ARK: Answer-Centric Retriever Tuning via KG-augmented Curriculum Learning

Anonymous ACL submission

Abstract

Retrieval-Augmented Generation (RAG) has emerged as a powerful framework for knowledge-intensive tasks, yet its effectiveness in long-context scenarios is often bottlenecked by the retriever's inability to distinguish sparse yet crucial evidence. Standard retrievers, optimized for query-document similarity, frequently fail to align with the downstream goal of generating a precise answer. To bridge this gap, we propose a novel fine-tuning framework that optimizes the retriever for Answer Alignment. Specifically, we first identify high-quality positive chunks by evaluating their sufficiency to generate the correct answer. We then employ a curriculum-based contrastive learning scheme to fine-tune the retriever. This curriculum leverages LLM-constructed Knowledge Graphs (KGs) to generate augmented queries, which in turn mine progressively challenging hard negatives. This process trains the retriever to distinguish the answer-sufficient positive chunks from these nuanced distractors, enhancing its generalization. Extensive experiments on 10 datasets from the Ultradomain and LongBench benchmarks demonstrate that our fine-tuned retriever achieves state-of-the-art performance, improving 14.5% over the base model without substantial architectural modifications and maintaining strong efficiency for long-context RAG. Our work presents a robust and effective methodology for building truly answer-centric retrievers.

1 Introduction

Large Language Models (LLMs) have achieved human-level performance on many NLP tasks (Achiam et al., 2023; Touvron et al., 2023), but still struggle with long-term memory, often omitting or conflating details in scenarios requiring complex reasoning or extended context (Li et al., 2024; Lazaridou et al., 2021).

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) addresses this limitation by connecting LLMs to external knowledge sources, refreshing their memory at inference time. Since its introduction, RAG has rapidly evolved from naive RAG pipeline, with vector retrieval (Shi et al., 2023; Borgeaud et al., 2022) to advanced pipelines that incorporate recursive chunking (Sarthi et al., 2024), knowledge graphs (KGs) (Edge et al., 2024), and internal memory modules (Qian et al., 2024), substantially improving the handling of long-context input.

While KG-integrated pipelines (Edge et al., 2024) have shown promising gains for complex summarization, they suffer from efficiency and accuracy bottlenecks in broader retrieval tasks. The indexing phase in KG-based RAG (Edge et al., 2024) and follow-up works (Gutiérrez et al., 2024; Guo et al., 2025) requires processing extremely large token volumes with powerful LLMs, resulting in high computational cost. In addition, KGs often struggle with fine-grained entity disambiguation: community-curated clusters, though rich, are noisy and insufficiently filtered. Consequently, retrieval may aggregate irrelevant or even conflicting content, reducing both the consistency and quality of generated outputs.

To train a retriever for true answer sufficiency, we propose ARK (**A**nswer-centric **R**etriever fine-tuning via **K**G-driven curriculum), a framework that redefines the role of Knowledge Graphs in RAG. Rather than serving as a direct retrieval source, the KG powers an **Answer-Centric Curriculum Learning** scheme, enabling fine-grained discrimination, sufficiency-aware retrieval, and improved generalization of the retriever.

At its core, ARK first identifies what makes evidence truly useful—whether it suffices to generate the correct answer. We formalize this with an in-context sufficiency metric combining three alignment strategies (Forward, Backward, Retriever) to extract high-quality **positive chunks** as anchors. Building on these positives, ARK leverages the KG

as a **hard-negative generator**: It runs Personalized PageRank(PPR) over the KG to extract answer-relevant subgraphs, which in turn guide the creation of augmented queries. These queries are specifically designed to mine progressively challenging hard negatives. Concretely, we use PPR on a query-specific subgraph to expose co-occurrence neighbors near the gold entities, and inject them into query augmentations so the retriever is drawn to false positives—highly related yet insufficient evidence. This community-aware mining yields harder, more calibration-relevant negatives than random or keyword-based baselines. Through contrastive training against this curriculum, ARK learns to prioritize truly answer-informative segments while filtering misleading context, mastering both sufficiency and fine-grained discrimination.

To summarize, our contributions are:

- We propose ARK, a framework that finetunes the retriever through contrastive learning for scalable long-context retrieval.
- We devise a synthetic query-generation pipeline that uses KG subgraphs to produce challenging hard negatives, and integrate them into an answer-centric curriculum learning scheme that progressively increases negative difficulty.
- We introduce an in-context answer sufficiency metric, composed of three complementary alignment scores (Forward, Backward, and Retriever), to identify high-quality positive chunks that serve as the anchor for contrastive learning.
- Through extensive experiments, we demonstrate that ARK achieves state-of-the-art retrieval performance on 8 out of 10 datasets across the LongBench and Ultradomain benchmarks, with an average F1-score improvement of 14.5% over the base model, showcasing its effectiveness and efficiency.

