define a matching probability $p(q, R_i)$ based on the similarity between the fused representation $\mathbf{h}_{R_i}$ and the query's utility requirement $\mathbf{g}_q$:

$$p(q, R_i) = \text{sigmoid}\left(\operatorname{sim}(\mathbf{h}_{R_i}, \mathbf{g}_q)\right). \quad (6)$$

The optimization target is to predict whether a retriever is "useful" (*i.e.*, belongs to the top-k positive set defined in the Generation Utility Encoder optimization). Let $y_{q_i} \in \{0, 1\}$ be the binary label

indicating if R_i is a positive sample for q. The classification loss is:

$$\mathcal{L}_{\text{cls}} = \mathbb{E}_q \sum_{R_i} \text{BCE}\left(p(q, R_i), y_{q_i}\right). \quad (7)$$

where $\text{BCE}(\cdot, \cdot)$ denotes the binary cross-entropy loss.

Additionally, we apply a regularization term to prevent the fused representation from deviating excessively from the generation utility embedding and ensure stability:

$$\mathcal{R}(\mathbf{h}_{R_i}, \mathbf{g}_{R_i}) = \|\mathbf{h}_{R_i} - \mathbf{g}_{R_i}\|_2^2. \quad (8)$$

The final objective for this stage is:

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \lambda_{\text{reg}} \mathcal{R}(\mathbf{h}_{R_i}, \mathbf{g}_{R_i}). \quad (9)$$

5 Experiments

5.1 Experimental Setup

Benchmarks To comprehensively evaluate R^{3}AG across varying levels of retrieval complexity, we conduct experiments on three representative knowledge-intensive benchmarks: (1) **TriviaQA** (Joshi et al., 2017), a large-scale open-domain reading comprehension dataset featuring factoid questions collected from trivia enthusiasts and paired with evidence from web documents; (2) **Natural Questions (NQ)** (Kwiatkowski et al., 2019), a benchmark consisting of real-world search queries that typically require identifying short answers within Wikipedia articles; and (3) **HotpotQA** (Yang et al., 2018), a challenging dataset that requires multi-hop reasoning across multiple supporting documents to derive the final answer. Following previous studies (Jin et al., 2025), we apply Exact Match (EM) and F1 score as the evaluation metrics.

Candidate Retrievers To simulate a realistic and heterogeneous retrieval environment, we construct a candidate pool of eight widely used retrievers. This set includes the classical sparse retriever BM25 and seven dense retrievers, spanning a broad spectrum of model sizes from 0.1B to 4B parameters. Comprehensive statistics on model scale and inference latency are reported in Table 1, while detailed implementation specifications are provided in Appendix A.1.

Table 1: Model size and latency of different retrievers. BM25 does not contain trainable parameters.

Retriever	Param. (B)	Latency (ms)
E5-base-v2	0.11	6.40
E5-large	0.34	10.11
BGE-large	0.34	11.09
BGE-m3	0.57	11.42
DIVER-Retriever-0.6B	0.60	23.39
Qwen3-embedding-0.6B	0.59	25.11
Qwen3-embedding-4B	4.02	32.19
BM25 (warm)	–	43.08
BM25 (cold)	–	91.49

Settings We employ `LLaMA3-8B-Instruct` as the backbone generator (Team, 2024), configured with a maximum context window of 2,048 tokens. For the knowledge source, we use the 2018 English Wikipedia dump (Karpukhin et al., 2020), retrieving the top-5 documents per query. For baselines utilizing a single fixed retriever, we select `E5-large` by default, as it demonstrated the best average performance in our preliminary analysis. To ensure fair comparison, standardized prompts are employed for all RAG and non-RAG settings, respectively, with full details provided in Appendix A.2.

To evaluate the effectiveness of different retrieval paradigms, we categorize the compared methods into four distinct types: (1) *Sequential*: Standard RAG approaches that follow a fixed "retrieve-then-generate" pipeline using a single retriever; (2) *Conditional*: Adaptive methods that explicitly decide whether to perform retrieval based on the input query; (3) *Loop*: Iterative methods that interleave retrieval and generation steps, dynamically determining when and what to retrieve during inference; and (4) *Route*: Routing-based methods, including our approach, which dynamically select the most suitable retriever from a candidate pool for each query.

Supervision Cost and Reproducibility. Since R^{3}AG relies on both retrieval-quality and generation-utility supervision, we report the cost in a throughput-based form. Specifically, generation-utility supervision is constructed from answer generation over (q, R_i) pairs, while retrieval-quality supervision is constructed using an external LLM judge over retrieved documents. We report measured throughput and approximate cost per 10k items in Appendix A.6 for easier scaling to different budgets.

Table 2: Performance comparison of R^{3}AG against retriever-based, rule-based, and adaptive RAG baselines across various knowledge-intensive tasks and RAG settings. EM and F1 scores on the test set are reported. "Pipeline Type" denotes the generation control paradigm. The best results are highlighted in **bold**, and the second best results are underlined.

