Q-RAG: LONG CONTEXT MULTI-STEP RETRIEVAL VIA VALUE-BASED EMBEDDER TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-Augmented Generation (RAG) methods enhance LLM performance by efficiently filtering relevant context for LLMs, reducing hallucinations and inference cost. However, most existing RAG methods focus on single-step retrieval, which is often insufficient for answering complex questions that require multi-step search. Recently, multi-step retrieval approaches have emerged, typically involving the fine-tuning of small LLMs to perform multi-step retrieval. This type of fine-tuning is highly resource-intensive and does not enable the use of larger LLMs. In this work, we propose Q-RAG, a novel approach that fine-tunes the Embedder model for multi-step retrieval using reinforcement learning (RL). Q-RAG offers a competitive, resource-efficient alternative to existing multi-step retrieval methods for open-domain question answering and achieves state-of-the-art results on the popular long-context benchmarks BabiLong and RULER for contexts up to 10M tokens.

1 INTRODUCTION

Large language models (LLMs) have achieved impressive results across a wide range of tasks (Novikov et al., 2025; Guo et al., 2025; Yang et al., 2025). However, they still face some several fundamental limitations such as static knowledge, computational inefficiency on long contexts, degraded performance caused by attention dilution, and hallucinations (Hsieh et al., 2024; Kuratov et al., 2024; Liu et al., 2025). Retrieval-Augmented Generation (RAG) is one of the most widely used techniques to address these issues (Yu et al., 2024).

RAG works by extracting only the most relevant parts from a large external corpus or context, such as newly added knowledge or lengthy texts. This allows LLMs to operate on shorter and more focused inputs, improving efficiency and output quality. Most current RAG methods rely on single-step retrieval. This setup performs well in relatively simple tasks like Needle-in-a-Haystack (Hsieh et al., 2024). Still, more complex problems require multi-step interaction with the context. Multi-step retrieval can be viewed as a form of search-based reasoning. There are several existing approaches to multi-step retrieval reasoning. One direction involves constructing a knowledge graph from the retrieved information (Ma et al., 2025; Li et al., 2024). These methods are often slow at inference time, since the LLM must process the entire context to build the graph for each new input. Another line of work uses LLM agents, which interleave RAG queries with LLM-generated instructions (Singh et al., 2025; Anokhin et al., 2024). These systems are sensitive to noisy or inaccurate retrieved passages, which may disrupt the generation of future queries. This shows the need for joint optimization of the retrieval and generation components. Recently, methods have emerged that fine-tune LLMs to interact more effectively with retrieval tools (Song et al., 2025; Jin et al., 2025; Chen et al., 2025). These methods tend to perform better, but they require expensive fine-tuning of the LLM itself. This makes them impractical for large models and limits accessibility for most researchers and practitioners.

In this work, we focus on developing a resource-efficient multi-step RAG approach using reinforcement learning. Instead of fine-tuning an LLM, we train an agent that performs retrieval directly in the latent space of text chunk embeddings. This allows us to learn a compact and efficient model using value-based RL methods.

Our approach achieves state-of-the-art results on long-context commonsense reasoning, multi-hop QA, and NIAH tasks with contexts up to 10 million tokens. It also performs competitively on

open-domain QA benchmarks such as Musique and HotpotQA (Yang et al., 2018;?), while being significantly faster and cheaper to train and run compared to existing multi-step RAG methods. Our contributions are the following:

- We propose a new method for training a multi-step retrieval agent using temporal difference reinforcement learning.
- We achieve state-of-the-art results on benchmarks that require commonsense reasoning and NIAH tasks over ultra long contexts (up to 10M tokens).
- We introduce a new way to incorporate temporal information into the multi-step embedder, enabling temporal reasoning during retrieval. Our temporal reasoning mechanism generalizes well to long contexts at inference time.

2 Related Works

There are several main directions for tackling complex retrieval scenarios on long context tasks.

A highly popular approach involves building fine-tuning free LLM Agents that combine off-the-shelf retrievers with LLMs, such as Search-o1 (Li et al., 2025). Many of these works further enhance retrieval quality by constructing large knowledge graphs over the context, which, while requiring little additional training, are extremely slow at inference due to the need for LLMs to process the entire context, e.g. GraphReader (Li et al., 2024), HippoRAG (Jimenez Gutierrez et al., 2024), AriGraph (Anokhin et al., 2024).

Another line of work fine-tunes LRMs to perform multi-step retrieval, allowing the model to generate intermediate search queries inside the reasoning for long contexts. The first work to apply this idea was IM-RAG (Yang et al., 2024), which fine-tuned the LLM with a frozen embedder using PPO (Schulman et al., 2017). More recent papers, such as R1-Searcher (Song et al., 2025), Search-R1 (Jin et al., 2025), RAG-RL (Huang et al., 2025), and ReSearcher (Chen et al., 2025), extended this direction by employing GRPO (Shao et al., 2024) for the task. Unlike these methods, which freeze the embedder and fine-tune the LLM, our approach fine-tunes only the embedder, allowing it to pair with LLMs of any size, including proprietary ones, while keeping fine-tuning efficient and inexpensive.

A different approach is to fine-tune the retriever itself using feedback from the LLM, as in REPLUG (Shi et al., 2024). This direction is most similar to ours, but REPLUG did not address multi-step reasoning or use reinforcement learning in this setting. BeamRetriever (Zhang et al., 2024) achieves state-of-the-art results on short-context QA by training a reranker for BeamSearch-style planning. In contrast, Q-RAG trains the embedder with reinforcement learning, enabling faster inference and better scalability to long contexts through efficient vector similarity instead of transformer-based trajectory scoring.

Extremely long-sequence processing is demonstrated by models that combine recurrence with Transformer architecture. The Mamba family of state space models (Gu & Dao, 2024) replaces attention with structured recurrent dynamics, offering linear-time scalability and strong performance on long sequences, though often at the cost of weaker in-context learning and less expressive token-to-token interaction compared to Transformer-based architectures. The Recurrent Memory Transformer (RMT) (Bulatov et al., 2022) introduces segment-level recurrence by passing memory tokens between fixed-size segments, enabling Q&A on sequences up to 10M tokens. Titans (Behrouz et al., 2024) frames recurrent memory training as a meta-learning problem, showing scaling beyond 2M tokens. Building on this idea, ATLAS (Behrouz et al., 2025) increases memory capacity, achieving better long-context performance than both RMT and Titans. The Associative Recurrent Memory Transformer (ARMT) (Rodkin et al., 2024) employs quasi-linear, associative attention in each layer and attains the best long-context scores among recurrent models. Our approach outperforms all of these models on contexts beyond 1M tokens while belonging to a different class of methods.

LongRoPE2 (Shang et al., 2025) tackles the positional encoding bottleneck, extending the effective context window of pre-trained LLMs to 128K tokens while retaining short-context performance through RoPE rescaling and mixed-window training.

3 METHODS

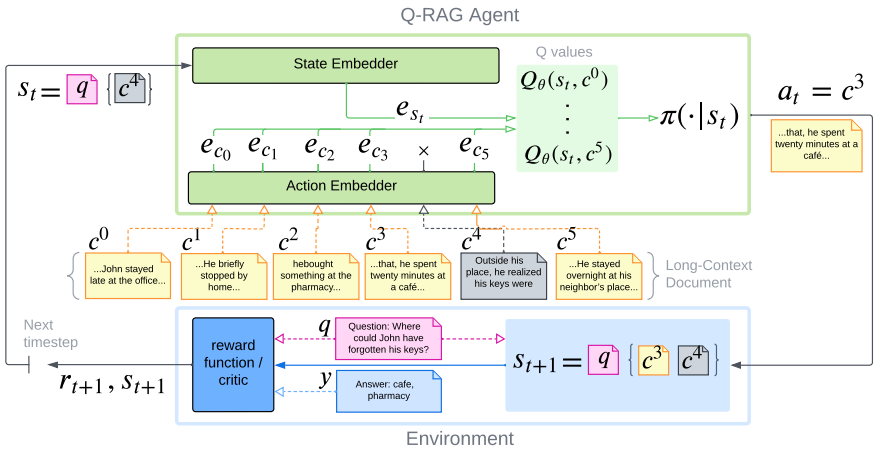


Figure 1: Q-RAG agent interacts with multi-step retrieval environment. The starting state s_0 contains the initial query q. At the start of the episode, the agent embeds all chunks of the long context $\mathbb{C}$. At each step t, the agent computes a vector embedding of the current state s_t, which includes q and all previously selected chunks. For every chunk $c^i \in \mathbb{A}_t$, the utility of retrieving it is evaluated by the Q-function $Q_\theta(s_t, a = c^i)$. The policy π_θ selects the next chunk from $\mathbb{A}_t$ with probability proportional to its $Q_\theta(s_t, c^i)$ value.

3.1 PRELIMINARIES

Let $\mathcal{D}$ be a dataset of triples $(\mathbb{C}, q, y)$, where $\mathbb{C}$ is a long context, q is an initial query, and y is the gold answer. The query q can be either a user question about $\mathbb{C}$ or a generated claim whose factuality or consistency with earlier parts of $\mathbb{C}$ must be verified. We assume $\mathbb{C}$ is pre-segmented into non-overlapping[1] text chunks $\mathbb{C} = \{c^{(i)}\}_{i=1}^m$ in document order. The agent's goal is to identify the information in $\mathbb{C}$ that is missing from q but necessary to produce the correct answer y. We model multi-step retrieval as a finite-horizon Markov Decision Process, or MDP $(\mathbb{S}, \mathbb{A}, p, r, \gamma)$, where $\mathbb{A}$ is the action space, $\mathbb{S}$ is the state space, r is the reward function, p is the (deterministic) transition function, and $\gamma \in [0, 1]$ is the discount factor. At step $t = 0$, the action set is $\mathbb{A}_0 = \mathbb{C}$, where an action $a_t \in \mathbb{A}_t$ selects one chunk. At later steps, previously selected chunks are removed so $\mathbb{A}_t = \mathbb{C} \setminus \{a_0, \ldots, a_{t-1}\}$. Superscripts indicate document positions and subscripts indicate episode timesteps. The notation a^i (equivalently $c^{(i)}$) denotes the chunk/action at position i in the document; selecting the chunk with index i at step t is written a_t^i. Symbols c and a are used interchangeably, depending on context.

