RES-RAG: Reinforced Evidence Set Retrieval for Retrieval-Augmented Generation

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) is widely used to incorporate external knowledge into large language models (LLMs), effectively improving answer quality in question-answering (QA) tasks. Since external knowledge may far exceed the context window of LLMs, retrieving the key evidence they need within a limited context is challenging. Existing heuristic retrieval methods often add more evidence in context to improve answer quality, but some non-essential evidence may contribute little, potentially leading to over-retrieval and increased noise. To overcome these challenges, we propose RES-RAG, which, for the first time, models evidence retrieval as an action decision problem under context constraints. Compared to existing methods such as top-k and truncation, RES-RAG can adaptively retrieve evidence that contributes to answer quality. Specifically, we construct a lightweight retrieval strategy model. We use reinforcement learning (RL) to optimize the retrieval process, making evidence retrieval quality-oriented. Furthermore, we design a joint quality-cost function for the optimization signal in RL. We use the F1 score as a positive reward and penalize over-retrieval. This achieves a balance between answer quality and retrieval cost. Experiments across multiple open-domain QA benchmarks demonstrate that RES-RAG achieves higher answer quality at lower evidence cost, reflecting a superior quality-cost trade-off. Code and data can be found at https://anonymous.4open.science/r/RES-RAG-0106.

1 Introduction

Retrieval-Augmented Generation (RAG) has been emerged as a popular paradigm for mitigating the hallucination and knowledge timeliness problems of large language models (LLMs) (Shuster et al., 2021; Asai et al., 2024; Béchard and Ayala, 2024; Huang et al., 2025; Tonmoy et al., 2024; Izacard and Grave, 2020; Guu et al., 2020; Izacard et al., 2023). Its core idea is to retrieve external evidence from updatable corpora and combine it with the model's parameterized knowledge to answer given open questions(Karpukhin et al., 2020; Izacard et al., 2023; Kim and Lee, 2024; Gao et al., 2023c; Vu et al., 2023; Lewis et al., 2020). However, as longer and more complex contexts are introduced, the model's context window needs to expand longer and longer. As a result, the computational and latency costs of long-context reasoning typically increase significantly with input size, making evidence retrieval under budget constraints a key trade-off between performance and cost (Lin et al., 2025; Taguchi et al., 2025; Lahmy and Yozevitch, 2025).

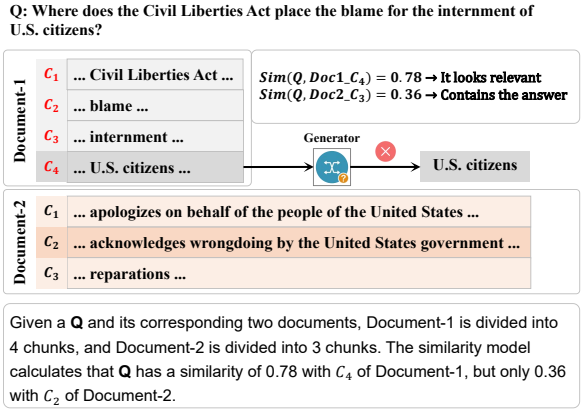

Figure 1: Given a query, similarity-based retrieval methods tend to select Document-1 because its text blocks match the query terms. However, Document-2 contains the key answer to the query.

Existing research typically focuses on optimizing each component (Gao et al., 2023b; Formal et al., 2021; Ren et al., 2021; Pradeep et al., 2020; Press et al., 2022). For example, stronger retrievers (Izacard et al., 2021; Xiong et al., 2020; Ni et al., 2021; Xiao et al., 2022a), more robust rerankers, or more reliable generators (Yu et al., 2021; Zemlyan-

skiy et al., 2021; Glass et al., 2022; Su et al., 2022a). In these pipelined designs, a common assumption is that the context can be simplified to a set of text fragments (Chen et al., 2020; Li et al., 2021). Furthermore, evidence retrieval relies on heuristic truncation, fixed windows, or similarity-based top-k strategies. This approach presents two main challenges when dealing with long, complex contexts or scattered evidence: (1) it introduces a large amount of irrelevant noise that interferes with the generator's reasoning; (2) it may miss crucial evidence within the context window, leading to incomplete answers (Karpukhin et al., 2020; Krishna et al., 2021; Dai et al., 2025). These challenges limit existing paradigms' ability to approach the optimal efficiency frontier along the answer quality–evidence cost dimension.

To address the above challenges, we propose RES-RAG (Reinforced Evidence Set Retrieval for Retrieval-Augmented Generation). Given a query and its corresponding set of context documents, we train a lightweight model as the evidence retriever that outputs the set of retrieved chunk indices for each document. This design explicitly models evidence retrieval as a structured prediction task, enabling the system to produce controllable results under context constraints. Meanwhile, structured output has been shown to significantly improve the model's controllability and usability (Scholak et al., 2021; Muralidharan and Thomas, 2024; Beurer-Kellner et al., 2024; Park et al., 2024).

First, we place evidence retrieval within a budget-aware reinforcement learning (RL) optimization framework. We call the maximum context capacity available for retrieving evidence the budget, and the context actually consumed by the evidence the cost. The set of indices output by the retrieval model is treated as actions. This process offers two key advantages: (1) the retrieval results are presented in an indexed format, reducing the complexity of the action space; (2) the generator receives a compact set of retrieved evidence, reducing the burden of long contextual noise and increasing the effective evidence density.

Secondly, to improve the RL optimization process, we design a quality-cost joint reward: the reward signal is measured by the F1 score; redundant evidence and over-retrieval are penalized by an over-budget penalty. This reward characterizes the two-dimensional trade-off between answer quality and evidence cost on the same scale, thereby driving the model to converge at the minimally sufficient evidence granularity. Optimizing evidence retrieval via RL under joint quality–cost constraints provides a practical way to improve the quality–cost trade-off in the RAG pipeline (Guo et al., 2025; Li et al., 2025b).

Finally, we systematically evaluate RES-RAG's performance across several popular QA datasets, demonstrating that it can improve QA task quality at lower cost.

The main contributions of this paper are as follows:

- We propose an optimization framework centered on evidence retrieval that optimizes retrieval strategies via RL.
- We design a joint quality-cost function to improve the trade-off between answer quality and evidence cost.
- We evaluate the system across multiple open-domain question-and-answer benchmarks and report end-to-end metrics and costs.

In summary, our goal is to propose a budget-aware evidence-retrieval mechanism for QA tasks, enabling LLMs to select better factual evidence within a limited budget and to optimize the quality and cost of answer generation.

2 Related Work

2.1 RAG for LLM Inference

A typical RAG pipeline usually includes steps such as query rewriting or expansion, candidate retrieval, reranking, and fusing query and evidence to generate the answer (Ma et al., 2023; Wang et al., 2024b). In QA scenarios, RAG performance depends not only on retrieval recall but also heavily on the evidence organization method and the generator's ability to locate evidence within the context. Therefore, the coupling effect of retrieval — evidence—generation has been widely discussed (Barnett et al., 2024; Siriwardhana et al., 2023; Besta et al., 2024). Although existing work has made significant progress in retrieval training (Santhanam et al., 2021; Shi et al., 2023), reranker design (Zhang et al., 2023; Edge et al., 2024), and generator fusion strategies (de Jong et al., 2022; Akimoto et al., 2024), most methods still simplify evidence retrieval to global top-k or similarity threshold filtering (Saxena et al., 2025; Xin et al., 2025), assuming that these heuristic retrieval strategies can adequately support downstream short-answer

generation. This default assumption is prone to failure in situations involving long contexts and scattered evidence. In particular, when complementary information across chunks, or the elimination of interfering segments, is required, relevance ranking with a fixed k often fails to align with the final metric (Hei et al., 2024; Mortaheb et al., 2025).