Methods	Pipeline Type	TriviaQA		NQ		HotpotQA		Average	
		EM	F1	EM	F1	EM	F1	EM	F1
Naive	–	55.34	61.81	22.35	31.41	19.16	26.96	32.28	40.06
E5-base-v2	Sequential	58.83	68.12	36.12	47.35	25.31	35.75	40.09	50.41
E5-large	Sequential	60.16	69.56	36.89	48.00	26.87	37.87	41.31	51.81
BGE-large	Sequential	57.02	66.30	34.51	45.27	27.54	38.60	39.69	50.06
BGE-m3	Sequential	55.15	64.21	33.52	43.49	24.73	35.01	37.80	47.57
Qwen3-embedding-0.6b	Sequential	55.89	65.33	32.35	42.71	24.42	34.63	37.55	47.56
Qwen3-embedding-4b	Sequential	59.75	69.38	34.49	45.80	26.64	37.62	40.29	50.93
DIVER-Retriever-0.6b	Sequential	55.56	64.76	30.66	40.36	22.75	32.78	36.32	45.97
BM25	Sequential	48.70	59.04	12.93	18.15	17.87	26.15	26.50	34.45
Random	Conditional	56.49	65.60	30.53	40.51	22.13	32.09	36.38	46.07
Oracle Single Best	Conditional	60.16	69.56	36.89	48.00	27.54	38.60	41.53	52.05
FLARE	Loop	56.73	66.01	22.51	32.42	20.77	28.03	33.34	42.15
IRCoT	Loop	60.71	69.92	35.81	47.95	29.05	40.52	41.86	52.80
Adaptive-RAG	Conditional	57.41	66.81	35.12	45.55	28.87	40.54	40.47	50.97
AAR-contriever-kilt	Sequential	58.82	67.23	30.12	40.44	23.04	33.37	37.33	47.01
LTRR	Route	58.79	68.00	35.76	46.90	19.86	28.64	39.14	48.85
RouterRetriever	Route	57.42	66.73	32.44	43.31	24.37	34.98	38.08	48.34
R^{3}AG (ours)	Route	**62.24**	**71.52**	**37.81**	**49.32**	**29.13**	**40.73**	**43.06**	**53.86**

Baselines We compare R^{3}AG against a comprehensive set of baselines organized into three categories. (1) **Individual Retrievers and Naive Generation:** We first report the performance of the underlying generator without retrieval (**Naive**) and combined with each of the eight candidate retrievers individually (e.g., BM25, E5, etc., as detailed in the previous section). These results establish the fundamental performance spectrum of standard RAG systems. (2) **Heuristic Routing Strategies:** To assess the difficulty of the routing task, we provide reference points using rule-based strategies: (i) **Random**, which selects a retriever uniformly at random for each query; and (ii) **Oracle Single Best**, which ideally selects the best-performing retriever on each dataset. (3) **RAG Baselines:** Finally, we compare against advanced methods that optimize the RAG workflow: (i) **FLARE** (Jiang et al., 2023) and (ii) **IRCoT** (Trivedi et al., 2023), loop-based methods that interleave retrieval and generation, utilizing active confidence checks or chain-of-thought reasoning to guide retrieval; (iii) **Adaptive RAG** (Jeong et al., 2024), a conditional method that dynamically routes queries to no-retrieval, single-step, or multi-step pipelines based on complexity; (iv) **AAR** (Yu et al., 2023), which optimizes the retriever to align with LLM preferences, serving as a strong single-retriever baseline; (v) **RouterRetriever** (Lee et al., 2025), a routing method designed for IR settings that selects among domain-specific experts; and (vi) **LTRR** (Kim and Diaz, 2025), which formulates retriever routing as a learning-to-rank problem to dynamically rank candidate retrievers.

5.2 Overall Results

Comparison with Single Non-Routed Retrievers. As shown in Table 2, R^{3}AG attains the strongest test-set performance across diverse knowledge-intensive benchmarks and RAG configurations, reaching 43.06 average EM and 53.86 average F1. Compared with the top performed standalone retriever, E5-large, which obtains 41.31 EM and 51.81 F1, R^{3}AG yields clear gains, underscoring the benefit of routing. In addition, R^{3}AG exceeds the Oracle Single Best baseline, which chooses the most effective individual retriever for each dataset and achieves 41.53 EM and 52.05 F1. This comparison indicates that dynamically routing among multiple retrievers can better exploit their complementary capabilities and achieve results unattainable by any single retriever alone.

Comparison with Single and Static routing baselines. R^{3}AG significantly outperforms single and static routing baselines, for example, RouterRetriever (38.08 and 48.34) and LTRR (39.14 and 48.85). These findings highlight the limitations of

Table 3: Ablation study of R^{3}AG components. ΔEM and ΔF1 denote the average performance change across TQA, NQ, and HQA relative to the full model. RQ Encoder and GU Encoder denote the Retrieval Quality Encoder and Generation Utility Encoder, respectively.