States are ordered lists that always begin with the query, $s_t = \mathrm{ord}([q, a_0, \ldots, a_{t-1}])$, where $\mathrm{ord}(\cdot)$ sorts by the original document order to avoid permutation ambiguity; the initial state contains only the query, $s_0 = [q]$. Transitions are deterministic, $p(s_t, a_t) = \mathrm{ord}([q, a_0, \ldots, a_{t-1}, a_t])$. An episode terminates either when a step budget T is reached or when a special STOP action is taken.

When supervision provides a set of support facts $F^\star \subseteq C$, we use a sparse terminal reward: the reward is 0 at all intermediate steps, and at the end of the episode it is 1 if all support facts are included in the final state (otherwise 0). When only answer supervision is available, one could instead use an LLM to generate $\hat{y}$ from the final state and define a terminal reward via an answer-quality metric (e.g., exact match or F1). In this work we do not pursue LLM-based rewards; all reported experiments rely on the support-fact signal, and exploring LLM-based reward design is left for future work.

[1] Chunk overlapping may complicate the explanation but does not affect our proposed solution.

3.2 VALUE-BASED RL FOR EMBEDDER FINE-TUNING

Action selection in multi-step retrieval is performed by a value-based agent. Specifically, maximum-entropy reinforcement learning (Ziebart, 2010; Haarnoja et al., 2018) is adopted together with the corresponding definitions of the soft Q^π and V^π value functions for a policy π:

$$Q^\pi(s,a) = r(s,a) + \gamma V^\pi(s' = p(s,a)) \tag{1}$$

$$V^\pi(s) = \mathbb{E}_{a\sim\pi(\cdot|s)}\left[Q^\pi(s,a) - \alpha \log \pi(a|s)\right] \tag{2}$$

Here, $\alpha > 0$ is a temperature that controls the strength of exploration. This choice is primarily motivated by the need for effective exploration in the long-context multi-step retrieval environment. In Q-RAG, Q function is approximated using two embedders for states and actions. The state embedder $E_s(s_t;\theta_1) \in \mathbb{R}^d$ produces vector embedding for the current state s_t, while the action embedder $E_a(a^i, i; \theta_2) \in \mathbb{R}^d$ employ rotary position embeddings to encode both the candidate chunk content and its document-position index i. Q values are then estimated by an inner product between two embeddings: $Q_\theta(s, a^i) = \langle E_s(s;\theta_1), E_a(a^i, i; \theta_2)\rangle$. This factorization is theoretically grounded; we derive its convergence guarantees with explicit rates in Appendix A. Given Q_θ, the chunk selection probability is computed using a Boltzmann policy:

$$\pi(a_t|s_t) = \frac{\exp\frac{1}{\alpha}\left(Q_\theta(s_t,a_t) - q\right)}{\sum_{a\in\mathcal{A}_t}\exp\frac{1}{\alpha}(Q_\theta(s_t,a) - q)} \tag{3}$$

with $q = \max_{a\in\mathcal{A}_t} Q_\theta(s_t,a)$ and temperature α annealed from an initial value to zero during training (proportionally to the learning rate).

As the backbone Temporal Difference learning algorithm, we adopt the recent PQN method by Gallici et al.. Compared to DQN (Mnih et al., 2015), PQN removes the need for a replay buffer. In our setting with a large number of chunks, a replay buffer would require re-embedding all document chunks for each sample drawn from the replay buffer to estimate V/Q values for subsequent states s_{t+1}. Which significantly slows the training process and increases memory space requirements. Using PQN enables an on-policy value-based training that avoids these costs. The key departures in Q-RAG, relative to the original PQN backbone, are the use of soft value functions and target networks. Ablation results demonstrating the benefit of these choices are reported in Section 4.5.

As the training target, rather than the one-step return (see r.h.s. in Eq. 1), a λ-return is used to improve stability and learning speed:

$$G_t^\lambda = (1-\lambda)\sum_{n=1}^{T-t-1}\lambda^{n-1}G_{t:t+n} \;+\; \lambda^{T-t-1}G_t,$$

where $G_{t:t+n} = \sum_{k=1}^{n}\gamma^{k-1}r_{t+k} + V_{\theta'}(s_{t+n})$. The approximation of the state value function can be computed from Q values in the case of discrete actions:

$$V_{\theta'}(s_t) = \alpha \log \sum_{a\in\mathcal{A}_t}\exp\left(\frac{Q_{\theta'}(s_t,a)}{\alpha}\right) \tag{4}$$

Here θ' denotes slowly updated target network parameters. The model parameters θ are finetuned to minimize the mean squared error to the λ-returns:

$$\mathcal{L}_Q = \mathbb{E}[(Q_\theta(s_t,a_t) - G_t^\lambda)^2] \tag{5}$$

The Q-RAG pseudocode is presented in Algorithm 1.

3.3 TEMPORAL REASONING FOR LONG-CONTEXT SEARCH

When dealing with narrative text, the information contained in a text chunk c may be insufficient to determine whether c helps us answer the question q. For example, we may need to know what happened before some specific event. A standard retriever can find several relevant text chunks that specify the character's location, but choosing the correct one can be impossible without taking into account temporal information. To address this, we propose a *relative postional encoding* of chunks that explicitly encodes their position with respect to the facts already extracted into the state. At step t, let $S_t = \{i_1 < \cdots < i_k\}$ be the (sorted) document indices of selected chunks and $\mathbb{A}_t$ the set

Algorithm 1 Q-RAG

Hyperparameters:
  Environments count K, retrieval steps T, temperature α, TD parameter λ, EMA τ.
Initialize:
  State embedder $E_s(s;\theta_1)$
  Action embedder $E_a(a^i, i;\theta_2)$ with position i
  Critic $Q_\theta(s,a^i) = E_s(s;\theta_1)^T E_a(a^i,i;\theta_2)$
  Critic target $Q_{\theta'}(s,a^i)$
procedure COMPUTETARGETS($\{s_t,a_t,r_t,v_t\}_{t=1}^{T+1}$)
    Initialize λ-returns $G_T = r_T + \gamma v_{T+1}$
    for $t = T-1$ **downto** 1 **do**
      $G_t = r_t + \gamma\big[(1-\lambda)v_{t+1} + \lambda G_{t+1}\big]$
    end for
    return $\{G_t\}_{t=1}^T$
end procedure
Training (one update step)
for env $k \in 1,\dots,K$ **in parallel do**
    $s_1, \mathcal{A}_1 =$ ResetQueryAndContext()
    Compute $E_a = E_a(\mathcal{A};\theta)$ and $E'_a = E_a(\mathcal{A};\theta')$
    for step $t \in 1,\dots,T+1$ **do**
      $a_t \sim \text{softmax}_{a\in\mathcal{A}_t} \frac{1}{\alpha} E_s(s;\theta)^T E_a$
      $v_t = \alpha \log \sum_{a\in\mathcal{A}} \exp \frac{1}{\alpha} E_s(s;\theta')^T E'_a$
      $r_t =$ ComputeReward(s_t,a_t)
      $s_{t+1} =$ concatenate(s_t,a_t)
      $\mathcal{A}_{t+1} = \mathcal{A}_t \setminus \{a_t\}$
    end for
    $\mathcal{B} = \{s_t,a_t,r_t,v_t\}_{t=1}^{T+1}$
    $\{G_t^k\}_{t=1}^T =$ ComputeTargets$(\mathcal{B})$
end for
$\nabla\mathcal{L}_Q = \frac{1}{TK}\sum_{k=1}^K \sum_{t=1}^T \nabla_\theta (Q_\theta(s_t^k,a_t^k) - G_t^k)^2$
Update θ using $\nabla\mathcal{L}_Q$
Update target parameters: $\theta' \leftarrow \tau\theta + (1-\tau)\theta'$

of available actions. The indices in S_t partition the document into $k{+}1$ disjoint intervals: "before the earliest selected fact", "between consecutive selected facts", and "after the latest selected fact." The relative positional mapping $\rho_t : \mathbb{N} \to \mathbb{R}^+$ assigns to every original chunk index a real-valued index that (i) identifies the interval it belongs to and (ii) preserves the relative order between chunks. This mapping makes explicit *between which extracted facts* a chunk lies, while remaining invariant to global shifts of absolute positions.

Formally, the interval boundaries are defined as $b_0{=}1$, $b_j{=}i_j$ for $j{=}1{:}k$, and $b_{k+1}{=}m{+}1$ for $\mathbb{C} = \{c^{(i)}\}_{i=1}^m$. To compute relative index $\rho_t(i)$ for a chunk c^i, find the unique j such that $b_j \le i < b_{j+1}$ and set

$$\rho_t(i) \;=\; j\,\delta \;+\; \ell\,\frac{i-b_j}{b_{j+1}-b_j}, \tag{6}$$

where $\delta > 0$ is the inter-interval step and $\ell \in (0,\delta)$ controls the within-interval resolution (e.g., $\delta{=}10$, $\ell{=}9$ in our experiments). In the action embedder, the absolute position is replaced by the relative one,

$$E_a\big(a^i,\, i;\theta_2\big) \;\Rightarrow\; E_a\big(a^i,\, \rho_t(i);\theta_2\big), \tag{7}$$

which allows the Q-function to exploit the spatial relation of candidates to already retrieved evidence while retaining local order within each interval. This design allows the retrieval agent to perform strongly not only on fact-finding over disjoint document collections, but also on long-form narrative tasks, enabling Q-RAG to compete with recurrent transformers (Bulatov et al., 2022; Rodkin et al., 2024; Behrouz et al., 2025; 2024) and other long context approaches.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

We evaluate our approach, Q-RAG, on tasks that cover commonsense reasoning, temporal reasoning, a bunch of needle in a haystack tasks and open-domain multi-hop question answering tasks on context lengths that range from 4k tokens to 10M tokens per sample. For commonsense and temporal reasoning we use **BabiLong** benchmark (Kuratov et al., 2024), for Needle-in-a-Haystack we use **RULER** benchmark Hsieh et al. (2024). For open-domain multi-hop QA we use **HotpotQA** Yang et al. (2018), **Musique** Trivedi et al. (2022) and **RULER** benchmarks. BabiLong and RULER require long contexts. Musique and HotpotQA use short contexts.