2 Related work

2.1 Traditional RAG Techniques

RAG systems enhance LLM outputs by combining retrieval with generation. Early implementations used fixed retrievers to supply documents to a reader model: classical methods such as BM25 (Robertson et al., 2009) relied on lexical matching, while neural retrievers like DPR (Karpukhin et al., 2020) employed dense embeddings for semantic retrieval. Typically, retrievers were trained on QA pairs, with readers fine-tuned independently.

Later work emphasized tighter integration. RAG models (Lewis et al., 2020) enabled end-to-end fine-tuning by treating retrieved documents as latent variables, aligning the retriever with a BART-based generator. This improved recall and consistency but increased complexity due to non-differentiable retrieval. To mitigate this, techniques such as hard negative mining (Robinson et al., 2020) and knowledge distillation from generators into retrievers (Izacard and Grave, 2021) were proposed, further enhancing retriever–reader synergy.

2.2 Advanced RAG Techniques

Recent advances in RAG aim to improve retrieval reasoning and integrate hybrid knowledge sources. One line of work focuses on query rewriting and decomposition. RQ-RAG (Chan et al., 2024) enhances multi-hop QA by decomposing complex queries into simpler sub-queries, while HyDE (Gao et al., 2022) generates hypothetical documents that serve as refined queries to boost retrieval precision. Context rewriting techniques have also emerged. MemoRAG (Qian et al., 2024) further addresses long-context retrieval by compressing memory and producing clue phrases to guide retrieval, composing answers from retrieved snippets.

Another direction integrates symbolic and neural approaches, often by incorporating KGs into retrieval pipelines. GraphRAG (Edge et al., 2024) constructs graphs from documents and summarizes clusters into "community reports" via community detection, supporting improved multi-hop reasoning. LightRAG (Guo et al., 2025) simplifies this pipeline by first retrieving low-level nodes and then following graph links to higher-level concepts, improving both recall and efficiency. HippoRAG (Gutiérrez et al., 2024) models memory consolidation using KGs and PPR, retrieving subgraphs at query time to emulate long-term memory access.

3 Methodology

3.1 Framework

Our proposed framework, ARK, follows a two-stage architecture—**Query Construction** and **Contrastive Finetuning**—as illustrated in Figure 1. The first stage prepares curriculum components, while the second performs training.

In **Query Construction**, we build an LLM-derived KG from the context, identify entities from the QA pair, and extract a query-specific subgraph. This subgraph supports the creation of *injected queries*, which preserve the semantics of the orig-

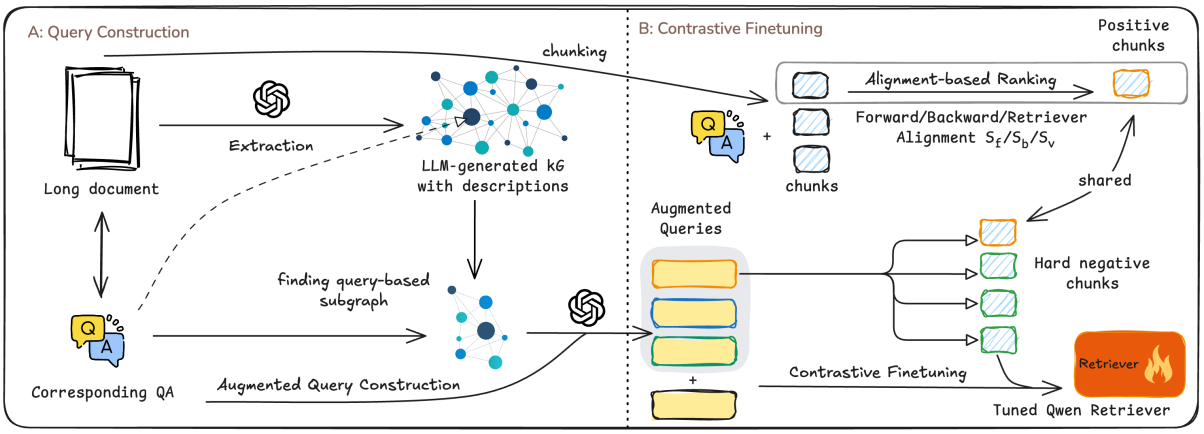

Figure 1: **Our RAG Retriever Finetuning Framework** ARK, which consists of two major stages: A (Query Construction): From long documents and their corresponding QA pairs, we extract a query-based subgraph using an LLM-generated KG. The subgraph is reformulated with knowledge injection to produce enriched queries. B (Contrastive Finetuning): Using both the original query and injected variants, we identify positive chunks (via alignment scoring) and hard negatives (that match injected queries but differ semantically from ground truth).