	MHA	RQ Encoder	GU Encoder	TQA		NQ		HQA		ΔEM	ΔF1
				EM	F1	EM	F1	EM	F1		
R^{3}AG	✓	✓	✓	62.24	71.52	37.41	48.72	29.05	41.33	0.00	0.00
w/o MHA	×	✓	✓	61.57	70.43	36.87	47.94	27.58	38.76	-0.89	-1.88
w/o RQ Encoder	×	✓	×	60.95	69.03	36.91	48.13	28.03	38.82	-0.93	-2.26
w/o GU Encoder	×	×	✓	61.80	70.77	36.83	47.56	27.05	38.00	-1.01	-2.15

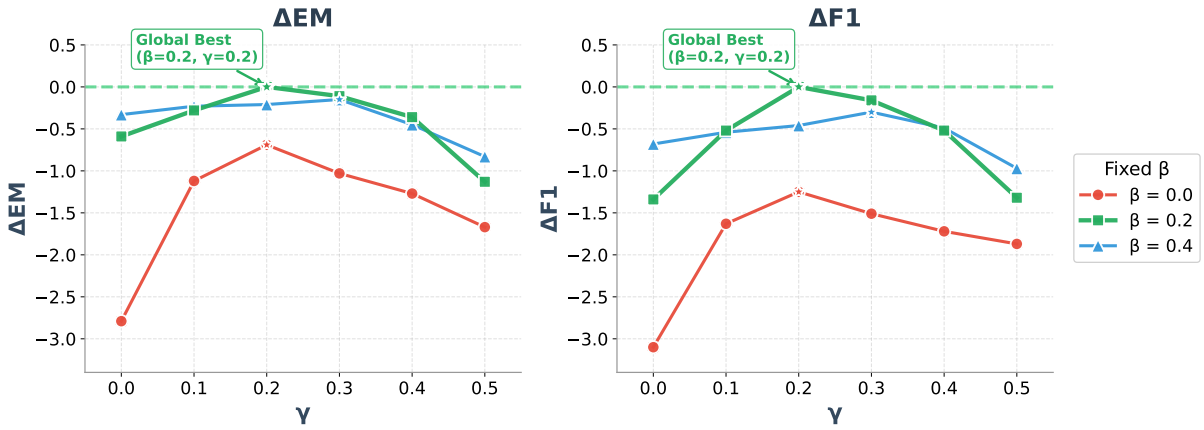

Figure 4: Sensitivity analysis of the β and γ weighting factors in training the Generation Utility Encoder.

approaches that do not explicitly capture retriever capability, including both retrieval quality and generation utility, and that are unable to adapt to the evolving interaction between the retriever and the query. Moreover, R^{3}AG also surpasses rule-based strategies such as Random (36.38 and 46.07) and Oracle Single Best (41.53 and 52.05). Taken together, these results substantiate our central claim that accurately routing retrievers requires explicitly modeling both retrieval quality and generation utility.

Comparison with Retrieval Necessity Judgement Approaches. R^{3}AG can also handle the task of retrieval necessity judgement. To reduce unnecessary retrievals, this problem has been extensively studied in the RAG literature. However, existing approaches typically focus on deciding whether to invoke a *single* retriever, treating retrieval as a binary decision. Such formulations can be viewed as a special case of retriever routing, which is naturally subsumed by R^{3}AG. Empirically, R^{3}AG outperforms these studies almost consistently, including FLARE (33.34 and 42.15), IRCOT (41.86 and 52.80), and Adaptive-RAG (40.47 and 50.97). These results suggest that retriever routing offers a principled generalization of retrieval necessity judgment in RAG settings.

5.3 Ablation Study and Sensitivity Analysis

Effects of R^{3}AG Architecture. We conduct an ablation study to evaluate the contributions of the Retrieval Quality (RQ) Encoder, Generation Utility (GU) Encoder, and the Multi-Head Attention (MHA) mechanism. As illustrated in Table 3, removing MHA and simply averaging retriever representations leads to consistent performance degradation across all datasets, highlighting the importance of query-adaptive capability fusion. Ablating either the RQ Encoder or the GU Encoder also results in noticeable performance drops, indicating that retrieval quality and generation utility provide complementary signals for effective routing. The full R^{3}AG model, which jointly models both factors and integrates them via MHA, achieves the best performance across all benchmarks.

Effects of Different Parameters in Sampling. We justify the choice of weighting factors β and γ through a sensitivity analysis shown in Figure 4. The results show that combining both signals consistently outperforms using either one alone, while removing both leads to substantial performance degradation. Performance peaks at moderate val-

Table 4: Analysis of R_0 (no-retrieval) selection. EM@R_0 denotes the exact match score on the subset where the router selects R_0.

Dataset	R_0 Selection Rate (%)	EM@R_0 (%)	No-Retrieval EM (%)	R^{3}AG EM (%)
TriviaQA	37.82	76.33	55.34	62.24
NQ	15.23	43.51	22.35	37.81
HotpotQA	9.52	31.78	19.16	29.13

Table 5: Cross-generator evaluation with Qwen3-8B at test time. The router is trained with LLaMA3-8B-Instruct and directly transferred without retraining.

Methods	TriviaQA		NQ		HotpotQA		Average	
	EM	F1	EM	F1	EM	F1	EM	F1
Naive	52.97	62.98	22.44	35.87	19.15	29.73	31.52	42.86
Random	53.57	64.98	26.41	37.22	29.09	40.06	36.36	47.42
Oracle-Single-Best	59.45	69.70	31.86	44.98	32.49	44.41	41.27	53.03
R^{3}AG	60.11	70.48	30.72	43.65	31.62	44.34	40.82	52.82

ues of β and γ (around 0.2). This trend indicates that retriever-level and answer-level supervision provide complementary but non-redundant signals, and balanced weighting is essential for stable training and effective retriever routing. Based on these observations, we set $\beta = \gamma = 0.2$ in all experiments.