2.2 Evidence Retrieval and Reranking

To improve the evidence quality of RAG, one type of method sorts candidate passages using a dense retriever, then truncates the top-k chunks as the evidence(Pradeep et al., 2023; Zheng et al., 2024). The advantage of this type of method is its relatively simple training and deployment. Still, its retrieval strategy is usually fixed-length, either globally or weakly adaptive, making it difficult to consider the contribution of different passages to the answer (Xu et al., 2023). Another type of research focuses on more refined evidence retrieval, such as learning a gating mechanism on the candidate set to decide which chunks to retain (Wang et al., 2023; Sheng et al., 2025; Xie et al., 2024). These selectors are often trained with supervisory signals to reduce noise and increase evidence density (Shen et al., 2022). However, supervised retrieval is often limited by labeling costs or pseudo-label bias, and a gap remains between local relevance and final answerability under long context conditions, making it difficult for the retriever to generalize stably (Su et al., 2022b; Asai et al., 2024; Christmann et al., 2024). Furthermore, previous work has pointed out that adding more seemingly relevant fragments reduces the generator's attention allocation to key evidence, thereby impairing the stability of answer generation (Günther et al., 2024; Ma et al., 2022; Wen et al., 2004). Therefore, evidence retrieval is also a problem of achieving high-density, low-interference evidence assembly under a limited context budget (Dai et al., 2025; Wang et al., 2024d).

2.3 RL for Retrieval Policies

To address the target-mismatch problem of inconsistency between relevance scores and answer quality, some works use reinforcement learning to treat retrieval as a decision-making process and train the retrieval model using the generator's answer quality as the reward signal (Kulkarni et al., 2024; Wang et al., 2024c). In this paradigm, the policy model can leverage end-to-end feedback to learn which combinations of evidence are more likely to yield a correct answer, thereby surpassing heuristics based on fixed thresholds or static top-k rules (Java et al., 2025; Tang et al., 2025). Existing RL-related work covers multiple levels, including using RL to learn query rewriting and retrieval action sequences (Khattab et al., 2022; Li et al., 2025a; Xu et al., 2025), passage retrieval/stopping strategies (Zhu et al., 2025), and treating the reranker as a learnable ranking strategy (Zhang et al., 2025; Gao et al., 2023a). However, these methods often employ a global retrieval action space, making it difficult to directly characterize the structured constraints on how context contributes evidence and the extent to which each piece of evidence contributes (Liu and Lapata, 2019; Xiao et al., 2022b).

We propose RES-RAG to enable the retrieval strategy to learn how to retrieve the sufficient and necessary set of evidence in complex, long-context scenarios.

3 Method

We propose RES-RAG, a budget-aware framework for evidence retrieval and answer generation in complex long-context. Given a query and its corresponding set of HTML documents, RES-RAG models evidence retrieval as a structured decision. The retriever policy selects a compact set of chunk indices under a chunk budget, and a frozen generator $\mathcal{G}$ produces the answer conditioned on the retrieved evidence and the JSON-LD[1] annotation. We optimize the retriever via RL, balancing rewards to encourage minimal sufficient evidence and high-quality answers.

3.1 Task Definition

Each instance consists of a query q, a set of M HTML documents:

$$\mathcal{D} = \{d_1, \ldots, d_M\}, \quad (1)$$

and a gold answer $a^\star$. Each document d_m ($1 \leq m \leq M$) includes structured annotation in JSON-LD format. Our goal is to learn a retriever policy π_θ that outputs a structured action:

$$\hat{S} = \{\hat{S}_1, \ldots, \hat{S}_M\}, \quad (2)$$

where $\hat{S}_m$ denotes the retrieved chunk indices from d_m. In parallel, we extract JSON-LD annotation from each document via HTML parsing[2]:

$$J_m = \mathrm{JSONLD}(d_m), \quad 1 \leq m \leq M, \quad (3)$$

[1] https://json-ld.org/

[2] https://www.crummy.com/software/BeautifulSoup/

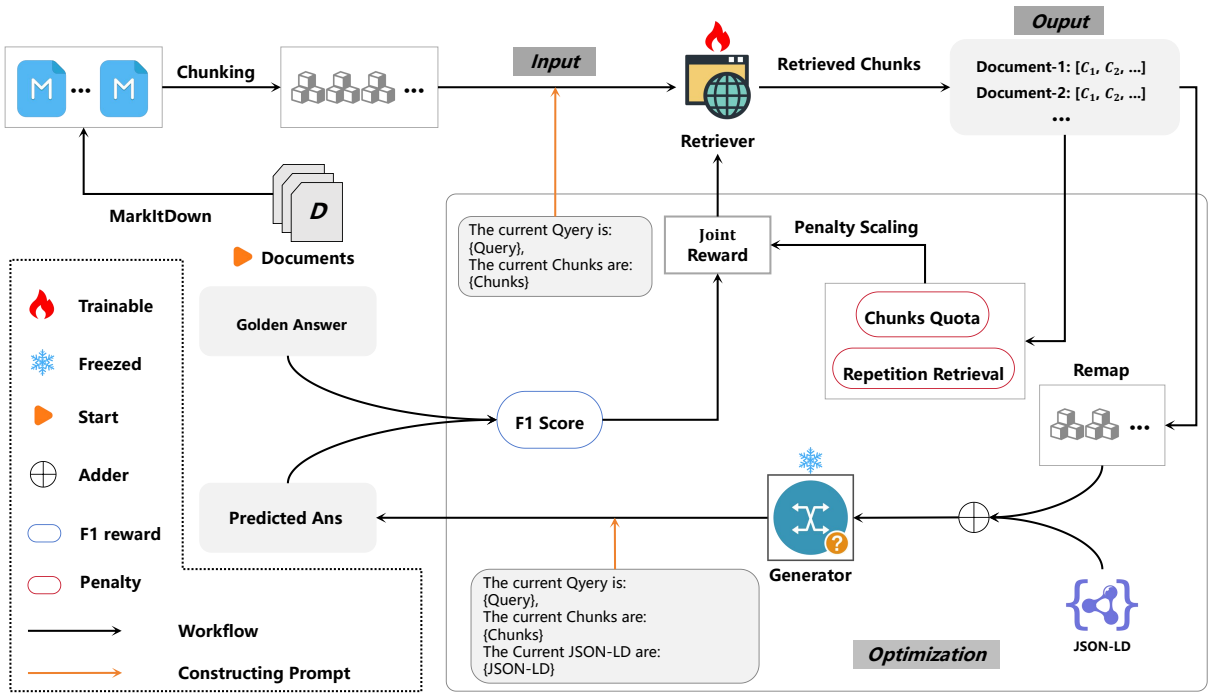

Figure 2: RES-RAG framework. Our training uses the GRPO algorithm. The optimization objective of RES-RAG is the Retriever, and the joint reward is calculated using the predicted answer generated by the Generator and a penalty.

and denote the aggregated annotation as $J = \{J_1, \ldots, J_M\}$. Given $\hat{S}$, retrieved indices are remapped to evidence text and, together with J, are fed into a frozen generator $\mathcal{G}$ to produce an answer $\hat{a}$.

3.2 Document Processing and Chunking

We convert each d_m into a Markdown string using MarkItDown[3]: $x_m = \text{Markdown}(d_m)$. We then split x_m into N_m chunks: $\mathcal{X}_m = \{x_{m,1}, x_{m,2}, \ldots, x_{m,N_m}\}$. Each chunk is constrained by a maximum token length L:

$$\text{len}\big(\text{Tok}_{\mathcal{G}}(x_{m,n})\big) \le L, \quad 1 \le n \le N_m, \tag{4}$$

where $\text{Tok}_{\mathcal{G}}(\cdot)$ uses the same tokenizer as the retriever, ensuring the chunk length matches the actual context usage at inference time. Algorithm A provides the overall chunking procedure. For convenience, we denote the set of all chunk lists as $\mathcal{X} = \{\mathcal{X}_1, \ldots, \mathcal{X}_M\}$.