inal while adding contextual structure. These enriched queries serve as the basis for generating hard negatives. In **Contrastive Finetuning**, we define

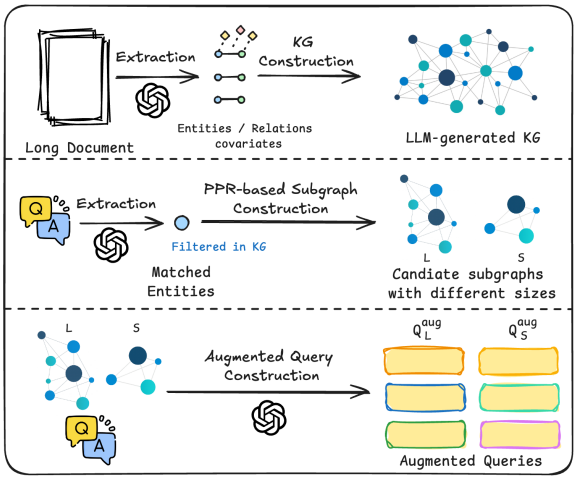

Figure 2: **Query Construction Phase.** The pipeline begins with *KG Construction*, where we extract entities, relations, and covariates from long documents to construct an LLM-generated KG. Given a corresponding QA pair, relevant entities are extracted and used to *construct PPR-based subgraphs* from the KG, with varying maximum sizes to control difficulty. Finally, *Augmented Queries* are formulated with LLM conditioned on these candidate subgraphs.

positives using our in-context answer sufficiency metric (combining three alignment scores) to rank context chunks and select the most sufficient ones. Hard negatives are chunks that score highly for an injected query but are absent from the positive set. With these positives and negatives, we finetune the retriever via contrastive learning, enhancing its discriminative ability. The resulting retriever integrates seamlessly into existing RAG pipelines without architectural changes.

3.2 KG-based Query Construction

During training, to effectively extract high-quality, answer-guided queries from ultra-long source contexts, we develop an innovative pipeline that integrates LLM-assisted KG construction, PPR-based subgraph construction, and finally query formation. As illustrated in Figure 2, the data generation process consists of three steps.

KG construction We adopt a prompt design incorporating Chain-of-Thought (CoT) reasoning to enhance the quality of entity recognition. Different from the GraphRAG (Edge et al., 2024) approach, our method focuses solely on extracting entities and their relationships, removing the need for generating community-level summaries.

Due to the inherent limitations of LLMs, specifically their focus on localized context, the constructed KG may exhibit lower edge density compared to traditional KGs, as only intra-chunk relationships are identified. To address this sparsity, we augment the graph by introducing undirected edges between entities whose embedding similarity exceeds a predefined threshold τ, thereby enriching the graph's connectivity and facilitating more effective downstream retrieval.

Cohesive Subgraph Extraction by Community Search To capture neighborhoods most relevant to the input answer query, we employ commu-

Algorithm 1: Cohesive Subgraph Extraction

Require: Matched entities V_a, LLM-generated KG $G(V, E)$, PPR parameter α, ϵ, Maximum subgraph size k
Ensure: Answer-related entities V_{comm}

$\chi_a \leftarrow$ Indicator function of V_a
Compute $\text{apr}_{\text{comm}}(\alpha, \chi_a)$ with ϵ threshold
$\{\text{apr}_i, v_i\}_{i \in V} \leftarrow \texttt{sort}(\text{apr}_{\text{comm}}, \texttt{desc} = \texttt{True})$
$\{\text{apr}_i, v_i\}_{i \in V'} \leftarrow$ Filter trailing terms of $\{\text{apr}_i, v_i\}_{i \in V}$ where $\{i \mid \text{apr}_i < \epsilon\}$
$\{\text{apr}_i, v_i\}_{i \in V'} \leftarrow \{-\log \text{apr}_i, v_i\}_{i \in V'}$
$\texttt{index} \leftarrow \Delta_{\arg\max_i, i < \min\{k, |V'|\}} \text{apr}$
$V_{\text{comm}} \leftarrow$ First $\texttt{index}$ entities from $\{\text{apr}_i, v_i\}_{i \in V'}$

nity search, which identifies query-dependent subgraphs rather than global clusters. This aligns with our retrieval objective: surfacing context that is semantically close to the query yet hard to distinguish from true positives, thus serving as high-quality hard negatives. LLM-derived knowledge graphs are typically sparse, with edges concentrated within intra-chunk links, making purely topology-driven methods less effective. To address this, we adopt Personalized PageRank (PPR) seeded on query entities to extract semantically coherent communities.