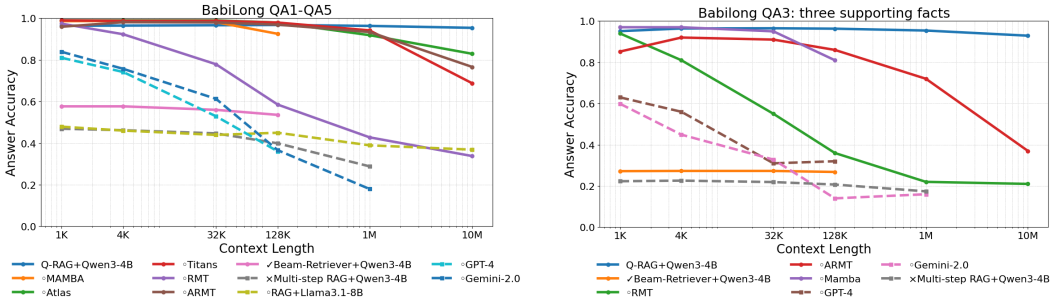
Baselines differ by task. Computing a uniform set of baselines across all datasets is difficult and time-consuming. Many methods do not release code. Some methods were evaluated only on some of these datasets. Even when the tasks match, the experimental settings often differ for the same benchmarks. Some baselines provide code but require heavy resources (e.g., at least 8×A100 GPUs Jin et al. (2025); Song et al. (2025); Huang et al. (2025)) to fine-tune, which are unavailable for us. Therefore, we report three types of baselines, and we mark each baseline in tables accordingly:

- $^{\times}$ **Ablation**: baselines that test the effectiveness of our proposed modifications.
- $^{\checkmark}$ **Reproduced**: baselines that we finetuned and/or evaluated on our datasets using released code or publicly available checkpoints.
- $^{\circ}$ **Reported**: baselines whose scores we take directly from the original papers.

4.2 COMMONSENSE REASONING ON ULTRA-LONG CONTEXTS

On the BabiLong Kuratov et al. (2024) benchmark, we compared our method with the state-of-the-art long-context processing approaches, including Titans Behrouz et al. (2024), Atlas Behrouz et al. (2025), ARMT Rodkin et al. (2024), RMT Bulatov et al. (2022), as well as proprietary LLMs and LLM-based agents. The results for most of these baselines were taken directly from the respective original papers. As shown in Figure 2b, our approach achieves the highest average performance on BabiLong in ultra-long contexts ranging from 1 to 10 million tokens, demonstrating superior generalization to long contexts compared to other specialized long-context methods.

In Figure 2a, we present separate results for the QA3 subtask, which is the hardest subtask in the BabiLong benchmark, which specifically requires the multistep search of at least 3 different facts and temporal reasoning. Experimental results show that the majority of models perform worst on the QA3 subtask. As the results indicate, alternative long-context approaches show even greater performance degradation on this task with increasing context length. In contrast, Q-RAG shows virtually no degradation, with the largest performance gap over all baselines observed on this most challenging subtask. We additionally fine-tuned the Beam-Retriever baseline specifically on the QA3



(a) (b)

Figure 2: Comparison of answer accuracy on the long-context benchmark BabiLong. Solid lines denote methods fine-tuned on the BabiLong, while dashed lines denote zero-shot methods. **a)** Average performance across tasks Q1–QA5. **b)** Performance on the hardest task, QA3, which requires the longest reasoning chain and temporal awareness.

subtasks, given its strong performance on open-domain QA datasets. However, this method failed to solve the task. Note that some methods, such as Titans Behrouz et al. (2024) and Atlas Behrouz et al. (2025), are absent from the Figure as they did not report detailed breakdowns by a subtask.

4.3 Needle in a Haystack and Long Context QA

While reasoning tasks are crucial for evaluating advanced retrieval systems, a substantial portion of real-world applications reduces to Needle-in-a-Haystack (NIAH) problems, making it equally important that models deliver consistently strong performance on these tasks.

RULER is a dataset that includes many long-context tasks. Most of these tasks follow the NIAH formulation. The NIAH setup evaluates the ability to retrieve a specific "needle" from a long distracting "haystack".

For RULER benchmark we use Titans Behrouz et al. (2024), Atlas Behrouz et al. (2025), Mamba2 Waleffe et al. (2024), and LongRope2 Shang et al. (2025) as baselines. Titans, Atlas are recurrent transformers. Mamba2 is a state space model (SSM) that combines transformer components with SSM. LongRope2 is a method for extending the effective context window of LLMs. All methods were fine-tuned either directly on RULER (Titans, Atlas, Mamba2) or on related synthetic NIAH-style datasets (LongRope2). Q-RAG was also fine-tuned on the NIAH subtasks. For the Multi-hop QA RULER subtask, Q-RAG was fine-tuned on HotpotQA and evaluated on the Multi-hop QA subtask out-of-distribution.

The results are shown in Table 1. Q-RAG achieves near-perfect performance on all NIAH subtasks. Q-RAG embedder was trained on 4K-length documents and generalizes to context lengths up to 1M tokens without loss of accuracy. On the Multi-hop QA subtask, Q-RAG shows significantly better

Table 1: Results on the RULER benchmark, evaluating long-context retrieval performance across various context lengths. **S** (Single-needle): Find one value for one key. **MK** (Multi-keys): Find one value for one key among many. **MV** (Multi-values): Find all values for one key. **MQ** (Multi-query): Answer multiple questions over the context. **MH QA**: open domain multi-hop question answering.

Length	**Methods**	S			MK			MV	MQ	NIAH Avg.	MH QA
		1-st	2-nd	3-rd	1-st	2-nd	3-rd				
4K	°Titans	98.4	99.8	89.4	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	°Atlas	99.2	100	90.6	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	°Mamba2-Hybrid	100	100	95.7	89.5	95.5	96	97.9	97.6	96.5	48.8
	°LongRoPe2-8B	100	100	99	100	100	100	99	99.7	99.7	60
	✓Beam-Retriever	100	100	98	98	98	97	98	99	98.5	28.3
	Q-RAG	100	100	100	100	100	100	100	100	**100**	**67**
16K	°Titans	96.2	80.2	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	°Atlas	97	84	n/a	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	°Mamba2-Hybrid	100	100	81.5	92	92.2	83	89.8	90.2	91.1	44
	°LongRoPe2-8B	100	100	100	99	100	98	95	98.2	98.8	58
	✓Beam-Retriever	100	100	97	96.5	96	95	80	98	95.3	28.3
	Q-RAG	100	100	100	100	100	100	100	100	**100**	**67**
32K	°Mamba2-Hybrid	100	100	96.7	84	76.5	81.5	84.3	80.9	88.0	38.5
	°LongRoPe2-8B	100	100	100	99	98	100	98	96.2	98.9	55
	Q-RAG	100	100	100	100	100	100	100	100	**100**	**67**
128K	°LongRoPe2-8B	100	100	99	96	91	94	96.5	97	96.7	50
	Q-RAG	100	100	100	100	100	100	100	100	**100**	**62**
1M	Q-RAG	100	100	100	100	98.5	99.0	100	100	**99.7**	**57**

results than all our baselines at all context lengths we consider. Some degradation with increasing context length starts only from 128K.

4.4 Open-domain Question Answering

For our experiments on the HotPotQA and Musique datasets, we compared our method against several strong baselines. The first baseline is Beam Retriever, which enables multi-step retrieval by training a model to score sequences of retrieved chunks. During evaluation, Beam-Retriever is given the oracle number of supporting facts (i.e., the gold hop count) and always retrieves exactly that many facts. Although this approach is slower than traditional retrieval methods and does not scale well to longer contexts, it achieves state-of-the-art results on HotPotQA. Another baseline we considered is SearchR1, a recent method from a family of approaches that train the LLM itself to compose text queries for multi-step retrieval. Additionally, we evaluated the performance of LLM-agent-based methods, including GraphReader. Q-RAG and Beam-Retriever were fine-tuned on HotPotQA and evaluated on Musique for out-of-distribution testing. Baseline numbers were taken directly from the corresponding papers. Missing entries indicate metrics not reported by the original authors.

The comparison results are presented in Table 2. Our method achieves fact retrieval accuracy on par with Beam Retriever, surpasses all other baselines on HotPotQA, and matches the performance of full-LLM-tuning Search-R1 while outperforming all alternatives on the out-of-distribution Musique dataset, resulting in the best overall performance across benchmarks. Results also include another Q-RAG version *Plan Q-RAG* that combines Q-RAG value function and beam search based planning (see Appendix D). Plan Q-RAG showed similar performance to vanilla Q-RAG. For both methods involving retrieval mechanism fine-tuning (Q-RAG and Beam Retriever), we used the QwQ-32B model to produce the final answer.

Table 2: Comparison of methods on HotPotQA and Musique benchmarks. Bold text and underline denote the best and second best scores respectively.