3.3 Components of RES-RAG and Reward Design

Structured Evidence Retriever The Retriever takes $(q, \mathcal{X})$ as input and outputs a structured action $\hat{S}$ specifying which chunks to include as evidence. To make evidence decisions explicit and auditable, we serialize $\hat{S}$ as a JSON object:

$$\begin{aligned}\hat{S} = \{&\texttt{"Document\_1"} : [s_{1,1}, \ldots, s_{1,K_1}],\\ &\ldots,\\ &\texttt{"Document\_M"} : [s_{M,1}, \ldots, s_{M,K_M}]\},\end{aligned} \tag{5}$$

where each $s_{m,k}$ is an integer chunk index satisfying $s_{m,k} \in \{1, \ldots, N_m\}$, and we allow $\hat{S}_m = \varnothing$, this means no evidence be retrieved from d_m. Within each d_m, we encourage π_θ to output indices in descending order of their estimated relevance to the q. We parameterize the retriever as a conditional generative policy: $\hat{S} \sim \pi_\theta(\cdot \mid q, \mathcal{X})$, implemented by an instruction-tuned LLM that directly outputs the JSON action. To improve parsability and reduce invalid actions, we produce constraints during decoding (Appendix B).

Generator and Evidence Remapping Given a sampled action $\hat{S}$, we remap indices to evidence text by selecting corresponding chunks and serializing them in a fixed order:

$$E(\hat{S}) = \text{Concat}_{m=1}^{M}\ \text{Concat}_{n \in \hat{S}_m} x_{m,n}, \tag{6}$$

where Concat follows the index order in each list $\hat{S}_m$. We then use $\mathcal{G}$ to produce an answer:

$$\hat{a} = \mathcal{G}\big(q,\ E(\hat{S}),\ J\big). \tag{7}$$

[3] https://github.com/microsoft/markitdown

We serialize (E, J) into $\mathcal{G}$ prompt using a fixed template (Appendix C).

Chunk-level Budget Penalty We penalize over-retrieval at the chunk level to encourage compact evidence sets. For each d_m, define the retrieved-chunk count $C_m(\hat{S}) = |\hat{S}_m|$. Let N_m be the number of chunks in d_m after chunking. We define a segmented per-document penalty with a free region proportional to $\sqrt{N_m}$:

$$\phi_m(\hat{S}) = \begin{cases} 0, & \text{if } C_m(\hat{S}) \le \sqrt{N_m}, \\ 0.5, & \text{if } \sqrt{N_m} < C_m(\hat{S}) \le N_m, \\ 1, & \text{if } C_m(\hat{S}) > N_m. \end{cases} \tag{8}$$

The overall budget penalty averages across documents and scales by $\eta > 0$:

$$P_{\text{budget}}(\hat{S}) = \eta \cdot \frac{1}{M} \sum_{m=1}^{M} \phi_m(\hat{S}). \tag{9}$$

Answer Quality and Retrieval Consistency We measure answer quality using token-level F1 between $\hat{a}$ and $a^\star$: $\text{F1}_{\text{ans}}(\hat{a}, a^\star) \in [0, 1]$, with standard normalization, including lowercasing, punctuation removal, and whitespace normalization. To stabilize learning when $\mathcal{G}$ produces unusable answers, we use a fixed non-positive penalty $\gamma_{\text{bad}} < 0$:

$$U_{\text{ans}}(\hat{a}) = \begin{cases} \alpha_{\text{ans}} \cdot \text{F1}_{\text{ans}}(\hat{a}, a^\star), & \hat{a} \text{ is not empty}, \\ \gamma_{\text{bad}}, & \text{otherwise}, \end{cases} \tag{10}$$

where $\alpha_{\text{ans}} > 0$. Beyond answer quality, we add an auxiliary retrieval-consistency signal against heuristic labels $\tilde{S}$ (Sec . 3.4.1), which provides a denser learning signal than answer-only rewards. Although $\hat{S}_m$ and $\tilde{S}_m$ are output as index lists, each index uniquely identifies a chunk text $x_{m,n}$, so we compute consistency at the *chunk level*. Let $\text{set}(\cdot)$ remove duplicates from a list, and define the retrieved and labeled evidence-chunk sets for d_m as

$$\begin{aligned} \hat{\mathcal{E}}_m &= \{x_{m,n} \mid n \in \text{set}(\hat{S}_m)\}, \\ \tilde{\mathcal{E}}_m &= \{x_{m,n} \mid n \in \text{set}(\tilde{S}_m)\}. \end{aligned} \tag{11}$$

We then compute per-document chunk-level F1:

$$\text{F1}_m(\hat{\mathcal{E}}_m, \tilde{\mathcal{E}}_m) = \frac{2|\hat{\mathcal{E}}_m \cap \tilde{\mathcal{E}}_m|}{|\hat{\mathcal{E}}_m| + |\tilde{\mathcal{E}}_m|}, \tag{12}$$

with the convention that $\text{F1}_m = 1$ if both sets are empty, and 0 if only one is empty. The macro-averaged retrieval consistency is

$$\text{F1}_{\text{ret}}(\hat{S}, \tilde{S}) = \frac{1}{M} \sum_{m=1}^{M} \text{F1}_m(\hat{\mathcal{E}}_m, \tilde{\mathcal{E}}_m). \tag{13}$$

Joint Reward Putting everything together, for valid actions, we define the reward:

$$\begin{aligned} R(\hat{S}) = \text{clip}\Big(&U_{\text{ans}}(\hat{a}) \\ &+ \alpha_{\text{ret}} \cdot \text{F1}_{\text{ret}}(\hat{S}, \tilde{S}) \\ &- P_{\text{budget}}(\hat{S}),\ R_{\min}, R_{\max}\Big), \end{aligned} \tag{14}$$

where $\alpha_{\text{ret}} > 0$, and $\text{clip}(\cdot, R_{\min}, R_{\max})$ improves optimization stability. In our design, we apply an additional fixed negative reward for samples caused by external failures.

3.4 Training the Retriever via GRPO

We optimize only the Retriever π_θ using GRPO. The $\mathcal{G}$ is frozen and serves as part of the environment that outputs $\hat{a}$ based on $\hat{S}$, then outputs a scalar reward:

$$(q, \mathcal{X}) \xrightarrow{\pi_\theta} \hat{S} \xrightarrow{\text{remap}} (E, J) \xrightarrow{\mathcal{G}} \hat{a} \xrightarrow{\text{score}} R(\hat{S}). \tag{15}$$

3.4.1 SFT Warm-up

To stabilize structured generation and reduce RL exploration difficulty, we warm up the Retriever with supervised fine-tuning (SFT). We construct heuristic labels $\tilde{S}$ using a similarity-based rule and fine-tune π_θ to imitate the labeled JSON actions.

Specifically, for each d_m, we compute a similarity score between the q and each chunk $x_{m,n}$ using a sentence embedding model[4]. We keep chunks with similarity greater than a threshold (0.4 in our design) as the top K_m indices to form the label list:

$$\tilde{S}_m = [\tilde{s}_{m,1}, \ldots, \tilde{s}_{m,K_m}], \quad \tilde{s}_{m,k} \in \{1, \ldots, N_m\}. \tag{16}$$

All indices in both $\tilde{S}$ and $\hat{S}$ are arranged by relevance to, in the same form as Equation 5. The resulting checkpoint initializes the RL policy, and a frozen copy serves as the reference policy, $\pi_{\theta_{\text{ref}}}$.

3.4.2 Group Sampling and Relative Advantage

For each training instance t, we sample a group of W candidate actions using the behavior policy $\pi_{\theta_{\text{old}}}$:

$$\{\hat{S}_{t,1}, \ldots, \hat{S}_{t,W}\} \sim \pi_{\theta_{\text{old}}}(\cdot \mid q_t, \mathcal{X}_t). \tag{17}$$

[4] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

Each candidate is evaluated by the $\mathcal{G}$, producing reward $\hat{R}_{t,w} = R(\hat{S}_{t,w})$. To mitigate reward-scale differences across prompts and reduce gradient variance, we compute group-relative advantages by normalizing rewards within each prompt group:

$$\bar{R}_t = \frac{1}{W}\sum_{w=1}^{W}\hat{R}_{t,w},\ \sigma_t = \sqrt{\frac{1}{W}\sum_{w=1}^{W}(\hat{R}_{t,w} - \bar{R}_t)^2}, \tag{18}$$

$$\mathrm{ADV}_{t,w} = \frac{\hat{R}_{t,w} - \bar{R}_t}{\sigma_t + \epsilon_{\mathrm{adv}}}, \tag{19}$$

where ϵ_{adv} is a constant for numerical stability.