However, due to the sparsity of LLM-derived KGs, where edges are primarily confined to intra-chunk relationships, conventional topology-driven community search methods often fail to perform effectively. To overcome this limitation, we utilize PPR to assess and construct communities based on semantically aligned entities. We begin by extracting the positive query entities V_a from the answer and identifying their corresponding entities within the KG. Using these matched entities, we perform PPR based on the matched ones, formally defined as:

$$\text{pr}(\alpha, \chi_a) = \alpha \chi_a + (1 - \alpha) W \, \text{pr}(\alpha, \chi_a) \quad (1)$$

Here, $\text{pr}(\alpha, \chi_a)$ denotes the PPR value for the given entities a on KG G, χ_a denotes the indicator function of a, α denotes the teleport probability, and $W = A^\top D^{-1}$ represents the normalized transition matrix based on adjacency matrix A and degree matrix D from graph G.

Due to scalability challenges in computing PPR over large graphs, we employ power iteration to approximate the solution. The approximation vector $\text{apr}^{(0)}(\alpha, \chi_a)$ is initialized as χ_a, and iteratively updated as follows:

$$\text{apr}^{(n+1)}(\alpha, \chi_a) = \alpha \chi_a + (1 - \alpha) W \, \text{apr}^{(n)}(\alpha, \chi_a) \quad (2)$$

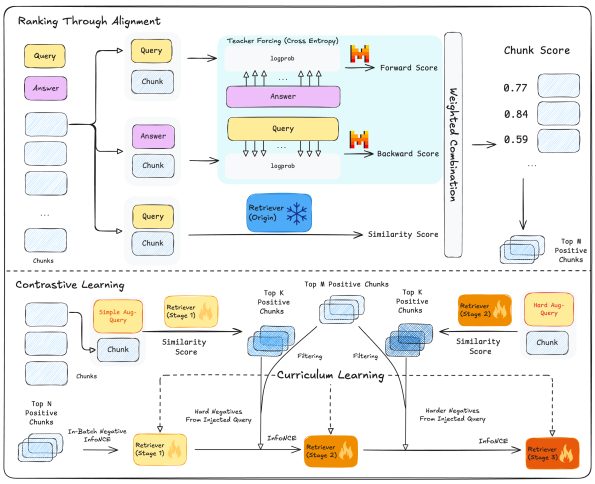

Figure 3: **Contrastive Finetuning Phase.** Our finetuning pipeline comprises two sequential components: *Ranking Alignment*, in which for each sample, we combine three alignment scores to select the Top-M chunks as positive chunks; followed by *Curriculum-based Contrastive Learning*, which progressively refines the retriever through (i) in-batch negative sampling, (ii) hard negatives $\mathcal{T}^-_{\text{hard}_L}$ mined via query set $\mathcal{T}^-_{\text{hard}_L}$, and (iii) more challenging negatives $\mathcal{T}^-_{\text{hard}_S}$ obtained from $\mathcal{Q}^{\text{aug}}_S$.

Given the absolute approximation error bound: $\left\|\text{apr}^{(n+1)}(\alpha, \chi_a) - \text{apr}^{(n)}(\alpha, \chi_a)\right\|_\infty < \epsilon$ The computational complexity of the iterative method is $O\left(|E| \log \frac{1}{\epsilon}\right)$, as initially proposed by (Haveliwala and Kamvar, 2003).

Following the approach of (Andersen and Chung, 2007; Zhou et al., 2025), we detect sharp drops in the approximated PPR scores to delineate topologically coherent communities, but here we choose the largest first difference of $|\log \text{apr}|$ as it naturally shapes a cohesive subgraph. From the resulting subgraph, community information is subsequently used to construct augmented queries to generate hard negatives $\mathcal{T}^-_{\text{hard}}$ used for the retriever. The overall subgraph extraction is outlined in Algorithm 1.

3.2.1 Query Formation

Once we have retrieved community entities V_{comm}, we synthesize a richer query collection $\mathcal{Q}^{\text{aug}}$ by perturbing and extending the original QA instance. Each new query carries additional context, such as answer spans, related entities, or semantic relations, and thus provides the retriever with more signals for hard negative chunks. Concretely, for each original QA pair, we sample subgraph entities $V_{\text{comm}} \subseteq V$ of controllable size and connectivity, and then apply the following query-transformation requirements:

Category	Example
Comparison	Compare or contrast among the given entities.
Causal	Ask for a cause or effect based on the descriptions.
Quotation	Use a quote or a redefined term from the descriptions.
Perspective Shift	Frame the question from a different perspective.