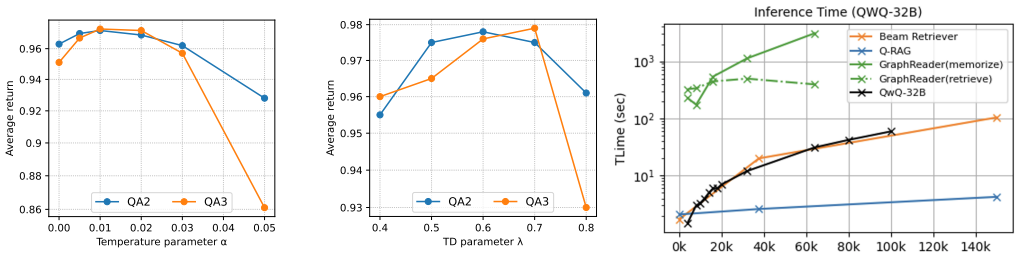
	HotPotQA				Musique (OOD)				Avg	
Methods	**Fact F1**	**Fact EM**	**Ans F1**	**Ans EM**	**Fact F1**	**Fact EM**	**Ans F1**	**Ans EM**	**Ans F1**	**Ans EM**
Finetuned on HotPotQA										
Plan Q-RAG	<u>0.95</u>	<u>0.91</u>	<u>0.76</u>	<u>0.60</u>	0.69	0.53	<u>0.51</u>	0.36	**0.64**	**0.48**
Q-RAG	0.93	0.89	0.76	0.59	**0.71**	**0.55**	**0.52**	<u>0.37</u>	**0.64**	**0.48**
✓ Beam-Retriever	**0.97**	**0.94**	**0.77**	**0.61**	0.61	0.36	0.40	0.27	0.59	0.44
✓ Search-r1	0.81	0.66	0.65	0.52	**0.71**	**0.55**	0.51	**0.39**	0.58	0.46
° RAG-RL	0.82	–	0.69	–	0.65	–	0.47	–	0.58	–
× Multi-step RAG w.o. FT	0.73	0.54	0.65	0.50	0.51	0.30	0.40	0.27	0.53	0.39
Zero Shot methods										
✓ GraphReader	–	–	0.46	0.24	–	–	0.40	0.20	0.43	0.22
✓ Single step RAG	–	–	0.53	0.39	–	–	0.28	0.17	0.41	0.28

4.5 Ablation Study

To assess the impact of the architectural choices in Q-RAG, an ablation study was conducted on the BabiLong-QA3 task. This benchmark was selected because it is among the most challenging long-context tasks used in the experiments and it supports evaluation at arbitrary context lengths. The following baselines were compared against Q-RAG:

Multi-step RAG w.o. FT. This baseline reproduces the full Q-RAG retrieval pipeline and uses the same state and action embedders, but relies on their original pretrained weights without any reinforcement learning fine-tuning. This setting tests whether RL fine-tuning of the embedders is beneficial for multi-step retrieval quality.

Multi-step RAG w. SFT. This baseline applies supervised fine-tuning using ground-truth support facts as supervision. The loss follows the objective used in BeamRetriever for trajectory supervision,



(a) (b) (c)

Figure 3: Ablation for (a) policy entropy coefficient (α) in soft Q function and (b) for λ-return parameter. Inference runtime comparison (c), context length, tokens on x-axes.

adapted to the multi-step retrieval setting. This setting isolates the effect of RL by comparing it to supervised learning on the same supervision signal.

Q-RAG w.o. target. This variant removes target networks from the PQN-based value learning, following the original PQN recipe without target parameters. It measures the contribution of target networks to stability and performance in the Q-RAG training loop.

Q-RAG w.o. Soft-Q. This variant replaces the maximum-entropy (soft) value functions with standard (non-entropy-regularized) Q-learning objectives. It evaluates the effect of entropy regularization and the soft value formulation on retrieval performance.

All baselines were evaluated with three random seeds. Table 3 reports results at a 32k-token context length on QA3. Figure 3 shows the sensitivity of Q-RAG to the λ-return parameter and the temperature α (the strength of entropy regularization) on QA2 and QA3.

Table 3: Ablation results on BabiLong QA3. Table shows F1 score for support facts retrieval. All values are averaged over 3 runs with different seeds.

Method	**1K**	**4K**	**32K**	**128K**	**1M**
Q-RAG	97.8 ± 0.17	97.4 ± 0.14	97.1 ± 0.08	96.8 ± 0.08	96.5 ± 0.16
$^{\times}$Q-RAG w.o. Soft-Q	95.9 ± 0.70	95.5 ± 0.80	94.5 ± 0.50	94.0 ± 0.30	93.3 ± 0.45
$^{\times}$Q-RAG w.o. Target	79.2 ± 26.0	78.1 ± 26.6	77.6 ± 27.2	77.4 ± 27.3	75.9 ± 28.2
$^{\times}$Multi-Step RAG w. SFT	20.33 ± 0.32	20.87 ± 0.35	20.10 ± 0.20	18.30 ± 0.36	—
$^{\times}$Multi-Step RAG w.o. FT	15.52 ± 0.11	16.38 ± 0.10	15.51 ± 0.16	15.34 ± 0.12	—

5 Conclusion

This work introduced Q-RAG, a resource-efficient method for multi-step retrieval trained with reinforcement learning directly in the latent space of text-chunk embeddings. Across long-context benchmarks (e.g., *BabiLong*, *RULER*) and open-domain QA datasets (e.g., *Musique*, *HotpotQA*), Q-RAG attains state-of-the-art or highly competitive results. Its advantage over baselines widens as context length grows, and performance shows minimal degradation even at ultra-long scales.

A key practical benefit is compute efficiency: all training was performed on a single A100 GPU with 80 GB memory, whereas recent RL-based multi-step retrievers such as Search-R1/R1-Searcher typically report training on clusters of about eight A100 GPUs. By fine-tuning only the embedder while keeping the LLM frozen, Q-RAG remains easy to pair with powerful pre-trained or proprietary LLMs, enabling efficient training, flexible deployment, and strong retrieval over very long contexts.

Looking ahead, promising directions include using structured LLM feedback as a reward signal, strengthening compositional and temporal reasoning directly in the embedding space, and exploring tighter integration with generation while preserving the method's efficiency and scalability.

6 Reproducibility Statement.

We make all results reproducible by providing a code package with exact configs and run scripts; **all code is included in the supplementary materials.** The package includes utilities to download and *minimally* preprocess the public *HotPotQA* and *MuSiQue* datasets and to re-run every experiment and table with fixed random seeds (Fact F1/EM, Answer F1/EM). We fine-tune only publicly available embedders — `multilingual-e5-large` and `facebook/contriever` — strictly following the hyperparameters and schedules described in Appendix G. All reported runs are reproducible on a single GPU; our main experiments were executed on one `A100-80GB` device. The repository contains evaluation scripts that reproduce the reported tables without modification; full implementation specifics are referenced from Appendix G and the supplementary materials.

References

Petr Anokhin, Nikita Semenov, Artyom Sorokin, Dmitry Evseev, Andrey Kravchenko, Mikhail Burtsev, and Evgeny Burnaev. Arigraph: Learning knowledge graph world models with episodic memory for llm agents. *arXiv preprint arXiv:2407.04363*, 2024.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time. *arXiv preprint arXiv:2501.00663*, 2024.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Atlas: Learning to optimally memorize the context at test time. *arXiv preprint arXiv:2505.23735*, 2025.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory transformer. *Advances in Neural Information Processing Systems*, 35:11079–11091, 2022.

Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z Pan, Wen Zhang, Huajun Chen, Fan Yang, et al. Learning to reason with search for llms via reinforcement learning. *arXiv preprint arXiv:2503.19470*, 2025.

Matteo Gallici, Mattie Fellows, Benjamin Ellis, Bartomeu Pou, Ivan Masmitja, Jakob Nicolaus Foerster, and Mario Martin. Simplifying deep temporal difference learning. In *The Thirteenth International Conference on Learning Representations*.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces, 2024. URL `https://arxiv.org/abs/2312.00752`.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Jerry Huang, Siddarth Madala, Risham Sidhu, Cheng Niu, Hao Peng, Julia Hockenmaier, and Tong Zhang. Rag-rl: Advancing retrieval-augmented generation via rl and curriculum learning. *arXiv preprint arXiv:2503.12759*, 2025.

Bernal Jimenez Gutierrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in Neural Information Processing Systems*, 37:59532–59569, 2024.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Yury Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. Babilong: Testing the limits of llms with long context reasoning-in-a-haystack. *Advances in Neural Information Processing Systems*, 37:106519–106554, 2024.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 12758–12786, 2024.

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

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*, 2025.

Chuangtao Ma, Yongrui Chen, Tianxing Wu, Arijit Khan, and Haofen Wang. Large language models meet knowledge graphs for question answering: Synthesis and opportunities, 2025. URL https://arxiv.org/abs/2505.20099.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Erich Novak and Henryk Woźniakowski. *Tractability of Multivariate Problems: Volume I: Linear Information*, volume 6 of *EMS Tracts in Mathematics*. European Mathematical Society, Zürich, 2008.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Ivan Rodkin, Yuri Kuratov, Aydar Bulatov, and Mikhail Burtsev. Associative recurrent memory transformer. *arXiv preprint arXiv:2407.04841*, 2024.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. Longrope2: Near-lossless llm context window scaling. *arXiv preprint arXiv:2502.20082*, 2025.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8364–8377, 2024.

Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. Agentic retrieval-augmented generation: A survey on agentic rag, 2025. URL https://arxiv.org/abs/2501.09136.

Huatong Song, Jinhao Jiang, Yingqian Min, Jie Chen, Zhipeng Chen, Wayne Xin Zhao, Lei Fang, and Ji-Rong Wen. R1-searcher: Incentivizing the search capability in llms via reinforcement learning. *arXiv preprint arXiv:2503.05592*, 2025.

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

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Chenghan Yang, Ruiyu Zhao, Yang Liu, and Ling Jiang. Survey of specialized large language model. *arXiv preprint arXiv:2508.19667*, 2025.

Diji Yang, Jinmeng Rao, Kezhen Chen, Xiaoyuan Guo, Yawen Zhang, Jie Yang, and Yi Zhang. Im-rag: Multi-round retrieval-augmented generation through learning inner monologues. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 730–740, 2024.