3.4.3 Clipped Policy Optimization Objective

Let $\pi_{\theta_{\mathrm{old}}}$ denote the behavior policy used to sample actions. We define the importance ratio

$$\rho_{t,w} = \frac{\pi_\theta(\hat{S}_{t,w} \mid q_t, \mathcal{X}_t)}{\pi_{\theta_{\mathrm{old}}}(\hat{S}_{t,w} \mid q_t, \mathcal{X}_t)}. \tag{20}$$

We optimize a clipped policy-gradient objective:

$$\begin{aligned}\mathcal{L}(\theta) = \mathbb{E}\Big[\min\Big(&\rho_{t,w}\mathrm{ADV}_{t,w},\\ &\mathrm{clip}(\rho_{t,w}, 1-\epsilon, 1+\epsilon)\mathrm{ADV}_{t,w}\Big)\Big]\\ -\beta\cdot\mathbb{E}\Big[&\mathrm{KL}\Big(\pi_\theta(\cdot \mid q_t, \mathcal{X}_t)\,\Big\|\,\pi_{\theta_{\mathrm{ref}}}(\cdot \mid q_t, \mathcal{X}_t)\Big)\Big]\end{aligned} \tag{21}$$

where ϵ is the clipping range, and $\beta \geq 0$ controls KL regularization to the reference policy, and $\mathbb{E}(\cdot)$ is taken over actions $\hat{S}_{t,w}$ sampled from $\pi_{\theta_{\mathrm{old}}}(\cdot \mid q_t, \mathcal{X}_t)$.

4 Experiments

We conduct experiments on the following questions:

RQ1: How can RES-RAG improve performance in terms of quality-cost trade-off?

RQ2: How does the design of the joint quality-cost reward function in RES-RAG affect the effectiveness of evidence retrieval?

RQ3: Can RES-RAG maintain universality across RAG systems with different document structures and generator configurations?

4.1 Experimental Settings

During the training phase, we run GRPO on two NVIDIA RTX 4090 GPUs, and during the inference phase, we deploy the trained model on one NVIDIA RTX 4090 GPU. The retrieval model we used is Qwen[5], a small model with only 4B parameters.

Datasets We conducted experiments with RES-RAG and various baseline models on six open-domain QA datasets, including: (1) two ambiguous datasets, ASQA (Stelmakh et al., 2022) and AbmigQA (Min et al., 2020); (2) three multi-hop QA datasets, Hotpot-QA (Yang et al., 2018), 2wikiMultiHop (Ho et al., 2020), and MusiQue (Trivedi et al., 2022); and (3) an open web QA dataset, Trivia-QA (Joshi et al., 2017).

Evaluation Metrics Our goal is for Retriever to retrieve as few query-relevant evidence chunks as possible from complex web environments. Therefore, we evaluate the short answer output by the generator $\mathcal{G}$ as the final result. For ambiguous datasets, we report both Hit@1 and EM metrics; for other datasets, we report the EM metric. Hit@1 indicates that at least one golden answer perfectly matches the $\mathcal{G}$'s output.

Implementation Details For each query, we sample $W = 4$ candidate structured actions. We keep the KL coefficient at $\beta = 0.05$. More training parameters can be found in Appendix D.

4.2 Comparisons with Other Methods

We compare RES-RAG against widely used baselines, including lexical retrieval BM25 (Robertson et al., 2009), dense retrievers BGE (Xiao et al., 2024) and E5-Mistral (Wang et al., 2024a), and compression systems LongLLMLingua (Jiang et al., 2024) and JinaAI Reader (JinaAI, 2024). For retrieval baselines (BM25/BGE/E5-Mistral), we retrieve evidence chunks and pass them to the same Generator $\mathcal{G}$ with the same chunk budget as in RES-RAG. For compression systems (LongLLMLingua/JinaAI Reader), we follow their standard pipelines to fit the same maximum prompt constraint. To be fair, we redeploy the above methods, and Table 1 reports detailed indicators.

Specifically, RES-RAG achieves a better quality-cost frontier: it improves answer correctness while using fewer evidence chunks, indicating that the learned retriever policy tends to select minimally sufficient evidence. This behavior is more pronounced on ambiguous data, where the ambiguity structure implies that blindly increasing retrieved

[5] https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507

Table 1: Comparison results of the RES-RAG method with other baselines. Optimal and suboptimal results are indicated by bold and underlined, respectively. The symbol † indicates that our method achieved optimality.

Method	ASQA		HotpotQA	TriviaQA	AmbigQA		2Wiki	MuSiQue
	EM	Hit@1	EM	EM	EM	Hit@1	EM	EM
`Llama-3.1-8B-Instruct-8k`								
BM25	3.25	<u>48.75</u>	<u>23.93</u>	62.29	17.80	46.60	14.00	13.25
BGE	<u>3.50</u>	43.50	20.40	63.91	20.42	50.79	13.50	12.25
E5-Mistral	3.25	43.56	**24.43**	61.50	**25.65**	<u>51.83</u>	21.50	**15.00**
LongLLMLingua	2.50	43.32	15.87	**66.25**	<u>23.04</u>	46.60	26.25	14.25
JinaAI Reader	3.00	45.25	14.86	58.25	16.75	49.46	**35.29**	13.46
RES-RAG	**4.75**†	**52.75**†	23.68	<u>66.06</u>	**25.65**	**57.59**†	<u>35.00</u>	<u>14.79</u>
`Llama-3.1-70B-Instruct-8k`								
BM25	1.75	51.50	<u>31.49</u>	73.92	23.56	56.54	45.25	18.50
BGE	2.25	57.25	27.96	74.00	22.51	47.12	39.75	16.75
E5-Mistral	<u>3.75</u>	55.50	<u>31.49</u>	**78.25**	<u>31.41</u>	66.49	49.00	<u>21.50</u>
LongLLMLingua	2.50	52.75	20.65	<u>77.48</u>	26.70	63.35	<u>49.75</u>	18.50
JinaAI Reader	2.25	<u>57.50</u>	18.89	61.25	**31.94**	<u>70.16</u>	29.34	14.75
RES-RAG	**4.00**†	**58.25**†	**31.74**†	75.25	<u>31.41</u>	**72.25**†	**56.03**†	**22.75**†

evidence often introduces distractors. RES-RAG mitigates this by selecting only the small subset of chunks that directly support at least one gold answer. In multi-hop, where reasoning requires composing evidence across documents, RES-RAG better allocates the budget, reducing redundant near-duplicate chunks. On Trivia-QA, which contains open-web pages with diverse structures and substantial noise, RES-RAG, rather than over-retrieving semantically similar but non-evidential chunks, learns evidence patterns aligned with the Generator's success signal. Dense retrievers still rely on fixed retrieval rules, and compression systems do not explicitly optimize whether the generator can correctly select supporting discrete evidence. In summary, the results in Table 1 answer **RQ1**: RES-RAG improves the quality-cost trade-off by learning to retrieve compact, highly useful evidence with a joint quality-cost objective.

4.3 Ablation Experiments Analysis

We analyze how the joint quality–cost reward in RES-RAG shapes evidence retrieval behavior. Table 2 reports ablation results. Starting from the full RES-RAG objective, we consider: (1) **top-k-only**; (2) **SFT-only**; (3) **Joint quality–cost reward**.

A fixed top-k strategy cannot adapt evidence size to instance difficulty: increasing k improves recall but often over-retrieves, while decreasing k under-retrieves on hard instances, yielding an inferior quality–cost frontier. SFT-only provides a strong warm start by imitating heuristic labels $\tilde{S}$, but it still does not optimize the utility–cost trade-off, so the retriever remains sensitive to heuristic biases and budget violations.