Table 1: **Query Transformation Categories.** Examples are instructions used in generating injected queries.

By instantiating the augmented query generation process using an LLM over subgraphs of two different sizes, we construct a diverse pool of augmented queries $\mathcal{Q}^{\text{aug}} = \{q_1, \ldots, q_N\}$, categorized into large and small variants, $\mathcal{Q}_L^{\text{aug}}$ and $\mathcal{Q}_S^{\text{aug}}$, respectively. Each query type is paired with a corresponding set of hard negatives, denoted as $\mathcal{T}^-_{\texttt{hard}_L}$ and $\mathcal{T}^-_{\texttt{hard}_S}$. This augmentation strategy facilitates the curriculum learning of the retriever, which enhances its capacity to capture fine-grained semantic distinctions.

3.3 Alignment-Based Fine-Tuning

After extracting answer-relevant chunks, we fine-tune the retriever using a multi-stage framework. Specifically, we introduce an answer-centric in-context scoring approach, carefully designed to mitigate weight collapse by maintaining balanced and informative gradients. Building upon this alignment-based scoring and ranking, we progressively apply a curriculum learning strategy that incrementally exposes the retriever to increasingly challenging negative samples, thereby systematically enhancing its discriminative power and ensuring stable generalization performance.

3.3.1 Alignment-Based Ranking

To retrieve positive chunks, we propose several alignment-based scoring functions that link a candidate chunk not only to the query but also to the expected answer, which is relevant to the query and sufficient to support accurate answer generation.

Forward Alignment Given a chunk t and the full question q, we evaluate the likelihood of the generator LLM (parameterized by θ) reproducing the reference answer $a = \langle a_1, \ldots, a_{|a|} \rangle$. This score quantifies the *sufficiency* of the chunk t to generate the correct answer. During scoring, the concatenated query and chunk prompt $[q; t]$ is fed to the model. We employ teacher forcing, using the ground-truth answer tokens as targets to calculate the mean token log-likelihood without them being part of the conditioning context. The forward alignment score S_f is thus defined as:

$$S_f(q, t, a) = \frac{1}{|a|} \sum_{i=1}^{|a|} \log p_\theta(a_i | q, t, a_{<i}) \quad (3)$$

This score is computed in a single forward pass by aggregating the log-softmax probabilities for each token in the ground-truth answer.

Backward Alignment Analogously, to measure the *relevance* of a chunk in linking the answer back to the original query, we pair the chunk t with the answer a and task the model with reconstructing the question q. The backward score S_b is calculated using the same teacher-forcing technique:

$$S_b(a, t, q) = \frac{1}{|q|} \sum_{j=1}^{|q|} \log p_\theta(q_j | a, t, q_{<j}) \quad (4)$$

This bidirectional scoring mechanism ensures that selected chunks are strongly correlated with the reasoning path from question to answer.

Parameter Alignment To regularize the fine-tuning process and mitigate catastrophic forgetting, we incorporate the original retriever's similarity score, S_v, as a form of parameter alignment. This score uses the cosine similarity to preserve the geometric structure learned inherent in the original retriever.

$$S_v(q, t) = \text{sim}(q, t) \quad (5)$$

The final unified score S for a given chunk t and QA q, a is a weighted combination of these three components. We did not tune these weights, as we believe adjustment is unnecessary. Instead, we adopt intuitive fixed values: equal weights for forward and backward alignment, and a slightly lower weight for parameter regularization ($\lambda_f = 1.0$, $\lambda_b = 0.3$, $\lambda_v = 1.0$).

$$S(t) = \lambda_f S_f + \lambda_b S_b + \lambda_v S_v \quad (6)$$

This unified score serves as the primary criterion for identifying high-quality positive samples for the initial stage of our fine-tuning curriculum.

3.3.2 Curriculum-Based Contrastive Finetuning

Illustrated in Figure 3. We structure the fine-tuning process as a three-stage curriculum, where the difficulty of the discrimination task increases at each

stage. This approach allows the model to first learn a robust answer-centric representation and then progressively refine it by focusing on increasingly subtle and challenging distractors.