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

Tan Yu, Anbang Xu, and Rama Akkiraju. In defense of rag in the era of long-context language models. *arXiv preprint arXiv:2409.01666*, 2024.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Liu Yong, and Shen Huang. End-to-end beam retrieval for multi-hop question answering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1718–1731, 2024.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A INNER PRODUCT APPROXIMATION FOR Q-FUNCTION

The Universal Approximation Theorem (UAT) states that neural networks with a single hidden layer can approximate any continuous function arbitrarily well under mild conditions. In this section, we prove a variant of the UAT for functions decomposed as an inner product involving Rotary Position Embedding (RoPE). Specifically, we show that any continuous q-function $Q(s, a^i)$ defined on a compact domain can be approximated by functions of the form:

$$F(s, a^i) = \langle E_s(s), E_a(a^i, i) \rangle, \quad E_a(a^i, i) = R_{\text{pos}(i)} E_a(a^i), \tag{8}$$

where E_s and E_a are continuous vector functions (e.g., neural networks) and R_t is the RoPE matrix of dimension r (even) parameterized by $t = \text{pos(i)}$:

$$R_t = \bigoplus_{j=1}^{r/2} \begin{bmatrix} \cos(\theta_j t) & -\sin(\theta_j t) \\ \sin(\theta_j t) & \cos(\theta_j t) \end{bmatrix}, \tag{9}$$

where θ_j are fixed frequencies. For notational simplicity in the following derivations, we introduce the following conventions:

$$(x, y) := (s, a), \quad t := \text{pos}(i), \quad h(x) := E_s(s), \quad g(y) := E_a(a^i).$$

For simplicity, we assume the domains of x, y and t are continuous, corresponding to the embeddings of text tokens.

Theorem 1. *Let $X \subset \mathbb{R}^{d_x}$, $Y \subset \mathbb{R}^{d_y}$, and $T \subset \mathbb{R}$ be compact sets, and define the compact domain $K = X \times Y \times T$. Let $C(K, \mathbb{R})$ be the space of continuous real-valued functions on K equipped with the uniform norm. Let R_t be the RoPE matrix of dimension r, defined as a block-diagonal rotation matrix (9). Define the function class:*

$$\mathcal{A} = \{F(x, y, t) = \langle h(x), R_t g(y) \rangle \mid h \in C(X, \mathbb{R}^r),\ g \in C(Y, \mathbb{R}^r)\}. \tag{10}$$

Then $\mathcal{A}$ is dense in $C(K, \mathbb{R})$. That is, for any $f \in C(K, \mathbb{R})$ and $\epsilon > 0$, there exist continuous functions $h : X \to \mathbb{R}^d$ and $g : Y \to \mathbb{R}^d$ such that:

$$\sup_{(x,y,t) \in K} |f(x, y, t) - \langle h(x), R_t g(y) \rangle| < \epsilon. \tag{11}$$

Proof. We prove the result via the Stone-Weierstrass theorem, which states that if a subalgebra $\mathcal{A} \subset C(K, \mathbb{R})$ contains the constant functions and separates points, then $\mathcal{A}$ is dense in $C(K, \mathbb{R})$. Thus, we show that $\mathcal{A}$ satisfies these requirements.

$\mathcal{A}$ is a subalgebra. We prove closure under addition, scalar multiplication, and multiplication of two arbitrary elements.

Scalar multiplication: Let $F(x, y, t) = \langle h(x), R_t g(y) \rangle \in \mathcal{A}$ and $c \in \mathbb{R}$. Define $h'(x) = ch(x)$. Then $cF(x, y, t) = \langle h'(x), R_t g(y) \rangle \in \mathcal{A}$.

Addition: Let $F_1(x, y, t) = \langle h_1(x), R_t g_1(y) \rangle$ and $F_2(x, y, t) = \langle h_2(x), R_t g_2(y) \rangle$. Define $h(x) = [h_1(x); h_2(x)] \in \mathbb{R}^{2d}$ and $g(y) = [g_1(y); g_2(y)] \in \mathbb{R}^{2d}$, and let $\widetilde{R}_t$ be a block-diagonal extension of R_t. Then

$$\langle h(x), \widetilde{R}_t g(y) \rangle = \langle h_1(x), R_t g_1(y) \rangle + \langle h_2(x), R_t g_2(y) \rangle = F_1(x, y, t) + F_2(x, y, t) \in \mathcal{A}. \tag{12}$$

Multiplication: Let F_1 and F_2 as above. Note that:

$$F_1(x, y, t) F_2(x, y, t) = \langle h_1(x) \otimes h_2(x), (R_t g_1(y)) \otimes (R_t g_2(y)) \rangle. \tag{13}$$

Since $(R_t g_1(y)) \otimes (R_t g_2(y)) = (R_t \otimes R_t)(g_1(y) \otimes g_2(y))$, and $R_t \otimes R_t$ is a block-diagonal rotation matrix with angles $\theta_j + \theta_k$ (a RoPE matrix of dimension d^2), define $h(x) = h_1(x) \otimes h_2(x) \in \mathbb{R}^{d^2}$, $g(y) = g_1(y) \otimes g_2(y) \in \mathbb{R}^{d^2}$, and let $\widetilde{R}_t$ be the RoPE matrix with frequencies $\{\theta_j + \theta_k\}$. Then:

$$F_1(x, y, t) F_2(x, y, t) = \langle h(x), \widetilde{R}_t g(y) \rangle \in \mathcal{A}. \tag{14}$$

Thus, $\mathcal{A}$ is a subalgebra.

$\mathcal{A}$ contains the constant functions. Show the constant function 1 is in $\mathcal{A}$. Augment the dimension: let $d' = d+1$, and define $h(x) = (1, 0, \ldots, 0)^T \in \mathbb{R}^{d'}$, $g(y) = (1, 0, \ldots, 0)^T \in \mathbb{R}^{d'}$. Define a modified RoPE matrix R'_t that acts as the identity on the first coordinate and as R_t on the remaining d coordinates. Then

$$\langle h(x), R'_t g(y) \rangle = 1. \tag{15}$$

$\mathcal{A}$ separates points. Let $(x_1, y_1, t_1) \neq (x_2, y_2, t_2) \in K$. Construct $F \in \mathcal{A}$ such that $F(x_1, y_1, t_1) \neq F(x_2, y_2, t_2)$.

Case 1: $x_1 \neq x_2$ or $y_1 \neq y_2$. Choose $g(y) = v$ (a constant non-zero vector) and let h be continuous with $h(x_1) \neq h(x_2)$. Then $F(x, y, t) = \langle h(x), R_t v \rangle$. Since $R_t v$ traces a circle (for v with at least two non-zero components), for generic v, $R_{t_1} v$ and $R_{t_2} v$ are not orthogonal to $h(x_1) - h(x_2)$, so $F(x_1, y_1, t_1) \neq F(x_2, y_2, t_2)$. The case when $y_1 \neq y_2$ is identical to the 1st case.

Case 2: $t_1 \neq t_2$. Choose $h(x) = w$ and $g(y) = v$. Then $F(x, y, t) = \langle w, R_t v \rangle$. Since $t \mapsto R_t v$ is injective (for $v \neq 0$ and non-zero frequencies), $R_{t_1} v \neq R_{t_2} v$. Choose w not orthogonal to $R_{t_1} v - R_{t_2} v$, so $F(x_1, y_1, t_1) \neq F(x_2, y_2, t_2)$.

Thus, by the Stone-Weierstrass theorem, $\mathcal{A}$ is dense in $C(K, \mathbb{R})$.

□

Theorem 1 establishes that our architecture is capable of approximating any continuous function arbitrarily well. However, it does not specify how complex the network needs to be to achieve a given accuracy. The following quantitative result addresses this by providing an explicit convergence rate dependent on the smoothness of the target function.

Lemma 1 (Low-rank approximation of Sobolev kernels)**.** *Let $\Omega_x \subset \mathbb{R}^{d_x}$ and $\Omega_y \subset \mathbb{R}^{d_y}$ be bounded Lipschitz domains, and let $d = d_x + d_y$.*

Let $s > d/2$ and consider a real-valued kernel

$$a \in H^s(\Omega_x \times \Omega_y). \tag{16}$$

Then, for every integer $r \geq 1$, there exist continuous functions

$$h : \Omega_x \to \mathbb{R}^r, \qquad g : \Omega_y \to \mathbb{R}^r \tag{17}$$

such that

$$\sup_{x \in \Omega_x,\ y \in \Omega_y} \left| a(x, y) - \langle h(x), g(y) \rangle_{\mathbb{R}^r} \right| \leq C\, r^{-s/d}\, \|a\|_{H^s(\Omega_x \times \Omega_y)}. \tag{18}$$

Here $C > 0$ depends only on s, d_x, d_y, and the diameters of Ω_x, Ω_y.

Proof. Since $s > d/2$ and $\Omega_x \times \Omega_y$ is a bounded Lipschitz domain in $\mathbb{R}^d$, the Sobolev embedding theorem implies

$$H^s(\Omega_x \times \Omega_y) \hookrightarrow C(\Omega_x \times \Omega_y) \tag{19}$$

continuously. In particular there exists $C_{\text{emb}} > 0$ such that

$$\|u\|_{L^\infty(\Omega_x \times \Omega_y)} \leq C_{\text{emb}} \|u\|_{H^s(\Omega_x \times \Omega_y)} \quad \text{for all } u \in H^s(\Omega_x \times \Omega_y). \tag{20}$$

Consider the unit ball

$$\mathcal{K} := \left\{ a \in H^s(\Omega_x \times \Omega_y) :\ \|a\|_{H^s(\Omega_x \times \Omega_y)} \leq 1 \right\}. \tag{21}$$

Let $\mathcal{R}_r$ denote the set of all functions on $\Omega_x \times \Omega_y$ of the form $\sum_{j=1}^r u_j(x) v_j(y)$ with $u_j \in C(\Omega_x), v_j \in C(\Omega_y)$.

Classical results on the Kolmogorov r-widths of Sobolev classes (see, e.g., Novak & Woźniakowski (2008)) give

$$d_r\big(\mathcal{K}; L^\infty(\Omega_x \times \Omega_y)\big) := \inf_{\dim V \leq r} \sup_{a \in \mathcal{K}} \inf_{b \in V} \|a - b\|_{L^\infty} \ \leq\ C_0\, r^{-s/d}, \tag{22}$$

where C_0 depends only on s, d and the diameters of the domains. Moreover, the infimum can be taken over subspaces $V \subset \mathcal{R}_r$ consisting of separable sums; hence the same rate is attainable by rank-r approximations.