Optimizing utility only encourages over-retrieval since extra chunks are rarely penalized, increasing over-budget rates and weakening cost control. Overly emphasizing cost pushes the policy to retrieve too few chunks, reducing evidence cost but hurting answer correctness. These extremes confirm that both utility and cost are necessary for retrieval strategies. Compared to a linear penalty, the segmented penalty better distinguishes mild vs. severe budget violations, making the learning signal more robust and stabilizing the quality–cost frontier. Removing retrieval-consistency makes the reward sparse with respect to discrete index selection: many rollouts obtain similar answer rewards despite different evidence. The per-document F1 signal against $\tilde{S}$ densifies credit assignment, improving retrieval quality and budgeted selection behavior, which then translates to better downstream answer quality under the same budget.

Overall, the ablations answer **RQ2**: the joint reward design is crucial for learning effective, budget-

RES-RAG	ASQA		HotpotQA	TriviaQA	AmbigQA		2Wiki	MuSiQue
	EM	Hit@1	EM	EM	EM	Hit@1	EM	EM
`Llama-3.1-8B-Instruct-8k`								
(top-k) w/o SFT	2.56	43.25	18.43	56.25	17.34	48.95	23.75	13.25
(Warm-up) w SFT	3.75	44.50	21.16	61.75	21.47	51.83	27.25	14.75
(GRPO) w Joint reward	4.75	52.75	23.68	66.06	25.65	57.59	35.00	14.79

Table 2: Ablation experiment results.

feasible evidence retrieval.

4.4 Generality Experiment Studies

We test whether RES-RAG maintains effectiveness across (1) document structures with different noise profiles and layouts, and (2) generator configurations with different decoding behaviors and context capacities.

We compare RES-RAG on datasets with different sources: Wikipedia-style versus open-web pages. Across these variations, RES-RAG maintains an advantage in quality–cost trade-off, suggesting that the learned policy is not overfitting to a specific surface structure but instead captures transferable evidence selection behaviors. Although RES-RAG optimizes the Retriever, the reward is induced by the Generator $\mathcal{G}$, raising the question of whether the learned retriever is tightly coupled to a particular generator. We therefore swap $\mathcal{G}$ across different configurations while keeping the retriever fixed, and re-evaluate under the same chunk budget (Table 1). We observe that the benefits of RES-RAG persist across generator choices: absolute scores vary with generator capacity, indicating that the retriever policy learns evidence selection strategies that are broadly useful rather than exploiting idiosyncrasies of a single generator. The results in Table 1 answer **RQ3**: RES-RAG exhibits strong universality across both document structures and generator configurations, while retaining the core benefit of budget-aware, minimally sufficient evidence retrieval.

Dataset	Method	EvidTok	
		Avg	Max
ASQA	Top-k	318k	7259k
	RES-RAG	1k	7k
HotpoQA	Top-k	223k	5138k
	RES-RAG	2k	6k
TriviaQA	Top-k	53k	504k
	RES-RAG	2k	20k
AmbigQA	Top-k	288k	15631k
	RES-RAG	1k	7k
2Wiki	Top-k	11k	247k
	RES-RAG	1k	8k
MuSiQue	Top-k	78k	2359k
	RES-RAG	1k	10k

Table 3: Quality–cost comparison under identical settings. EvidTok counts retrieved evidence tokens. Avg represents the average number of tokens, and Max represents the maximum.

5 Conclusions

This paper presents RES-RAG, a budget-aware evidence retrieval framework for complex multi-document RAG scenarios. By formulating retrieval as a structured action that outputs per-document chunk indices, RES-RAG turns combinatorial evidence selection into a controllable decision problem that can be optimized for the retriever. We further introduce a joint quality–cost reward that unifies answer utility and evidence cost into a single scalar signal, enabling reinforcement learning to converge toward minimally sufficient evidence under a fixed context budget. Experiments across multiple open-domain QA benchmarks demonstrate that RES-RAG achieves a better trade-off between answer quality and evidence cost, and that indexed, traceable evidence selection reduces long-context noise while improving evidence density for downstream generation.

Limitations

The optimization signal relies on short-answer F1 and related anomaly penalties. This may not faithfully capture correctness for questions requiring richer reasoning, multi-hop justification, or long-form generation, and may encourage reward-

specific behaviors. In addition, although structured indexing improves controllability, the approach still depends on valid, parseable outputs. In other domains or under adversarial noise, format-compliance failures may increase or decrease robustness.

References

Kosuke Akimoto, Kunihiro Takeoka, and Masafumi Oyamada. 2024. Context quality matters in training fusion-in-decoder for extractive open-domain question answering. *arXiv preprint arXiv:2403.14197*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *ICLR*.

Scott Barnett, Stefanus Kurniawan, Srikanth Thudumu, Zach Brannelly, and Mohamed Abdelrazek. 2024. Seven failure points when engineering a retrieval augmented generation system. In *Proceedings of the IEEE/ACM 3rd International Conference on AI Engineering-Software Engineering for AI*, pages 194–199.

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

Maciej Besta, Ales Kubicek, Robert Gerstenberger, Marcin Chrapek, Roman Niggli, Patrik Okanovic, Yi Zhu, Patrick Iff, Michal Podstawski, Lucas Weitzendorf, and 1 others. 2024. Multi-head rag: Solving multi-aspect problems with llms. *arXiv preprint arXiv:2406.05085*.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. 2024. Guiding llms the right way: Fast, non-invasive constrained generation. *arXiv preprint arXiv:2403.06988*.

Wenhu Chen, Hanwen Zha, Zhiyu Chen, Wenhan Xiong, Hong Wang, and William Wang. 2020. Hybridqa: A dataset of multi-hop question answering over tabular and textual data. *arXiv preprint arXiv:2004.07347*.

Philipp Christmann, Svitlana Vakulenko, Ionut Teodor Sorodoc, Bill Byrne, and Adrià de Gispert. 2024. Retrieving contextual information for long-form question answering using weak supervision. *arXiv preprint arXiv:2410.08623*.

Yuqin Dai, Guoqing Wang, Yuan Wang, Kairan Dou, Kaichen Zhou, Zhanwei Zhang, Shuo Yang, Fei Tang, Jun Yin, Pengyu Zeng, and 1 others. 2025. Evinote-rag: Enhancing rag models via answer-supportive evidence notes. *arXiv preprint arXiv:2509.00877*.

Michiel de Jong, Yury Zemlyanskiy, Joshua Ainslie, Nicholas FitzGerald, Sumit Sanghai, Fei Sha, and William Cohen. 2022. Fido: Fusion-in-decoder optimized for stronger performance and faster inference. *arXiv preprint arXiv:2212.08153*.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. 2024. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade: Sparse lexical and expansion model for first stage ranking. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2288–2292.

Ge Gao, Jonathan D Chang, Claire Cardie, Kianté Brantley, and Thorsten Joachim. 2023a. Policy-gradient training of language models for ranking. *arXiv preprint arXiv:2310.04407*.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023b. 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, Yixin Dai, Jiawei Sun, Haofen Wang, and Haofen Wang. 2023c. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1).

Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Rajaram Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2g: Retrieve, rerank, generate. *arXiv preprint arXiv:2207.06300*.

Michael Günther, Isabelle Mohr, Daniel James Williams, Bo Wang, and Han Xiao. 2024. Late chunking: contextual chunk embeddings using long-context embedding models. *arXiv preprint arXiv:2409.04701*.

Yucan Guo, Miao Su, Saiping Guan, Zihao Sun, Xiaolong Jin, Jiafeng Guo, and Xueqi Cheng. 2025. Routerag: Efficient retrieval-augmented generation from text and graph via reinforcement learning. *arXiv preprint arXiv:2512.09487*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Zijian Hei, Weiling Liu, Wenjie Ou, Juyi Qiao, Junming Jiao, Guowen Song, Ting Tian, and Yi Lin. 2024. Dr-rag: Applying dynamic document relevance to retrieval-augmented generation for question-answering. *arXiv preprint arXiv:2406.07348*.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning

steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

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

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

Gautier Izacard and Edouard Grave. 2020. Leveraging passage retrieval with generative models for open domain question answering. *arXiv preprint arXiv:2007.01282*.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43.

Abhinav Java, Srivathsan Koundinyan, Nagarajan Natarajan, and Amit Sharma. 2025. Frugalrag: Learning to retrieve and reason for multi-hop qa. *arXiv preprint arXiv:2507.07634*.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677.