Stage 1: Initial Answer-Centric Alignment The primary goal of this stage is to align the retriever with chunks that are highly conducive to generating the correct answer. For each query q, we calculate the unified score $S(t)$ for all candidate chunks in $\mathcal{T}$. We select the top-M chunks with the highest scores as the positive set $\mathcal{T}^+$. The retriever is then trained using a contrastive objective. Specifically, for a given positive pair (q, t^+) where $t^+ \in \mathcal{T}^+$, we use other positive chunks within the same mini-batch as in-batch negatives. This is an effective and efficient method for initial training. The loss for this stage is the InfoNCE loss:

$$\mathcal{L}_{\text{Stage1}} = -\log \frac{\exp(\text{sim}(q, t^+)/\tau)}{\sum_{t'_j \in \text{Batch}} \exp(\text{sim}(q, t'_j)/\tau)} \tag{7}$$

where $\text{sim}(\cdot, \cdot)$ is the cosine similarity from the retriever being trained and τ is a temperature hyperparameter.

Stage 2: Coarse Alignment with $\mathcal{Q}_L^{\text{aug}}$ After the initial alignment, we further enhance the retriever's robustness by incorporating hard negatives. Specifically, we leverage $\mathcal{Q}_L^{\text{aug}}$ generated from the KG. Using the retriever fine-tuned in Stage 1, we retrieve the top-K chunks for each complex query $q \in \mathcal{Q}_L^{\text{aug}}$. From this set, we exclude any chunks that appear in the ground-truth positive set $\mathcal{T}^+$. The remaining chunks constitute the hard negative set $\mathcal{T}^-_{\texttt{hard}_L}$. Compared to $\mathcal{T}^-_{\texttt{hard}_S}$, these negatives are less challenging due to their greater semantic diversity. The retriever is then further trained to distinguish the original positive chunks from these hard negatives.

$$\mathcal{L}_{\text{Stage2}} = -\log \frac{e^{s(q,t^+)}}{\sum_{t \in \mathcal{C}(q)} e^{s(q,t)}}, \tag{8}$$

where $s(q, t) = \text{sim}(q, t)/\tau$ and $\mathcal{C}(q) = \{t^+\} \cup \mathcal{T}^-_{\text{hard}}$.

Stage 3: Fine-Grained Alignment with $\mathcal{Q}_S^{\text{aug}}$ In the final stage, we further sharpen the retriever using the simple augmented queries $\mathcal{Q}_S^{\text{aug}}$, which are minor perturbations of the original query (e.g., with only distracting covariates added). We use the retriever from Stage 2 to retrieve chunks and, after filtering out the golden positives, obtain a set of "harder" negative chunks $\mathcal{T}^-_{\texttt{hard}_S}$. The training objective remains the InfoNCE loss, but with a more challenging target.

4 Experiment

4.1 Datasets

In this paper, we focus on the long-text QA task. For training, we sample 200 cases each from the Finance and Legal domains (both from the Ultradomain dataset) to generate augmented queries. For evaluation, we first select five domains from the Ultradomain benchmark in MemoRAG (Qian et al., 2024), namely Biology, Fiction, Music, Technology, and Philosophy, each represented by a distinct domain-specific dataset. We also use five LongBench (Bai et al., 2023) dataset which includes both single-document QA: NarrativeQA (Kočiskỳ et al., 2018), Qasper (Dasigi et al., 2021) and multi-document QA: HotpotQA (Yang et al., 2018), 2WikiMQA (Ho et al., 2020), and MuSiQue (Trivedi et al., 2022).

4.2 Experiment Setup

Our Framework We adopt **Qwen3-embedding** (Zhang et al., 2025b) as our base model for fine-tuning, due to its state-of-the-art performance across a wide range of downstream tasks and its strong knowledge inheritance from the Qwen LLM. To emphasize the efficiency and scalability of our framework, we utilize the smallest variant (0.6B parameters). For each subgraph, we generate 10 augmented queries and retrieve 10 positive chunks per query. To construct the negative set, we sample the top 20 retrieved chunks and exclude any that overlap with the positives.

Model Selection We first employ the latest **Gemini-2.5-Flash** (Comanici et al., 2025) as our foundational model for OpenIE. The construction of our KG follows the GraphRAG (Edge et al., 2024), but the community generation is different. The NetworkX library is utilized to execute approximated PPR when conducting a query-based community search. We then employ **Gemini-2.5-Pro** (Comanici et al., 2025) for constructing injected queries with the same ground truth as the original query. We use **GPT-4.1** to evaluate win rates via pairwise comparisons.