Thus for any $a \in H^s(\Omega_x \times \Omega_y)$ with $\|a\|_{H^s} = M$, there exist continuous u_j, v_j such that

$$\sup_{x,y}\Big|a(x,y) - \sum_{j=1}^{r} u_j(x)v_j(y)\Big| \le C_0 M\, r^{-s/d}. \tag{23}$$

Setting $h(x) = (u_1(x), \dots, u_r(x))$ and $g(y) = (v_1(y), \dots, v_r(y))$ yields the claim. □

Theorem 2 (Approximation by RoPE-type feature maps)**.** *Let $X \subset \mathbb{R}^{d_x}$ and $Y \subset \mathbb{R}^{d_y}$ be bounded Lipschitz domains, and let $T = [0, 2\pi]$ with endpoints identified.*

Let $s > (d_x + d_y)/2$ be an integer. Assume that

$$f \in C\big(T;\, H^s(X \times Y)\big), \quad \partial_t^\ell f \in C\big(T;\, H^s(X \times Y)\big), \quad 1 \le \ell \le s. \tag{24}$$

Define

$$M := \max_{0 \le \ell \le s} \big\|\partial_t^\ell f\big\|_{C(T;H^s(X\times Y))}. \tag{25}$$

Then there exist constants $C > 0$ and $\beta > 0$, depending on s, d_x, d_y, the diameters of X, Y, and on M, such that for every integer $r \ge 1$ one can find

- *feature maps $h : X \to \mathbb{C}^r$, $g : Y \to \mathbb{C}^r$, and*
- *a family of unitary matrices $\{R_t\}_{t\in T} \subset \mathbb{C}^{r\times r}$ of the form*

$$R_t = \operatorname{diag}\big(e^{i\omega_1 t}, \dots, e^{i\omega_r t}\big), \quad \omega_j \in \mathbb{Z}, \tag{26}$$

satisfying

$$\sup_{(x,y,t)\in X\times Y\times T} \big|f(x,y,t) - \langle h(x), R_t g(y)\rangle_{\mathbb{C}^r}\big| \le C\, r^{-\beta}, \tag{27}$$

where one may take $\beta = s/(d_x + d_y + 1)$.

Proof. Since $t \mapsto f(\cdot,\cdot,t)$ is s-times continuously differentiable as an H^s-valued map, it has a Bochner–Fourier expansion

$$f(x,y,t) = \sum_{k\in\mathbb{Z}} a_k(x,y)\, e^{ikt}, \qquad \|a_k\|_{H^s} \le M\,(1+|k|)^{-s}. \tag{28}$$

A standard Jackson estimate gives the truncation bound

$$\sup_{(x,y,t)}\Big|\, f - \sum_{|k|\le N} a_k e^{ikt}\Big| \le C_1 M N^{-s}. \tag{29}$$

Let $\gamma := s/(d_x + d_y)$. Apply Lemma 1 separately to real and imaginary parts of each a_k (doubling the rank) to obtain continuous maps $h_k : X \to \mathbb{C}^{r_k}$, $g_k : Y \to \mathbb{C}^{r_k}$ with

$$\sup_{x,y}\big|a_k - \langle h_k, g_k\rangle\big| \le C_2 M\,(1+|k|)^{-s} r_k^{-\gamma}, \tag{30}$$

where C_2 depends only on s, d_x, d_y and the diameters. Define the total dimension $r := \sum_{|k|\le N} r_k$, and set $h := (h_k)_{|k|\le N}$, $g := (g_k)_{|k|\le N}$. Let R_t act block-diagonally as $R_t((z_k)) := (e^{ikt} z_k)$, so that

$$\langle h(x), R_t g(y)\rangle = \sum_{|k|\le N} e^{ikt}\langle h_k(x), g_k(y)\rangle. \tag{31}$$

The overall error then satisfies

$$\sup_{x,y,t}\big|f - \langle h, R_t g\rangle\big| \le C_1 M N^{-s} + C_2 M \sum_{|k|\le N} (1+|k|)^{-s} r_k^{-\gamma}. \tag{32}$$

To minimize the second term under the constraint $\sum r_k = r$, choose

$$r_k \sim (1+|k|)^{-s/(\gamma+1)}. \tag{33}$$

Then, since $s/(\gamma+1) = d_x + d_y \geq 2$,

$$\sum_{|k|\leq N} (1+|k|)^{-s} r_k^{-\gamma} \leq C_3 N^{-\gamma} r^{-\gamma}. \tag{34}$$

Thus

$$\sup_{x,y,t} \big| f - \langle h, R_t g \rangle \big| \leq C_1 M N^{-s} + C_4 M N^{-\gamma} r^{-\gamma}. \tag{35}$$

Choosing $N = \lfloor r^{\gamma/(s+\gamma)} \rfloor$ balances the two terms and yields

$$\sup_{x,y,t} \big| f - \langle h, R_t g \rangle \big| \leq C M\, r^{-s\gamma/(s+\gamma)}, \tag{36}$$

with $s\gamma/(s+\gamma) = s/(d_x + d_y + 1)$. □

B EARLY STOPPING EXPERIMENTS

In this section, we study a simple early stopping rule for the retrieval agent. Let

$$\mathbf{a} = (a_1, a_2, \ldots, a_T)$$

be the full sequence of chunks the agent would select if no stopping threshold were applied, and let G be a set of ground-truth chunks for the current question.

For each step t, the agent outputs a Q-value Q_t for taking the next retrieval action. Given a fixed Q-value threshold $Q_{\text{threshold}}$, we simulate an early-stopping policy that keeps taking actions while $Q_t \geq Q_{\text{threshold}}$ and terminates as soon as $Q_t < Q_{\text{threshold}}$. We denote by t_{stop} the number of actions actually taken under this policy, i.e. the number of selected chunks:

$$t_{\text{stop}} = \text{number of steps until the first } t \text{ with } Q_t < Q_{\text{threshold}}.$$

Independently of the stopping rule, we define t_{earliest} as the earliest step at which all ground-truth chunks have already been collected:

$$t_{\text{earliest}} = \min\big\{t \,:\, \{a_1, \ldots, a_t\} \supseteq G\big\}.$$

If the agent never collects all ground-truth chunks, i.e. such a t does not exist, we discard this episode from the analysis below.

For comparison, we also consider an oracle stopping policy that is allowed to look at the ground truth: it knows t_{earliest} for each episode and simply stops at this step. By construction, this oracle policy never stops too early or too late.

Depending on the relation between t_{stop} and t_{earliest} we distinguish three outcomes.

Early stop ("early"). If $t_{\text{stop}} < t_{\text{earliest}}$, the stopping rule terminates before all ground-truth chunks have been selected. In this case the error is due to stopping too early and missing potentially useful chunks.

Perfect stop ("perfect"). If $t_{\text{stop}} = t_{\text{earliest}}$, the stopping rule terminates exactly at the first step when the set of selected chunks already contains all ground-truth chunks. In this case, the stopping behavior is optimal with respect to our definition.

Late stop ("late"). If $t_{\text{stop}} > t_{\text{earliest}}$, then at some earlier step the agent had already collected all ground-truth chunks but continued to retrieve additional chunks. This corresponds to stopping too late and taking unnecessary steps.

Figure 4 (top row, panel (a)) shows how the proportions of early and late errors change as a function of the Q-value threshold $Q_{\text{threshold}}$ on HotPotQA. For small thresholds, the agent almost never stops too early but may continue to retrieve redundant chunks, which leads to late errors. As the threshold increases, late errors decrease, but the probability of stopping too early grows.

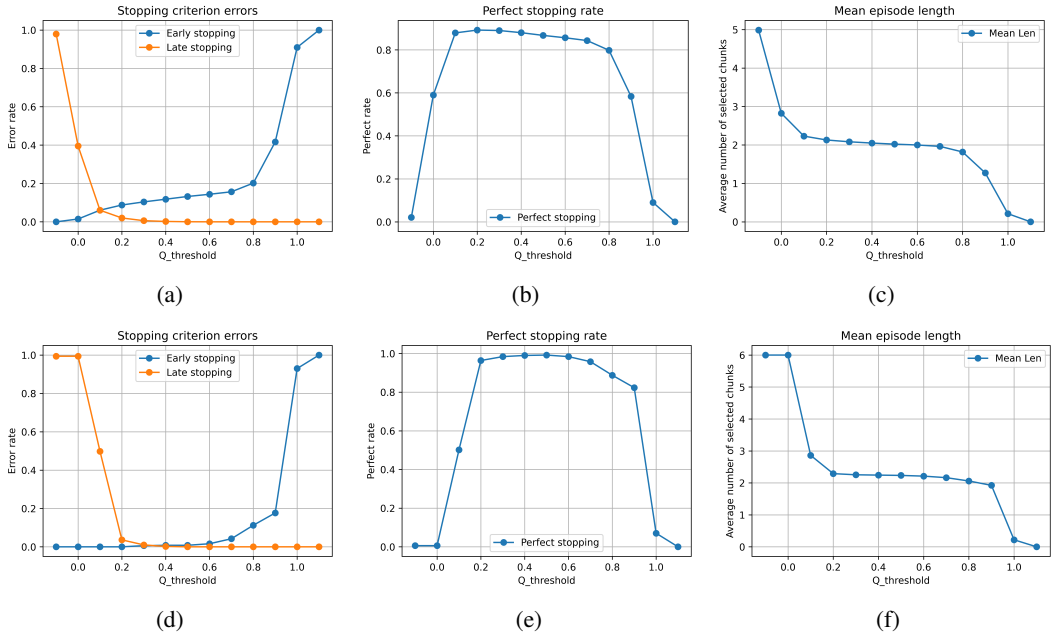


Figure 4: Early stopping analysis on HotPotQA (top row) and BabiLong QA2 (bottom row). Panels (a,d) show the proportions of early and late errors as a function of the Q-value threshold $Q_{\text{threshold}}$. Panels (b,e) show the proportion of perfect stops. Panels (c,f) show the average number of selected chunks (episode length).