JinaAI. 2024. Reader-lm: Small language models for cleaning and converting HTML to Markdown. [Online; accessed 2024-10-05].

Mandar Joshi, Eunsol Choi, Daniel S Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611.

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

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp. *arXiv preprint arXiv:2212.14024*.

Kiseung Kim and Jay-Yoon Lee. 2024. Re-rag: Improving open-domain qa performance and interpretability with relevance estimator in retrieval-augmented generation. *arXiv preprint arXiv:2406.05794*.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. *arXiv preprint arXiv:2103.06332*.

Mandar Kulkarni, Praveen Tangarajan, Kyung Kim, and Anusua Trivedi. 2024. Reinforcement learning for optimizing rag for domain chatbots. *arXiv preprint arXiv:2401.06800*.

Moshe Lahmy and Roi Yozevitch. 2025. Replace, don't expand: Mitigating context dilution in multi-hop rag via fixed-budget evidence assembly. *arXiv preprint arXiv:2512.10787*.

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.

Junlong Li, Yiheng Xu, Lei Cui, and Furu Wei. 2021. Markuplm: Pre-training of text and markup language for visually-rich document understanding. *arXiv preprint arXiv:2110.08518*.

Xiaoxi Li, Guanting Dong, Jiajie Jin, Yuyao Zhang, Yujia Zhou, Yutao Zhu, Peitian Zhang, and Zhicheng Dou. 2025a. Search-o1: Agentic search-enhanced large reasoning models. *arXiv preprint arXiv:2501.05366*.

Yuan Li, Qi Luo, Xiaonan Li, Bufan Li, Qinyuan Cheng, Bo Wang, Yining Zheng, Yuxin Wang, Zhangyue Yin, and Xipeng Qiu. 2025b. R3-rag: Learning step-by-step reasoning and retrieval for llms via reinforcement learning. *arXiv preprint arXiv:2505.23794*.

Xiaoqiang Lin, Aritra Ghosh, Bryan Kian Hsiang Low, Anshumali Shrivastava, and Vijai Mohan. 2025. Refrag: Rethinking rag based decoding. *arXiv preprint arXiv:2509.01092*.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. *arXiv preprint arXiv:1905.13164*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Nan Duan, and 1 others. 2023. Query rewriting in retrieval-augmented large language models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2022. Mega: Moving average equipped gated attention. *arXiv preprint arXiv:2209.10655*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5783–5797.

Matin Mortaheb, Mohammad A Amir Khojastepour, Srimat T Chakradhar, and Sennur Ulukus. 2025. Re-ranking the context for multimodal retrieval augmented generation. *arXiv preprint arXiv:2501.04695*.

Jananee Muralidharan and Tiju Thomas. 2024. Deliberate problem-solving with a large language model as a brainstorm aid using a checklist for prompt generation. *The Journal of the Association of Physicians of India*, 72(5):89–90.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, and 1 others. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. 2024. Grammar-aligned decoding. *Advances in Neural Information Processing Systems*, 37:24547–24568.

Ronak Pradeep, Xueguang Ma, Xinyu Zhang, Hang Cui, Ruizhou Xu, Rodrigo Nogueira, and Jimmy Lin. 2020. H2oloo at trec 2020: When all you got is a hammer... deep learning, health misinformation, and precision medicine. *Corpus*, 5(d3):d2.

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

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. 2021. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367*.

Stephen Robertson, Hugo Zaragoza, and 1 others. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Yash Saxena, Ankur Padia, Mandar S Chaudhary, Kalpa Gunaratna, Srinivasan Parthasarathy, and Manas Gaur. 2025. Ranking free rag: Replacing re-ranking with selection in rag for sensitive domains. *arXiv preprint arXiv:2505.16014*.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models. *arXiv preprint arXiv:2109.05093*.

Xiaoyu Shen, Svitlana Vakulenko, Marco Del Tredici, Gianni Barlacchi, Bill Byrne, and Adrià de Gispert. 2022. Low-resource dense retrieval for open-domain question answering: A comprehensive survey. *arXiv preprint arXiv:2208.03197*.

Boheng Sheng, Jiacheng Yao, Meicong Zhang, and Guoxiu He. 2025. Dynamic chunking and selection for reading comprehension of ultra-long context in large language models. *arXiv preprint arXiv:2506.00773*.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Rich James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2023. Replug: Retrieval-augmented black-box language models. *arXiv preprint arXiv:2301.12652*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the domain adaptation of retrieval augmented generation (rag) models for open domain question answering. *Transactions of the Association for Computational Linguistics*, 11:1–17.

Ivan Stelmakh, Yi Luan, Bhuwan Dhingra, and Ming-Wei Chang. 2022. Asqa: Factoid questions meet long-form answers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8273–8288.

Dan Su, Xiaoguang Li, Jindi Zhang, Lifeng Shang, Xin Jiang, Qun Liu, and Pascale Fung. 2022a. Read before generate! faithful long form question answering with machine reading. *arXiv preprint arXiv:2203.00343*.

Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, and 1 others. 2022b. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975*.

Chihiro Taguchi, Seiji Maekawa, and Nikita Bhutani. 2025. Efficient context selection for long-context qa: No tuning, no iteration, just adaptive-k. *arXiv preprint arXiv:2506.08479*.

Xiaqiang Tang, Qiang Gao, Jian Li, Nan Du, Qi Li, and Sihong Xie. 2025. Mba-rag: a bandit approach for adaptive retrieval-augmented generation through

question complexity. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3248–3254.

SMTI Tonmoy, SM Zaman, Vinija Jain, Anku Rani, Vipula Rawte, Aman Chadha, and Amitava Das. 2024. A comprehensive survey of hallucination mitigation techniques in large language models. *arXiv preprint arXiv:2401.01313*, 6.

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

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and 1 others. 2023. Freshllms: Refreshing large language models with search engine augmentation. *arXiv preprint arXiv:2310.03214*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916.

Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, and 1 others. 2024b. Searching for best practices in retrieval-augmented generation. *arXiv preprint arXiv:2407.01219*.

Zheng Wang, Shu Teo, Jieer Ouyang, Yongjun Xu, and Wei Shi. 2024c. M-rag: Reinforcing large language model performance through retrieval-augmented generation with multiple partitions. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1966–1978.

Zhiruo Wang, Jun Araki, Zhengbao Jiang, Md Rizwan Parvez, and Graham Neubig. 2023. Learning to filter context for retrieval-augmented generation. *arXiv preprint arXiv:2311.08377*.

Zilong Wang, Zifeng Wang, Long Le, Huaixiu Steven Zheng, Swaroop Mishra, Vincent Perot, Yuwei Zhang, Anush Mattapalli, Ankur Taly, Jingbo Shang, and 1 others. 2024d. Speculative rag: Enhancing retrieval augmented generation through drafting. *arXiv preprint arXiv:2407.08223*.

Ji-Rong Wen, Ni Lao, and Wei-Ying Ma. 2004. Probabilistic model for contextual retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 57–63.

Shitao Xiao, Zheng Liu, Yingxia Shao, and Zhao Cao. 2022a. Retromae: Pre-training retrieval-oriented language models via masked auto-encoder. *arXiv preprint arXiv:2205.12035*.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Wen Xiao, Iz Beltagy, Giuseppe Carenini, and Arman Cohan. 2022b. Primera: Pyramid-based masked sentence pre-training for multi-document summarization. In *Proceedings of the 60th annual meeting of the association for computational linguistics (Volume 1: Long Papers)*, pages 5245–5263.

Jiawen Xie, Pengyu Cheng, Xiao Liang, Yong Dai, and Nan Du. 2024. Chunk, align, select: A simple long-sequence processing method for transformers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13500–13519.

Chunlei Xin, Shuheng Zhou, Xuanang Chen, Yaojie Lu, Huijia Zhu, Weiqiang Wang, Zhongyi Liu, Xianpei Han, and Le Sun. 2025. Aligning retrieval with reader needs: Reader-centered passage selection for open-domain question answering. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1000–1012.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. 2020. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv preprint arXiv:2007.00808*.