Analysis of R_0 Selection. We further analyze the router's use of the no-retrieval option R_0. As shown in Table 4, R_0 is selected in 37.82%, 15.23%, and 9.52% of cases on TriviaQA, NQ, and HotpotQA, respectively, indicating that a non-trivial portion of queries can be handled without retrieval. In addition, EM@R_0 is consistently much higher than the overall no-retrieval baseline on all datasets, suggesting that the router can reliably identify queries for which retrieval is unnecessary. These results confirm that R_0 functions as an effective selective abstention option rather than a trivial fallback.

Analysis of Cross-Generator Generalization. We further evaluate whether the learned routing policy transfers across generators. In our main setting, the router is trained with LLaMA3-8B-Instruct as the backbone generator. To test robustness under a backbone swap, we directly replace the generator with Qwen3-8B at test time, while keeping the router fixed and performing no retraining. As shown in Table 5, R^{3}AG remains robust and competitive under this setting, achieving the best performance on TriviaQA and strong average results across the three datasets. This result suggests that, although generation utility may vary across generators, the learned routing policy transfers reasonably well across strong instruction-tuned LLMs. We attribute this robustness partly to our disentangled design, where the retrieval-quality channel provides a more stable, generator-agnostic signal.

6 Conclusion

In this paper, we explore the problem of retriever routing in RAG and propose R^{3}AG, a query-adaptive routing framework that goes beyond the conventional assumption of static retriever capability. Rather than treating retrievers as interchangeable modules or assessing them solely by retrieval relevance, R^{3}AG explicitly models two complementary aspects of retriever capability: retrieval quality and generation utility. By disentangling these factors and fusing them in a query-conditioned manner, R^{3}AG captures query-specific preferences over candidate retrievers. Experiments on multiple knowledge-intensive benchmarks show that R^{3}AG consistently outperforms strong single-retriever baselines, static routing methods, and retrieval necessity judgment approaches. Additional analyses further confirm that both retrieval quality and generation utility are crucial for effective routing. Overall, our results highlight the value of capability-aware retriever selection for building more adaptive and effective RAG systems.

7 Limitations

Despite achieving strong performance across multiple tasks and effectively fulfilling the intended routing objective, there remain several aspects that could be further explored to enhance the generality and applicability of the framework. First, motivated

by lightweight design considerations, the current approach adopts relatively simple representations for capability modeling; future work may investigate whether more expressive architectures can yield additional performance gains while maintaining a favorable efficiency–effectiveness balance. Second, although the router is able to select appropriate retrievers for individual queries, it does not yet consider more complex retrieval scenarios that involve combining multiple retrievers, which represents another promising direction for future exploration. Finally, the experimental evaluation is currently limited to question answering tasks, and extending the framework to a broader range of task settings would help better assess its generality.

Acknowledgments

This work was supported by National Natural Science Foundation of China No. 62402497 and No. 62272467, and the China Postdoctoral Science Foundation under Grant Number 2025T180440. The work was partially done at the Beijing Key Laboratory of Research on Large Models and Intelligent Governance and the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education.

References

Joshua Belofsky. 2023. Token-level adaptation of lora adapters for downstream task generalization. In *6th Artificial Intelligence and Cloud Computing Conference, AICCC 2023, Kyoto, Japan, December 16-18, 2023*, pages 168–172. ACM.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott M. Lundberg, Harsha Nori, Hamid Palangi, Marco Túlio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *CoRR*, abs/2303.12712.

Deng Cai, Yan Wang, Lemao Liu, and Shuming Shi. 2022. Recent advances in retrieval-augmented text generation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*, pages 3417–3419.

Shangbin Feng, Weijia Shi, Yuyang Bai, Vidhisha Balachandran, Tianxing He, and Yulia Tsvetkov. 2024. Knowledge card: Filling llms' knowledge gaps with plug-in specialized language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

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

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938. PMLR.

Hangfeng He, Hongming Zhang, and Dan Roth. 2023. Rethinking with retrieval: Faithful large language model inference. *CoRR*, abs/2301.00303.

Mahd Hindi, Linda Mohammed, Ommama Maaz, and Abdulmalik Alwarafy. 2025. Enhancing the precision and interpretability of retrieval-augmented generation (RAG) in legal technology: A survey. *IEEE Access*, 13:46171–46189.

Bay-Yuan Hsu, Pin-Yu Chen, and Chun-Yi Chen. 2025. Large-scale data retrieve in road networks with optimized k-retriever routing. *IEEE Transactions on Computational Social Systems*.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 874–880. Association for Computational Linguistics.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong 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), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 7036–7050. Association for Computational Linguistics.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 7969–7992. Association for Computational Linguistics.

Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In *Companion Proceedings of the ACM on Web Conference 2025, WWW 2025, Sydney, NSW, Australia, 28 April 2025 - 2 May 2025*, pages 737–740. ACM.

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, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1601–1611. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

To Eun Kim and Fernando Diaz. 2025. LTRR: learning to rank retrievers for llms. *CoRR*, abs/2506.13743.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Hyunji Lee, Luca Soldaini, Arman Cohan, Minjoon Seo, and Kyle Lo. 2025. Routerretriever: Routing over a mixture of expert embedding models. In *AAAI-25, Sponsored by the Association for the Advancement of Artificial Intelligence, February 25 - March 4, 2025, Philadelphia, PA, USA*, pages 11995–12003. AAAI Press.