Baselines We compare our approach against three categories of baselines: 1) **Full**: Directly providing the entire context to an LLM. 2) Dense Re-

Table 2: **Main Evaulation results**. The evaluation metrics are F1-score / Win Rate (%), with the **best** results highlighted in bold and the second-best results underlined. The improvement rate (↑ %, (ARK- Base) / Base) is calculated based on our base model Qwen3-embedding. Cell shading indicates relative win rates compared to ARK.

METRICS	MODELS	LongBench					UltraDomain				
		nar	qas	mus	2wiki	hot	bio	fic	music	tech	phil
F1	Full	12.95	22.79	6.74	20.13	26.87	27.47	25.75	25.50	22.68	23.05
	Qwen3-embedding	19.58	23.90	14.19	21.24	35.27	32.99	29.41	34.90	38.03	34.04
	BGE-M3	18.37	23.33	**21.13**	22.86	38.64	32.52	31.72	35.34	39.13	35.97
	Stella-v5	20.90	23.39	17.08	22.13	35.45	33.85	32.41	35.02	35.16	34.09
	Jina-emb-v3	19.39	20.70	20.58	19.34	39.17	32.88	29.00	33.74	38.74	36.81
	GraphRAG	4.21	7.69	2.15	5.52	3.03	18.87	16.92	14.97	21.93	20.01
	LightRAG	2.65	3.25	1.95	3.67	2.74	16.06	14.13	15.08	12.19	14.04
	HippoRAG	11.51	21.90	13.09	**30.96**	28.71	36.13	29.23	32.94	27.15	29.06
	MemoRAG	15.49	17.96	8.74	16.57	22.79	31.08	27.87	33.26	39.14	31.98
	ARK(Ours)	**21.57**	**24.04**	20.60	23.41	**42.35**	**36.19**	**32.59**	**38.03**	**40.16**	**37.86**
	↑ %	10.2	0.6	45.2	22.41	20.1	9.7	10.8	9.0	5.6	11.2
ARK WIN RATE	Full	83.33	46.03	80.00	64.52	70.97	95.00	100.00	88.89	94.74	100.00
	Qwen3-embedding	58.33	52.54	63.46	57.14	68.89	95.00	85.71	94.74	78.57	55.56
	BGE-M3	60.00	50.77	56.00	52.54	52.83	70.59	84.62	58.82	73.33	60.00
	Stella-v5	65.67	66.67	58.00	50.00	67.39	72.22	62.50	64.71	71.43	89.47
	Jina-emb-v3	63.08	54.84	54.90	57.41	43.24	77.78	61.54	66.67	58.82	50.00
	GraphRAG	93.62	90.70	78.26	83.75	78.57	100.00	100.00	85.00	95.00	100.00
	LightRAG	96.63	96.70	91.46	91.36	96.74	100.00	100.00	95.00	100.00	100.00
	HippoRAG	87.34	53.85	58.21	34.67	60.76	77.78	44.44	62.50	70.00	70.00
	MemoRAG	80.00	75.00	66.22	57.14	67.65	92.86	94.12	84.21	87.50	88.24

trieval model: **Qwen3-embedding**: The original retriever without fine-tuning. **BGE-M3** (Chen et al., 2023): A hybrid retrieval model that integrates multiple strategies to achieve high accuracy and generalization across benchmarks. **Stella-v5** (Zhang et al., 2025a): A top-ranking retriever on the MTEB leaderboard (Muennighoff et al., 2022). **Jina-emb-v3** (Sturua et al., 2024): A powerful and widely-used multilingual, multi-task embedding model. 3) Advanced RAG methods: **GraphRAG** (Edge et al., 2024): Utilizes LLM-generated knowledge graphs and the Leiden algorithm for hierarchical retrieval. **LightRAG** (Guo et al., 2025): Combines dual-level retrieval with vector and graph structures. **HippoRAG** (Gutiérrez et al., 2024): Leverages PPR for community-based retrieval.l. **MemoRAG** (Qian et al., 2024): Employs a lightweight long-context LLM to construct global memory and generate retrieval cues.

For online QA, we adopt **Mistral-7B-Instruct-v0.2-32K** (Jiang et al., 2023) as the default generator to avoid potential pretraining contamination. Effectiveness is evaluated with F1-score (following each dataset's original setup) and pairwise win rate averaged over five runs for consistency.

4.2.1 Running Environment

ARK is implemented in Python 3.10 with PyTorch 2.7.1. All experiments are conducted on a machine equipped with 8 NVIDIA H20 GPU. We utilize `NetworkX`, `PyTorch`, `transformers`, and `sentence transformers` as the main libraries. In addition, we use `Ollama` for inference and `SWIFT` (Zhao et al., 2024) to finetune the Qwen3-embedding.