Lingnan Xu, Chong Feng, Kaiyuan Zhang, Liu Zhengyong, Wenqiang Xu, and Fanqing Meng. 2025. Equipping retrieval-augmented large language models with document structure awareness. *arXiv preprint arXiv:2510.04293*.

Peng Xu, Wei Ping, Xianchao Wu, Lawrence McAfee, Chen Zhu, Zihan Liu, Sandeep Subramanian, Evelina Bakhturina, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Retrieval meets long context large language models. *arXiv preprint arXiv:2310.03025*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 2369–2380.

Donghan Yu, Chenguang Zhu, Yuwei Fang, Wenhao Yu, Shuohang Wang, Yichong Xu, Xiang Ren, Yiming Yang, and Michael Zeng. 2021. Kg-fid: Infusing knowledge graph in fusion-in-decoder for open-domain question answering. *arXiv preprint arXiv:2110.04330*.

Yury Zemlyanskiy, Joshua Ainslie, Michiel de Jong, Philip Pham, Ilya Eckstein, and Fei Sha. 2021. Readtwice: Reading very large documents with memories. *arXiv preprint arXiv:2105.04241*.

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

Zhebin Zhang, Xinyu Zhang, Yuanhang Ren, Saijiang Shi, Meng Han, Yongkang Wu, Ruofei Lai, and Zhao Cao. 2023. Iag: Induction-augmented generation framework for answering reasoning questions. *arXiv preprint arXiv:2311.18397*.

Yuanhang Zheng, Peng Li, Wei Liu, Yang Liu, Jian Luan, and Bin Wang. 2024. Toolrerank: Adaptive and hierarchy-aware reranking for tool retrieval. *arXiv preprint arXiv:2403.06551*.

Siyuan Zhu, Chengdong Xu, Kaiqiang Ke, and Chao Yu. 2025. Context-picker: Dynamic context selection using multi-stage reinforcement learning. *arXiv preprint arXiv:2512.14465*.

A Chunking Process

We employ a structure-aware chunking pipeline to transform each HTML webpage into an ordered list of chunks for retrieval. Our design goal is two-fold: (1) preserve evidence-bearing structures (e.g., headings, tables, and lists) after conversion to Markdown; and (2) enforce a strict length constraint so that each chunk can be consumed reliably by the Retriever. We use MARKITDOWN to convert HTML to Markdown, preserving document structure (headings, lists, and tables) while removing many HTML-specific artifacts. For each document d_m, we output an ordered chunk sequence $\{c_i\}_{i=1}^{N_d}$, where each chunk stores: `type`$\in \{$`text`, `table`, `list`$\}$, a `heading_path` (e.g., `H1`$\rightarrow$`H2`$\rightarrow$`H3`), the chunk `token_len` measured by the Retriever tokenizer, and an `offset` indicating its position in the original document.

We first remove low-signal HTML elements, normalize whitespace, and drop hyperlink targets while keeping anchor text. The cleaned HTML is then converted to Markdown using MARKITDOWN. We treat each table/list block in Markdown as an atomic evidence unit and extract it into a standalone chunk. For the remaining narrative text, we chunk by heading boundaries: encountering a heading closes the current text chunk and starts a new one under the updated heading path. We prepend the heading path to the chunk content to retain discourse context.

We set a maximum chunk length $T_{\max} = 20480$ tokens (Retriever tokenizer). If a text chunk exceeds $T_{\max}$, we split it into multiple sub-chunks at sentence (or paragraph) boundaries to maintain readability. If a single table/list chunk exceeds $T_{\max}$, we further split it by rows (for tables) or by top-level items (for lists).

Finally, we merge table/list chunks and text chunks, sort them by `offset`, remove duplicates by normalized hashing, and reindex them to obtain $\{c_i\}_{i=1}^{N_d}$.

Algorithm 1 Structure-aware Chunking

Require: HTML webpage h, tokenizer τ, max tokens $T_{\max}$
Ensure: Ordered chunks $\{c_i\}_{i=1}^{N}$

$h^{\text{clean}} \leftarrow \text{CleanHTML}(h)$
$m \leftarrow \text{MarkItDown}(h^{\text{clean}})$
$\mathcal{B} \leftarrow \text{ParseBlocks}(m)$ ▷ linear blocks with type, heading_path, and offset
$\mathcal{C} \leftarrow [\,], buf \leftarrow \emptyset, p \leftarrow \emptyset, off \leftarrow 0$
for all $b \in \mathcal{B}$ in order **do**
  if $b.\text{type} =$ `heading` **then**
    if $buf \neq \emptyset$ **then**
      $\mathcal{C} \leftarrow \mathcal{C} \| \text{MakeChunk}(buf, \texttt{text}, p, off)$
      $buf \leftarrow \emptyset$
    end if
    $p \leftarrow b.\text{heading\_path}; \; off \leftarrow b.\text{offset}$
  else if $b.\text{type} \in \{\texttt{table}, \texttt{list}\}$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \| \text{MakeChunk}(b.\text{content}, b.\text{type}, p, b.\text{offset})$
  else
    $buf \leftarrow buf \| b.\text{content}$
  end if
end for
if $buf \neq \emptyset$ **then**
  $\mathcal{C} \leftarrow \mathcal{C} \| \text{MakeChunk}(buf, \texttt{text}, p, off)$
end if
$\mathcal{C} \leftarrow \text{SplitOversized}(\mathcal{C}, \tau, T_{\max})$
$\mathcal{C} \leftarrow \text{Deduplicate}(\mathcal{C})$
$\mathcal{C} \leftarrow \text{SortByOffset}(\mathcal{C})$
$\{c_i\}_{i=1}^{N} \leftarrow \text{ReindexFromOne}(\mathcal{C})$
return $\{c_i\}_{i=1}^{N}$

B Decoding Constraints

To prevent format violations and ensure that the retriever policy $\pi_{\theta_{\text{ref}}}$ always outputs a *parsable* JSON string, we apply guided decoding with a decoding constraint over the output space. The target output follows a fixed schema: a single JSON object that maps each document id $i \in \{1, \ldots, m\}$ to an array of chunk indices. We performance

the following syntactic constraints during generation: (1) the JSON object contains *exactly* m document fields; (2) each field value is an array of integers (chunk indices); (3) each index is 1-based and bounded by the corresponding document length, i.e., $s \in \{1, \ldots, M_i\}$ for document i with M_i chunks. With these constraints, every decoded action is valid by construction and can be directly parsed and evaluated by the downstream reward computation.

C Prompts

Retriever Prompt. The retriever is the *learned policy* optimized in RES-RAG. Given a query q and a set of chunked documents $\mathcal{X} = \{d_m\}_{m=1}^{M}$, it is instructed to select a *minimally sufficient* subset of evidence under a chunk budget, and to output the selection in a *structured* format. Specifically, the prompt uses a JSON object that maps each document to a list of 1-indexed chunk IDs (cf. decoding constraints in Appendix B), enabling deterministic parsing and budget accounting. The prompt template is shown in Figure 5.

Generator Prompt. The generator is a frozen LLM accessed via an online API. It is used only for reward computation: conditioned on the query q and the evidence selected by the retriever (and optional document metadata), it produces a short answer $\hat{a}$, which is then scored against references to obtain the answer-quality signal (F1) used in our joint reward. No parameters of the generator are updated during training. The prompt template is shown in Figure 6.

D Training parameters

We optimize the retriever with GRPO implemented in TRL. To stabilize policy updates, we regularize the retriever against a frozen reference policy via a KL penalty with coefficient β. During training, rollouts are sampled with temperature and nucleus sampling (top-p). Table 4 summarizes the key hyperparameters, grouped by context budget, optimization, sampling, and reward shaping.