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.

Kuan Li, Liwen Zhang, Yong Jiang, Pengjun Xie, Fei Huang, Shuai Wang, and Minhao Cheng. 2025. Lara: Benchmarking retrieval-augmented generation and long-context llms - no silver bullet for LC or RAG routing. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*. OpenReview.net.

Kevin Lin, Kyle Lo, Joseph Gonzalez, and Dan Klein. 2023. Decomposing complex queries for tip-of-the-tongue retrieval. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 5521–5533. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

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.

Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, Nicola Tonellotto, Nazli Goharian, and Ophir Frieder. 2020. Efficient document re-ranking for transformers by precomputing term representations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–58.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 9802–9822. Association for Computational Linguistics.

Mohammed Muqeeth, Haokun Liu, Yufan Liu, and Colin Raffel. 2024. Learning to route among specialized experts for zero-shot generalization. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

OpenAI. 2024. Gpt-4 technical report.

Chunyi Peng, Zhipeng Xu, Zhenghao Liu, Yishan Li, Yukun Yan, Shuo Wang, Zhiyuan Liu, Yu Gu, Minghe Yu, Ge Yu, and Maosong Sun. 2025. Learning to route queries across knowledge bases for

step-wise retrieval-augmented reasoning. *Preprint*, arXiv:2505.22095.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Derong Xu, Tong Xu, and Enhong Chen. 2024. Large language model based long-tail query rewriting in taobao search. In *Companion Proceedings of the ACM Web Conference 2024*, pages 20–28.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 2463–2473. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, and 1 others. 2018. Improving language understanding by generative pre-training.

Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David A. Sontag. 2024. Learning to decode collaboratively with multiple language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 12974–12990. Association for Computational Linguistics.

Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34:25968–25981.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, and Xian Li. 2024. Branch-train-mix: Mixing expert llms into a mixture-of-experts LLM. *CoRR*, abs/2403.07816.

Llama Team. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 10014–10037. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Guohang Yan, Yue Zhang, Pinlong Cai, Ding Wang, Song Mao, Hongwei Zhang, Yaoze Zhang, Hairong Zhang, Xinyu Cai, and Botian Shi. 2025. Hetarag: Hybrid deep retrieval-augmented generation across heterogeneous data stores. *CoRR*, abs/2509.21336.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 40 others. 2025. Qwen3 technical report. *CoRR*, abs/2505.09388.

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. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380. Association for Computational Linguistics.

Zichun Yu, Chenyan Xiong, Shi Yu, and Zhiyuan Liu. 2023. Augmentation-adapted retriever improves generalization of language models as generic plug-in. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 2421–2436. Association for Computational Linguistics.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Fan Wu, and Guihai Chen. 2025a. Query routing for retrieval-augmented language models. *CoRR*, abs/2505.23052.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *CoRR*, abs/2506.05176.

Zhuocheng Zhang, Yang Feng, and Min Zhang. 2025c. Levelrag: Enhancing retrieval-augmented generation with multi-hop logic planning over rewriting augmented searchers. *CoRR*, abs/2502.18139.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *CoRR*, abs/2303.18223.

A Experimental Setup Details

A.1 Details of Retrievers

All candidate retrievers are evaluated under a unified and reproducible experimental setup. Dense

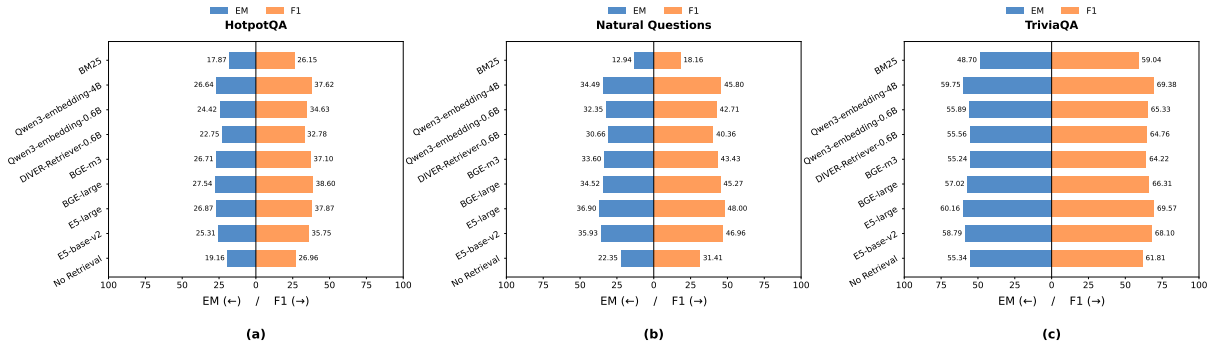

Figure 5: Mirrored horizontal bars show EM (left) and F1 (right) on HotpotQA, Natural Questions, and TriviaQA. Results are reported for multiple retrievers and a no-retrieval baseline, evaluated on the set of questions shared across all retrieval settings.

retrievers are locally deployed using the Faiss backend with GPU acceleration, while the sparse retriever BM25 is implemented via Pyserini. For dense models, inference latency is measured as the average encoding and retrieval time per query, excluding data loading and preprocessing overhead.