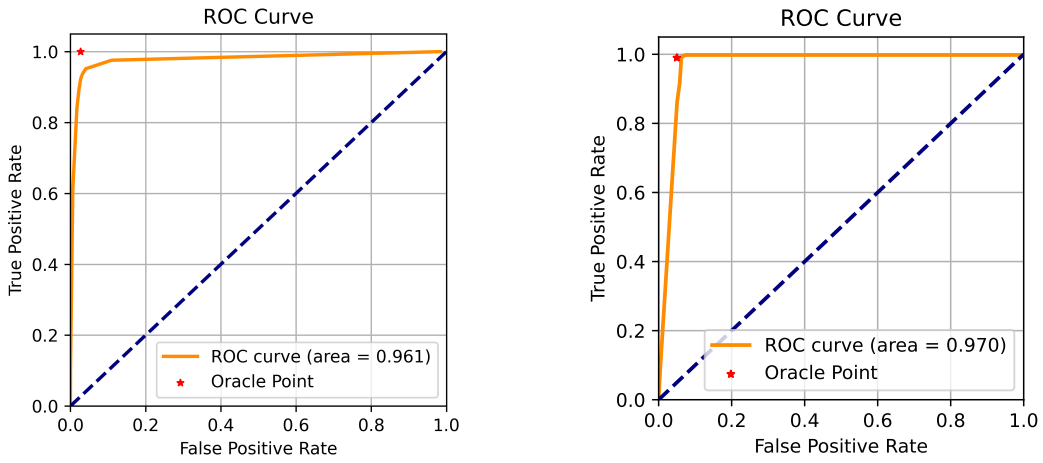
Table 4: HotPotQA early stopping experiments

Q-value threshold	stopped early	stopped later	perfect stop	TPR	FPR	Episode len	Fact EM	Fact F1	Ans EM	Ans F1
-0.1	0	0.979	0.021	0.983	0.380	4.99	0.968	0.563	0.588	0.759
0.0	0.015	0.395	0.590	0.976	0.110	2.82	0.954	0.843	0.592	0.761
0.1	0.061	0.060	0.879	0.952	0.041	2.23	0.910	0.915	0.593	0.756
0.2	0.088	0.020	0.892	0.937	0.032	2.13	0.883	0.917	0.587	0.752
0.3	0.104	0.006	0.890	0.927	0.029	2.08	0.868	0.915	0.585	0.747
0.4	0.118	0.002	0.880	0.919	0.027	2.05	0.854	0.911	0.575	0.737
0.5	0.132	0	0.867	0.910	0.025	2.02	0.840	0.907	0.571	0.734
0.6	0.144	0	0.856	0.903	0.024	2.00	0.829	0.902	0.570	0.730
0.7	0.157	0	0.843	0.891	0.023	1.96	0.817	0.895	0.564	0.724
0.8	0.202	0	0.798	0.840	0.017	1.82	0.773	0.847	0.546	0.702
0.9	0.417	0	0.583	0.611	0.006	1.27	0.565	0.620	0.444	0.588
1.0	0.910	0	0.090	0.105	0.000	0.21	0.088	0.111	0.266	0.385
1.1	1.000	0	0	0	0	0	0	0	–	–

Panel (b) of Figure 4 reports the proportion of "perfect" stopping events, peaking around thresholds $Q_{\text{threshold}} \approx 0.1$–$0.3$. Panel (c) shows the average number of selected chunks (episode length) under the same policy. Larger thresholds lead to shorter episodes, but once the threshold becomes too high, the early-stop error rate rapidly increases and performance degrades.

Table4 summarises these trade-offs quantitatively on HotPotQA for the GTE embedder with `penalize_extra_steps=True` and `never_terminate=True`. We report the fraction of early, late and perfect stops, the average episode length, and the final Fact EM and Fact F1 scores, as well as the corresponding true positive rate (TPR) and false positive rate (FPR) for the stopping rule viewed as a binary classifier. The best Fact F1 is achieved at $Q_{\text{threshold}} = 0.2$, confirming that moderate thresholds provide a good balance between taking enough retrieval steps and avoiding unnecessary ones.

Using the TPR and FPR columns of Tables 4 and 5, we can plot the receiver operating characteristic (ROC) curves of the early-stopping rule, shown in Figure 5. Panel (a) corresponds to HotPotQA and panel (b) to BabiLong QA2. Each point on the curves corresponds to a particular Q-value threshold $Q_{\text{threshold}}$. The red star in each panel marks the oracle stopping policy introduced above, which knows t_{earliest} and stops exactly at that step; this point serves as an upper bound on the achievable trade-off



(a) HotPotQA

(b) BabiLong QA2

Figure 5: ROC curves for the early-stopping rule. Panel (a) shows HotPotQA; panel (b) shows BabiLong QA2. The dashed line indicates random performance. Each point corresponds to a different Q-value threshold $Q_{\text{threshold}}$. The red star denotes the oracle stopping policy that always stops at t_{earliest}, i.e. exactly when the last ground-truth chunk has been retrieved.

Table 5: BabiLong QA2 early stopping experiments.

Q-value threshold	stopped early	stopped later	perfect stop	Episode len	Fact EM	Fact F1	Ans EM	Ans F1
-0.10	0.000	0.994	0.006	6.00	0.996	0.499	0.884	0.884
0.00	0.000	0.994	0.006	6.00	0.996	0.499	0.884	0.884
0.10	0.000	0.498	0.502	2.86	0.996	0.845	0.944	0.944
0.20	0.000	0.036	0.964	2.29	0.996	0.949	0.976	0.976
0.30	0.006	0.010	0.984	2.25	0.990	0.952	0.970	0.970
0.40	0.008	0.002	0.990	2.24	0.988	0.953	0.970	0.970
0.50	0.008	0.000	0.992	2.23	0.988	0.954	0.972	0.972
0.60	0.016	0.000	0.984	2.21	0.980	0.948	0.968	0.968
0.70	0.042	0.000	0.958	2.16	0.954	0.934	0.948	0.948
0.80	0.112	0.000	0.888	2.06	0.884	0.905	0.884	0.884
0.90	0.177	0.000	0.823	1.92	0.820	0.861	0.830	0.830
1.00	0.930	0.000	0.070	0.22	0.070	0.107	0.230	0.230
1.10	1.000	0.000	0.000	0.00	0.000	0.000	0.000	0.000

between TPR and FPR. On HotPotQA the area under the curve (AUC) is 0.96, and BabiLong QA2 - 0.97.

Figure 4 (bottom row) and Table 5 report the same analysis on BabiLong QA2. Qualitatively, the behaviour of the stopping rule is similar to HotPotQA: higher thresholds lead to shorter episodes and more early stops, while lower thresholds reduce early-stop errors at the cost of more late stops and longer episodes.

However, the transition between these regimes is much sharper on BabiLong QA2. For thresholds in the range $Q_{\text{threshold}} \in [0.2, 0.6]$ the fraction of perfect stops remains very high (≈ 0.95–0.99), while the average episode length is reduced from about 6 to roughly 2.2 retrieval steps. In this region Fact EM and Fact F1 stay close to their maximum values (Fact F1 around 0.95), and answer accuracy (Ans EM/F1) is also near-optimal. Only when the threshold approaches 1.0, performance collapses, as the agent stops almost immediately and misses relevant chunks.

C Sensitivity to retrieval budget

We investigate the dependence of final model performance on the number of Q-RAG retrievals (i.e., the retrieval budget). For this analysis, we used a Q-RAG system with an Alibaba-NLP/gte-multilingual-base embedder, trained on a combination of the HotpotQA and Musique datasets. This

Table 6: Sensitivity to the number of retrieves. Dataset: HotpotQA (1000 samples). Embedder Alibaba-NLP/gte-multilingual-base was trained on Hotpotqa+Musique.

Retrievals	Facts		Qwen3-4B		Qwen3-14B		Qwen3-32B	
	EM	F1	EM	F1	EM	F1	EM	F1
2	0.832	0.903	0.439	0.620	0.556	0.708	0.504	0.675
3	0.935	0.771	0.481	0.657	0.570	0.730	0.510	0.692
4	0.962	0.652	0.493	0.664	0.577	0.734	0.513	0.695
5	0.978	0.565	0.481	0.656	0.584	0.744	0.512	0.692

embedder supports contexts of up to 8192 tokens, enabling the use of a larger retrieval budget. We evaluated the system on 1000 samples from the HotpotQA dataset. The final generation of the answers was performed by three LLMs: Qwen3-4B, Qwen3-14B, and Qwen3-32B.

The results are presented in Table6. Here, EM (Exact Match) indicates the number of correct (ground-truth supporting) chunks retrieved, while F1 accounts for the inclusion of noise (non-supporting) chunks. The table shows that increasing the number of retrieves from 2 to 3 improves both the number of correct facts retrieved and the answer quality across all three LLMs. These experiments suggest that, within a reasonable range of retrieval counts, final answer accuracy is primarily dependent on the retrieval of correct chunks and is not degraded by the presence of noise chunks.

D Planning for Multi-Step Retrieval

We *can* apply **planning** at the multi-step retrieval stage, formulating source selection as a search over the space of action trajectories; see § 4.4 for an application. In the spirit of *Beam-Retriever*, we can run beam search where candidates are ranked by the learned action-value $Q_\theta(s, a)$. However, our planning is computationally cheaper because Q_θ is computed as a *dot product* of state and action embeddings, $Q_\theta(s, a) = \langle E_s(s),\, E_a(a)\rangle$, so no new transformer forward passes are required for each candidate chunk, whereas *Beam-Retriever* relies on a transformer reranker over trajectories, incurring fresh forward passes at every expansion. Details of the embedding-based scoring are provided in § 3.2. At inference, we perform *beam search over* Q and *deterministically* expand the top-k actions by Q_θ.