E Infering Example

During inference, our trained model π_θ receives a query and multiple document chunks, and outputs the indices of the chunks associated with the query in each document. We map these indexes back to the text form of the chunks, then feed the query

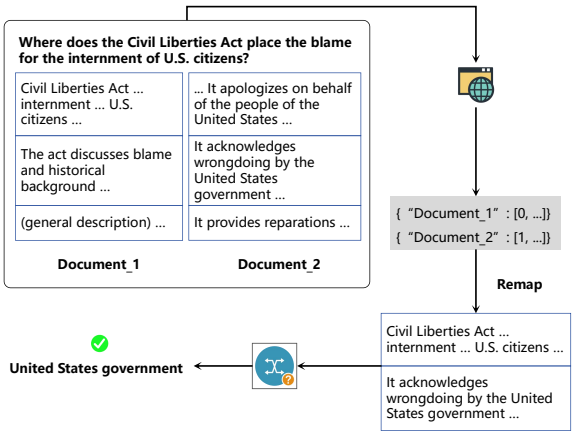

Figure 3: Inference example.

along with these texts and each document's JSON-LD into the generator, producing an answer. For consistency, we use the example in Figure 1, and this process is illustrated in Figure 3.

F Cost Study

We report the retrieval cost of RES-RAG and a similarity-threshold heuristic in terms of (1) the number of selected chunks and (2) the resulting token budget of the retrieved evidence. Figure 4 summarizes the average and maximum cost per query on six QA benchmarks.

Overall, the similarity-threshold baseline tends to over-retrieve, especially on datasets with ambiguous or multi-hop questions. For example, on ASQA, Ambig, and HotpotQA, it selects substantially more chunks on average (e.g., 22.93 vs. 4.60 on ASQA and 20.82 vs. 4.32 on Ambig), which further translates into a much higher token budget (e.g., 318k vs. 7.45k average tokens on ASQA and 288k vs. 6.86k on Ambig). More importantly, the heuristic exhibits severe tail cost: the maximum number of retrieved tokens can reach millions across several datasets, indicating a high risk of extreme-context inputs.

In contrast, RES-RAG consistently controls both the mean and the tail of retrieval cost. This behavior is expected because the policy is optimized under an explicit quality–cost objective, which encourages early stopping and avoids including long or redundant chunks unless they are necessary. Notably, on Musique, the chunk-count gap is moderate (8.71 vs. 4.92), while the token gap remains large (78k vs. 7.30k on average), suggesting that token cost is dominated not only by how many chunks are selected but also by how long the se-

Table 4: Key hyperparameters used in RES-RAG retriever training.

Name	Explanation	Value
`chunk-tokens`	Maximum token budget for each chunk	8192
`max-prompt-length`	Maximum input length to the retriever	8192
`max-completion-length`	Maximum generation length of the retriever action	1024
`batch-size`	Per-device batch size	1
`gradient-accumulation-steps`	Accumulation steps per update	4
β	KL regularization coefficient	0.05
`temperature`	Sampling temperature for training rollouts	1.0
top-p	Nucleus sampling threshold for training rollouts	0.8
W	Number of rollouts per instance	4
`lr`	Initial learning rate	1e-5
$\alpha_{\texttt{ans}}$	Weight of the F1 in the joint reward	2.0
$\alpha_{\texttt{ret}}$	Weight of the retrieval-consistency term	1.0
$R_{\texttt{min}}$	Lower bound for reward clipping	-5.0
$R_{\texttt{max}}$	Upper bound for reward clipping	5.0
η	Penalty coefficient for budget violation / over-retrieval	2.0

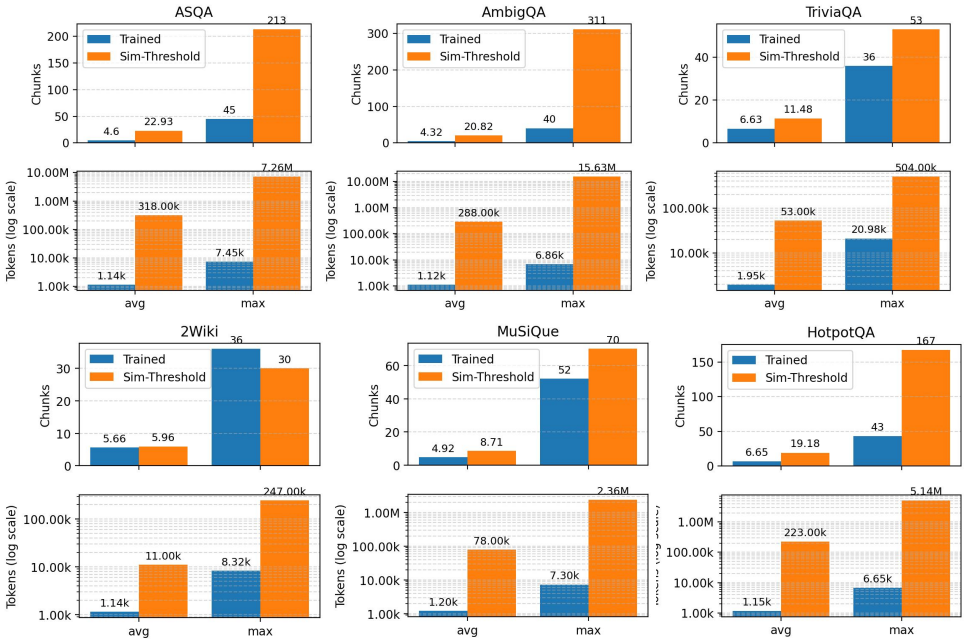

Figure 4: Comparison of costs between RES-RAG and top-k thresholding methods.

lected chunks are. On 2Wiki and TriviaQA, where evidence is typically more localized, the similarity-threshold baseline is less prone to over-retrieval, hence the average-cost gap is smaller; nevertheless, RES-RAG still reduces the risk of high-cost outliers.

These results highlight a practical advantage of RES-RAG: it provides a deployable mechanism to stabilize retrieval cost, reducing latency and the likelihood of exceeding the generator context window, while preserving answer quality.

system:

You are a Retrieval expert. Your sole responsibility is to receive a Query and Chunks from multiple documents, and then rigorously select the indexes(Integer) for the Chunks that are most relevant to the Query.

Input Instructions:
You will receive a Query string and a dictionary of Chunks from multiple documents, where the Chunks are in the following form:
{{
  "Document_i": [[chunk_0], ... [chunk_n]]
}}

Selection and Sorting Rules:
1. A Query typically corresponds to multiple documents, so you can select multiple Chunks from each document and output their indices (integers);
2. You need to carefully consider which Chunks in each document might contain the answer to the current Query. The Chunks you select will be input along with the Query to the downstream reader to generate the answer;

Output Format:
Regardless, you must adhere to the following output rules:
1. Your output must be a parsable JSON object;
2. Do not output any ``` tags;
3. Do not output any introductory words;
4. Your output must be the indexes(Integer) of the selected Chunks in each document;
5. The Chunks you select must be sorted in descending order of similarity to the Query.

Output Example:
{{
  "Document_0": [13, 2, 45, 7],
  "Document_1": [5, 17, 0, 3, 8, 6, 33],
}}

user:

Please select Chunks from each document that are relevant to the current Query.
The Query is: {question}
The document Chunks is as follows:
<JSON>
{html}
</JSON>

Figure 5: System and user prompt for the retriever model.

system:

You are a Reader expert. Your only task is to answer the given questions based only on the provided context paragraphs; output a list of short answers.
For each Query, you will receive multiple documents associated with it. Each document consists of multiple Chunks, all in Markdown format.

Answering Guidelines
1. Answer based on the context only; do not fabricate or introduce outside knowledge. If there is insufficient evidence, output [].
2. Short answers, using concise entity names/numbers/dates/phrases; no explanations, prefixes or suffixes (e.g., "The answer is").
3. When multiple answers are listed side by side (such as aliases, multiple entities, and multiple years), they are listed from high to low in order of confidence, and duplicates and noise are removed.
4. Normalization: Remove unnecessary spaces and punctuation; retain numbers/units of measure as is; format dates according to the original text; preserve capitalization of proper nouns.

Output format
Your output must be a parseable JSON array. Do not include any ``` or any other descriptive terms other than JSON objects. Your output should look like this:
["ans1", ...]

user:

Please output your answer directly in the form of a parsable JSON array based on the following passages:
The current Query is: {question}.
The current Chunks are: {item}.
The current JSON-LD are: {JSON-LD}.

Figure 6: System and user prompt for the generator model.