The candidate pool consists of one sparse retriever (BM25) and seven dense retrievers with parameter sizes ranging from 0.11B to 4.02B, covering a broad spectrum of efficiency–accuracy tradeoffs commonly encountered in practical RAG systems. All dense retrievers are used with their publicly released checkpoints, without additional fine-tuning, to ensure that observed performance differences stem from inherent model capability rather than task-specific adaptation. For BM25, we report both warm and cold latency to reflect different deployment conditions. Warm latency measures retrieval time with the index already loaded in memory, while cold latency additionally includes index loading overhead. In contrast, latency measurements for dense retrievers only account for model inference and nearest-neighbor search, as their indices remain resident in GPU memory during evaluation.

Comprehensive statistics on model size and inference latency are reported in Table 1. In addition, we provide per-dataset evaluation results for each retriever in the Figure 5, including EM and F1 scores on TriviaQA, NQ, and HotpotQA, to enable a more fine-grained comparison of retriever behavior across datasets.

A.2 Details of Global Setting

For the eight retrievers evaluated in this work, as well as other methods, unless explicitly stated otherwise, we adopt a standard retrieve–then–generate paradigm with a unified prompt template. Specifically, the retrieved documents are directly concatenated with the input query and fed into the generator, without any additional routing, filtering, or iterative control mechanisms. The system prompt instructs the model to generate answers strictly grounded in the provided documents:

> `Answer the question based on the given document. Only give me the answer and do not output any other words. The following are given documents:`

In the non-RAG setting, the prompt is defined as:

> `Answer the question based on your own knowledge. Only give me the answer and do not output any other words.`

The retrieved documents are appended in a structured format (`Doc` i `(Title:{title}) {content}`), where the top-5 documents are used for each query, followed by the user prompt `Question:{question}`. The system prompt, retrieved documents, and user prompt are combined using the `tokenizer.apply_chat_template` function to construct the final input sequence for the generator. This unified prompt design is consistently applied across all Naive RAG baselines to ensure a fair and controlled comparison.

A.3 Details of Method-Specific Setting

Beyond the general configuration described above, several methods require additional method-specific settings, which we describe in detail below.

IRCoT. IRCoT interleaves document retrieval with chain-of-thought reasoning. We employ the one-shot demonstrations provided in the original

work, and set the maximum number of reasoning–retrieval iterations to 2.[1]

AAR. For the Augmentation-Adapted Retriever (AAR), we directly use the pre-trained retriever checkpoints released by the authors.[2] Specifically, we adopt the AAR-Contriever-KILT variant for all experiments, without performing any additional fine-tuning.

A.4 Details of Retrieval Quality Scoring Prompt

To supervise retrieval quality independently of downstream generation, we employ a LLM as an automatic evaluator to assess the quality of retrieved documents. Given an input question and a set of retrieved documents, the evaluator is prompted to score the overall retrieval quality along multiple dimensions.

Prompt Example. Below we provide an example of the prompt used for retrieval quality scoring. The evaluator is instructed to act as a professional information retrieval expert and to assess the retrieved documents from several complementary perspectives, including relevance, coverage, accuracy, ranking quality, and redundancy.

```
You are a professional information retrieval quality evaluation expert. Please comprehensively evaluate the quality of documents returned by a retrieval system for the following question.

Question: {question}

Retrieved Documents:

Document 1 (ID: {id}): {content}

Document 2 (ID: {id}): {content}

...

Please evaluate the overall retrieval quality from the following dimensions:

(1) Relevance, (2) Coverage, (3) Accuracy, (4) Ranking Quality, and (5) Redundancy.

Output the evaluation strictly in the following JSON format:

{
  "relevance_score": <1-5>,
  "coverage_score": <1-5>,
  "accuracy_score": <1-5>,
  "ranking_score": <1-5>,
  "redundancy_score": <1-5>,
  "overall_score": <average score>,
}

Only output the JSON result and do not include any additional text.
```

The overall retrieval quality score is computed as the average of the five dimension-level scores and is subsequently used to construct positive and negative samples for contrastive training of the Retrieval Quality Encoder. This design allows us to obtain fine-grained and stable supervision signals that are decoupled from generation errors.

A.5 Details of Implementation

For R^{3}AG architecture, we use `qwen3-embedding-0.6b` (Zhang et al., 2025b) as both the Retrieval Quality Encoder ϕ_r and the Generation Capability Encoder ϕ_g. Encoders of the same type share parameters across queries and retrievers to ensure a consistent embedding space. Both the retrieval quality and generation utility embeddings have a dimension of 1024. To mitigate overfitting, we freeze all parameters of ϕ_r and ϕ_g except for the word embedding layer and the last two transformer layers during training. The encoders are optimized using AdamW (Loshchilov and Hutter, 2019) with a learning rate of 2×10^{-6} and an effective batch size of 512 (16 per device $\times$ 4 gradient accumulation steps $\times$ 8 GPUs), and are trained for 10 epochs. Training employs the InfoNCE loss with DeepSpeed ZeRO-3 optimization. We split the dataset with a validation ratio of 0.05. All experiments are conducted locally on 8 NVIDIA A100 GPUs.