E Method Complexity and Efficiency

Q-RAG produces a final answer using two main components. The first is a *multi-step retrieval agent* that performs iterative search over the full document to collect all context-relevant evidence (see sec. 3.2). The second is an *LLM Answerer* that conditions on the retrieved chunks and generates the final response. Importantly, only the retrieval agent interacts with the original long context; the effective context length seen by the LLM Answerer depends solely on the retrieval hyperparameters (e.g., number of retrieval steps T, maximum chunk length). Consequently, the time and memory complexity of the LLM Answerer with respect to the original context length N are both $\mathcal{O}(1)$. Retrieval agent consists of two embedders: state embedder E_s and action embedder E_a (see sec. 3.2).

Chunks embedding. The action embedder computes embeddings for chunks of the original document. If the document has length N and the chunk size is n_c, embedding the entire document takes $\mathcal{O}\Big(\frac{N}{n_c}\, t_{\text{act}}\Big)$, where t_{act} is the embedding time per chunk (treated as a constant). The action embedder performs a single pass over all chunks per retrieval episode; thus its complexity is linear in N, i.e., $\mathcal{O}(N)$.

State Embedding. The state embedder processes the state K times per episode (once per search step). From the construction of the state (see fig. 1), the total cost over an episode is $\mathcal{O}(K\, t_{\text{state}})$, where state embedding time t_{state} depends on n_c and K, but not on N. Hence, the state embedder is $\mathcal{O}(1)$ with respect to document length N.

Search Policy. To select the next chunk at each step, we compute the inner product between the current state embedding and all action embeddings. With naive implementation, selecting all K actions over the episode requires $\mathcal{O}\Big(K\, d_{\text{emb}}\, \frac{N}{n_c}\Big) = \mathcal{O}(N)$, where d_{emb} is size of embedding vectors. This can be reduced using approximate kNN methods that achieve sub-linear query time in practice (**??**).

Overall time complexity. Summing the terms above yields

$$\mathcal{O}\left(\frac{N}{n_c}\, t_{\text{act}} \;+\; K\, t_{\text{state}} \;+\; K\, d_{\text{emb}}\, \frac{N}{n_c}\right) \;=\; \mathcal{O}(N),$$

since K, t_{act}, t_{state}, and d_{emb} do not depend on N.

Space complexity. The main part that direcly depends on document length is the nubmer of chunk embeddings we need to store: $\mathcal{O}\Big(d_{\text{emb}}\, \frac{N}{n_c}\Big) = O(N)$. In practice, embeddings are lightweight; GPU memory is mainly consumed by the LLM weights and the action embedder forward passes. By capping the action embedder's batch size (parameter `chunk_batch`), the growth of peak memory with N becomes negligible.

Training Time Efficiency. A critical practical advantage of the Q-RAG framework is its efficient and rapid training convergence, as demonstrated in Figure 6. The learning curves depict the model's performance evolution on two distinct and challenging benchmarks: BabiLong QA3 and HotPotQA. The curves show a sharp initial rise in evaluation metric scores, followed by a stable plateau, indicating that the model quickly learns the core retrieval-augmented generation task. Notably, this convergence is achieved within approximately 12 hours of training time on a GPU setup.

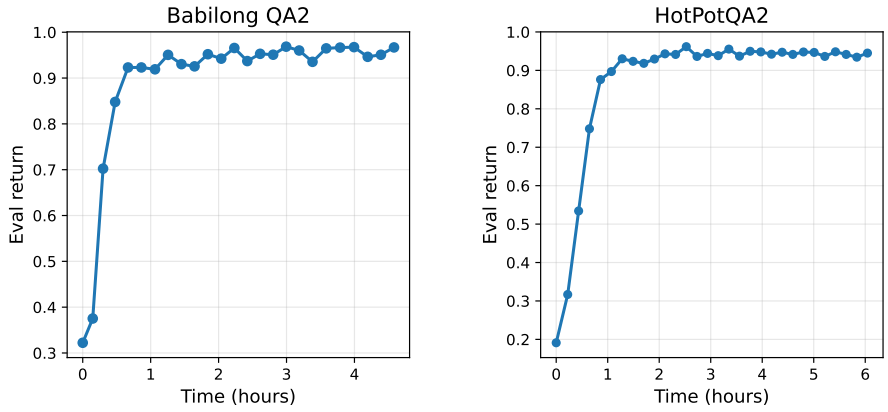


Figure 6: Learning curves for HotPotQA and BabiLong QA3 runs. Both graphs the average episodic return with respect to training time.

F Extra QA results

Table 7 compares multi-step retrieval methods on HotPotQA-distractors, Musique (in-distribution), and Musique (out-of-distribution). It reports both fact-retrieval (Fact F1, Fact EM) and answer-generation (Ans F1, Ans EM) scores. Q-RAG and its planned variant (Plan Q-RAG) achieve strong overall results, especially on out-of-distribution data, while Beam-Retriever leads on HotPotQA but generalizes less robustly. Methods with missing entries did not report results for the corresponding dataset or metric.

G Training details

We trained the model with AdamW (learning rate 1.5×10^{-5}, β_1=0.9, β_2=0.98, ϵ=10^{-6}, weight decay 5×10^{-4}). The learning rate followed a *linear* schedule: we used a warm-up of 1,000 steps, then linearly decayed the rate to 10% of its initial value over the remaining training steps. We applied

Table 7: Comparison of methods on HotPotQA-distractors, Musique (in-distribution), and Musique (OOD). Bold text and underline denote the best and second best scores respectively.

	HotPotQA				Musique				Musique (OOD)				Average	
Methods	Fact F1	Fact EM	Ans F1	Ans EM	Fact F1	Fact EM	Ans F1	Ans EM	Fact F1	Fact EM	Ans F1	Ans EM	Ans F1	Ans EM
Plan Q-RAG + QwQ-32B	0.95	0.91	0.76	0.60	0.84	**0.76**	**0.60**	**0.44**	0.69	0.53	0.51	0.36	**0.62**	**0.46**
Q-RAG+QwQ-32B	0.93	0.89	0.76	0.59	0.81	0.72	0.59	0.43	**0.71**	**0.55**	**0.52**	0.37	**0.62**	**0.46**
Beam-Retriever+QwQ-32B	**0.97**	**0.94**	**0.77**	**0.61**	**0.86**	0.69	0.59	0.43	0.61	0.36	0.40	0.27	0.59	0.44
Search-r1	0.81	0.66	0.65	0.52	–	–	–	–	**0.71**	**0.55**	0.51	**0.39**	–	–
Search-o1	–	–	–	–	–	–	–	–	–	–	–	–	–	–
GraphReader	–	–	–	–	–	–	–	–	–	–	–	–	–	–
HippoRAG	–	–	–	–	–	–	–	–	–	–	–	–	–	–

gradient clipping with a maximum ℓ_2 norm of 2.0 and used gradient accumulation for 8 steps. The base mini-batch size was 12; with accumulation this yields an effective batch size of $12 \times 8 = 96$ per update (scaled by the number of devices if using distributed training).

In the objective and algorithmic components we set γ=0.99, α=0.05, λ=0.5, and τ=0.02. Action representations were capped at a maximum length of 220 tokens.

The end-to-end training of a single model did not exceed 12 hours on a single `A100-80GB` GPU.

Models per benchmark. For open-domain QA benchmarks (*HotPotQA*, *Musique*), we trained an `multilingual-e5-large` encoder. For *Ruler* and *BabiLong*, we trained `facebook/contriever`.

H Evaluation details

LLM Models for generation. To compute answer-level metrics (Ans EM and Ans F1), we condition the QwQ-32B model on the question and the retrieved text chunks. All answer-generation results reported for Q-RAG and Plan Q-RAG on the HotPotQA and Musique benchmarks were obtained under consistent generation settings: decoding with temperature 0.0 and a maximum output length of `max_tokens` = 8000. For the BabiLong and RULER experiments, we instead used Qwen-4B with `max_tokens` = 512 and reasoning disabled (`enable_thinking` = `False`).

Retrieval configuration. For Q-RAG we limit the number of retrieval steps to $T = 2$ on HotPotQA, RULER and Babilong we use $T = 4$. The same step limits are used when evaluating Search-R1 and Beam Retriever.

We split documents into fixed-length, non-overlapping chunks, aiming not to break sentences across chunk boundaries. The chunk length is primarily determined by the context window of the embedders used in our main experiments (512 tokens) and the number of retrieval steps. For Needle-in-a-Haystack and BabiLong we use a chunk length of 64 tokens. For open-domain QA tasks we set the chunk length as a function of the number of retrieval steps i.e. for HotPotQA we segment the corpus into chunks of at most 220 tokens ($T = 2$); for Musique we use action chunks of at most 110 tokens ($T = 4$). In additional experiments with a 'Alibaba-NLP/gte-multilingual-base' (8k context length) we use a chunk length of 256 tokens.

heightDataset	Setting	Chunk size	T	Backbone retriever	Answering LLM
HotPotQA	Q-RAG / Plan Q-RAG	220	2	`multilingual-e5-large`	QwQ-32B
HotPotQA	Q-RAG (early stopping)	256	5	`Alibaba-NLP/gte-multilingual-base`	QwQ-32B
Musique	Q-RAG / Plan Q-RAG	110	4	`multilingual-e5-large`	QwQ-32B
Babilong	Q-RAG	64	4	`facebook/contriever`	Qwen3-4B
RULER	Q-RAG	64	4	`facebook/contriever`	Qwen3-4B

Table 8: Retrieval and generation configuration for each dataset. Chunk size is in tokens; T is the maximum number of retrieval steps.

Fact-level metrics. Let S_{gt} be the set of ground-truth supporting facts and S_{pred} be the set of predicted supporting facts returned by the retriever. Our Fact EM metric is defined as

$$\text{Fact-EM} = \begin{cases} 1, & \text{if } S_{\text{gt}} \subseteq S_{\text{pred}}, \\ 0, & \text{otherwise.} \end{cases}$$

Equivalently, in code: `em = 1.0 if gt_sf.issubset(pred_sf) else 0.0`. Thus Fact EM gives full credit whenever the prediction covers all ground-truth facts, even if it also contains additional, irrelevant chunks; it does not require the predicted and ground-truth sets to be exactly equal.