A.6 Details of Supervision Throughput, Cost, and Reproducibility

To make the supervision overhead of R^{3}AG more transparent and reproducible, we report the measured throughput of each supervision component and provide an approximate cost per 10k supervision items. All measurements are conducted on one 8×A100 node with vLLM acceleration, whose hourly cost is approximately $5.68 in our environment.

For generation-utility supervision, one item corresponds to generating one RAG answer for a (q, R_i) pair, from which EM/F1-based utility labels are derived. For retrieval-quality supervision, one

[1] https://github.com/StonyBrookNLP/ircot/blob/main/prompts/2wikimultihopqa/gold_with_3_distractors_context_cot_qa_codex.txt

[2] https://huggingface.co/OpenMatch/AAR-Contriever-KILT

Table 6: Throughput and approximate cost per 10k supervision items (1×8×A100 node, vLLM; ≈$5.68/hour). Qwen3-8B and LLaMA3-8B-Instruct are used for generation-utility supervision, while Qwen3-Next-80B-A3B-Instruct is used as the retrieval-quality judge. RQ and GU denote the Retrieval Quality and Generation Utility, respectively.

Supervision Role	Model	items/s	Input toks/s	Output toks/s	Approx. cost / 10k
GU	Qwen3-8B	363.60	7,760.91	12,132.31	≈ $0.04
GU	LLaMA3-8B-Instruct	523.62	26,336.37	4,119.47	≈ $0.03
RQ Judge	Qwen3-Next-80B-A3B-Instruct	21.20	31,154.70	1,399.88	≈ $0.74

item corresponds to evaluating the retrieved documents for a query using the external LLM judge described in Appendix A.4. The approximate cost per 10k items is computed as:

$$\text{Cost per 10k items} = \frac{5.68}{3600} \times \frac{10000}{\text{items/s}}.$$

Table 6 shows that generation-utility supervision is relatively inexpensive under our setup, while retrieval-quality supervision is more costly due to the stronger external evaluator. We emphasize that the judge is used only to construct the retrieval-quality signal and is intentionally decoupled from the end-to-end generation-utility supervision. This makes it straightforward to swap the evaluator model or prompt in future reproductions and extensions.

B Retriever Encoding Strategy

In R^{3}AG, retrievers are represented as explicit, learnable entities within the capability encoding space, rather than being modeled through textual metadata or sampled retrieval outputs.

Retriever Representation. Each retriever $R_i \in \mathcal{R}$ is assigned a unique special token $\langle \mathsf{RET}_i \rangle$, which serves as its sole identifier. This token is added to the tokenizer vocabulary with a trainable embedding. The retriever is thus represented as a single-token input:

$$x_{R_i} = [\langle \mathsf{RET}_i \rangle].$$

No retriever descriptions, corpus statistics, or retrieval traces are used.

Encoding and Parameter Sharing. Both queries and retrievers are encoded by the same capability encoders. Specifically, the Retrieval Quality Encoder ϕ_r and the Generation Utility Encoder ϕ_g are shared across all queries and retrievers:

$$\mathbf{r}_q = \phi_r(q), \quad \mathbf{g}_q = \phi_g(q),$$

β	γ	TQA		NQ		HQA		ΔEM	ΔF1
		EM	F1	EM	F1	EM	F1		
0.2	**0.2**	**62.24**	**71.52**	**37.41**	**48.72**	**29.05**	**41.33**	**0.00**	**0.00**
0.2	0.0	61.66	70.18	36.81	47.39	28.44	39.99	-0.59	-1.34
0.2	0.1	61.94	70.99	37.14	48.21	28.78	40.82	-0.28	-0.52
0.2	0.3	62.13	71.37	37.29	48.56	28.94	41.16	-0.11	-0.16
0.2	0.4	61.88	70.98	37.06	48.19	28.69	40.83	-0.36	-0.52
0.2	0.5	61.12	70.19	36.27	47.41	27.91	40.03	-1.13	-1.32
0.0	0.0	59.44	68.41	34.62	45.63	26.27	38.22	-2.79	-3.10
0.0	0.1	61.13	69.88	36.28	47.09	27.94	39.71	-1.12	-1.63
0.0	0.2	61.54	70.28	36.73	47.46	28.36	40.07	-0.69	-1.25
0.0	0.3	61.22	70.01	36.37	47.21	28.02	39.81	-1.03	-1.51
0.0	0.4	60.98	69.79	36.14	46.98	27.78	39.63	-1.27	-1.72
0.0	0.5	60.57	69.64	35.74	46.84	27.39	39.47	-1.67	-1.87
0.4	0.0	61.92	70.83	37.07	48.04	28.72	40.66	-0.33	-0.68
0.4	0.1	62.01	70.97	37.18	48.19	28.82	40.78	-0.23	-0.54
0.4	0.2	62.04	71.07	37.19	48.27	28.84	40.86	-0.21	-0.46
0.4	0.3	62.09	71.21	37.26	48.43	28.89	41.04	-0.15	-0.30
0.4	0.4	61.78	71.02	36.96	48.24	28.61	40.84	-0.45	-0.49
0.4	0.5	61.41	70.56	36.57	47.74	28.23	40.36	-0.83	-0.97

Table 7: Sensitivity analysis of Generation Utility Encoder weights.

Table 8: End to end latency of different routing methods, where Lat. donetes Latency.

Methods	Min Lat. (ms)	Avg Lat. (ms)	Max Lat. (ms)
R^{3}AG	7.51	28.37	119.34
Random	7.23	38.79	102.66
Oracle Single Best	17.81	18.54	18.89

$$\mathbf{r}_{R_i} = \phi_r(\langle \mathsf{RET}_i \rangle), \quad \mathbf{g}_{R_i} = \phi_g(\langle \mathsf{RET}_i \rangle).$$

The only retriever-specific parameters are the embeddings of the retriever tokens, ensuring that queries and retrievers lie in a unified metric space.

Discussion. This design models retrievers as learned latent capability embeddings, optimized end-to-end via contrastive supervision. It enables efficient encoding, avoids reliance on heuristic retriever features, and allows new retrievers to be incorporated by simply introducing new tokens without modifying the architecture.

Query	Question	Gold	Non-RAG	RAG (selected)	Retrieved evidence (selected)
test_435	Which English word comes from an old French word meaning sour wine?	Vinegar	Vinegar (EM=1.0)	Qwen3-emb-0.6B → Sack; E5 → Bastardo; BM25 → Cheese	**Distracting:** "*Sack (wine)* ... derived from French *sec* (dry)"; "*Trousseau* ... also known as *Bastardo*"; "*Cheese* ... root meaning 'become sour' "
test_9847	Which perennial herb appeared on the twelve sided three penny bit coin?	Thrift	Thistle (EM=0.0)	Qwen3-emb-4B → Thrift plant (Acc=1.0); DIVER → Thrift plant (Acc=1.0); BM25 → Loonie	**Helpful:** "reverse shows a *three-headed thrift plant*" (History of the threepence); **Off-topic:** Canadian penny / loonie pages; unrelated plants (e.g., *Protea caffra*)

Table 9: Case study with retrieved evidence summaries. For readability, we show representative retrievers and one discriminative snippet per outcome (helpful vs. distracting).

C Extended Analysis

C.1 Extended Analysis of Parameters

This subsection reports the complete numerical results for the sensitivity analysis of the generation utility weighting coefficients β and γ in Eq. (5), as summarized in Table 7. We evaluate different combinations of β and γ on three downstream benchmarks (TQA, NQ, and HQA), and report both EM and F1 scores, along with their absolute differences relative to the default setting $\beta = \gamma = 0.2$. The results show that removing either query-level answer supervision ($\beta = 0$) or retriever-level global supervision ($\gamma = 0$) consistently degrades performance, while removing both leads to a substantial drop across all datasets. Introducing non-zero values for either coefficient partially recovers performance, but the best and most stable results are achieved when both signals are combined with moderate weights. In particular, settings with β and γ in the range of 0.1 to 0.3 yield comparable performance, whereas overly small or large values result in suboptimal outcomes. These observations indicate that query-level answer quality and retriever-level global correctness provide complementary but non-redundant supervision signals for generation utility modeling. Based on this analysis, we adopt $\beta = \gamma = 0.2$ in all experiments.

C.2 Extended Analysis of Efficiency

We further analyze the end-to-end efficiency of different routing strategies, with results reported in Table 8. The latency is measured in an end-to-end manner, including both the routing overhead (retriever selection and retrieval) and the downstream generation time. For all methods, the generation component is executed locally using the `vLLM` framework (Kwon et al., 2023) with identical decoding configurations, ensuring a fair comparison under a high-throughput inference setting.

As shown in Table 8, the proposed method achieves competitive average latency while maintaining a low minimum latency comparable to Random routing, indicating negligible routing overhead in favorable cases. Although the maximum latency of our method is higher due to dynamic retriever selection and varying generation lengths, it remains within a reasonable range and does not significantly impact overall efficiency. In contrast, Random routing exhibits higher average latency, reflecting its lack of informed selection, while Oracle Single Best shows stable but higher minimum latency, as it consistently routes all queries to a fixed retriever regardless of query difficulty. Overall, these results demonstrate that our approach strikes a favorable balance between routing adaptivity and end-to-end efficiency.

D Case Study

We analyze two representative TriviaQA cases to highlight (i) scenarios where retrieval is unnecessary and even detrimental, and (ii) scenarios where retrieval is essential but highly sensitive to retriever quality. Together, these cases motivate query-adaptive routing and retriever-aware decision making.

Case 1: Retrieval Can Be Unnecessary and Harmful. For some queries, the generator's parametric knowledge is sufficient to produce the correct answer. Introducing retrieval in such cases may inject misleading evidence and shift the generation distribution. As a result, RAG can underperform non-RAG baselines, indicating that routing mechanisms should allow skipping retrieval when confidence in parametric knowledge is high.

Case 2: Retrieval Is Necessary but Retriever Capability Matters. Other queries fundamentally depend on external evidence. In these cases, effective retrievers surface precise and discriminative information that enables correct generation, while weaker retrievers return noisy or irrelevant documents that still lead to failure. This underscores that, beyond deciding whether to retrieve, routing must account for retriever-specific capability.

E Usage of AI Assistants

We use ChatGPT to improve the presentations of this paper.[3]

[3] https://chatgpt.com/