4.3 Performance Evaluation

Table 7 reports the F1 scores and pairwise win-rate on the LongBench and UltraDomain benchmarks. Overall, ARK consistently outperforms its base model (Qwen3-embedding) across all datasets and achieves state-of-the-art performance compared to both KG-based baselines and top-ranking dense retrievers. In particular, ARK attains consistently higher pairwise win rates across the 10 evaluated datasets, outperforming both graph-based approaches (e.g., GraphRAG, LightRAG) and strong dense encoders (e.g., BGE-M3, Stella, Jina, Qwen3). Notably, it exceeds a 50% win rate on the majority of benchmarks.

ARK shows strong generalization beyond its training domains (Finance and Legal), maintaining robust performance on unseen datasets, showing that our KG-guided curriculum not only improves retrieval accuracy but also enhances the ability to surface contextually meaningful evidence beyond the training distribution. Gains are especially pronounced on reasoning-intensive tasks such as

MuSiQue and HotpotQA, where retrieval must synthesize dispersed evidence. By explicitly optimizing for *answer sufficiency*, the retriever learns to favor chunks that are both relevant and sufficient for generating faithful answers. On certain multi-hop tasks like 2Wiki, graph-centric methods such as HippoRAG remain competitive due to their traversal advantage. Nevertheless, ARK matches or surpasses their performance without requiring costly graph construction or long-context LLMs, thus offering higher efficiency. Moreover, unlike resource-intensive systems such as MemoRAG, which demand end-to-end training, our approach fine-tunes only the retriever. This modular design enables seamless integration into existing RAG pipelines, making ARK both practical and scalable. In summary, by emphasizing *answer sufficiency* while preserving *query similarity*, ARK consistently yields relevant and sufficient evidence for long-context retrieval without altering the underlying RAG architecture.

STAGE	SCORING	mus	hot	bio	phil
Original	-	14.19	35.27	32.99	34.04
1st	Full	18.44	40.96	32.35	34.89
	w/o F.A.	14.31	36.40	31.36	32.95
	w/o B.A.	16.73	39.74	33.57	34.12
	w/o P.A.	15.29	37.03	33.18	34.73
2nd	Full	18.47	42.13	34.74	36.11
3rd (ARK)	Full	**20.60**	**42.35**	**36.19**	**37.86**

Table 3: **Ablation study**. STAGE denotes finetuning stages (*Original* is the base model), and SCORING specifies alignment types: Forward (F.A.), Backward (B.A.), and Parameter (P.A.).

4.4 Ablation Study

To better understand the role of each design component, we conduct ablations over the three alignment strategies and the curriculum process (Table 3). Forward alignment proves most critical, as it directly measures chunk sufficiency for generating correct answers; its removal leads to the largest drop in performance. Backward alignment further supports complex reasoning by enforcing semantic coherence, and removing it causes retrieved passages to match superficially but lack utility. Parameter alignment, though less dominant, stabilizes training by anchoring the embedding space, reducing overfitting and collapse in noisier domains.

The curriculum stages also show clear benefits: Stage 1 (answer-aligned positives with in-batch negatives) provides a strong initial boost, Stage 2 introduces coarse hard negatives from larger subgraphs to expose the model to more diverse distractors and enhance robustness, and Stage 3 employs fine-grained negatives from smaller subgraphs for sharper discrimination. This progressive structure teaches the retriever not only to capture relevance but also to identify evidence truly necessary for accurate answer generation.

METHOD	mus	2wiki	nar	tech	phil
Llama3.1-8B-Instruct					
QWEN3	9.63	28.26	**17.70**	19.91	20.72
ARK	**11.48** (+)	**36.00** (+)	15.04 (-)	**21.47** (+)	**21.17** (+)
Qwen2.5-7B-Instruct					
QWEN3	10.74	27.65	16.70	19.46	22.62
ARK	**14.61** (+)	**29.47** (+)	**16.77** (+)	**20.09** (+)	**24.64** (+)

Table 4: Transferability of the ARK to different generators.

4.5 Transferability Across Generators

We further investigate whether the retriever fine-tuned on QWEN3-EMBEDDING can transfer to other generators without additional adaptation. We evaluate end-to-end QA by directly plugging ARK into two instruction-tuned LLMs: LLAMA-3.1-8B-INSTRUCT and QWEN2.5-7B-INSTRUCT. As shown in Table 4, ARK consistently improves performance across both generators, indicating the generalization of out training method. A slight drop on NAR with Llama-3.1 suggests interaction between generator decoding preferences and retrieval signals, motivating future work on generator-aware optimization or joint training aligned with generation loss.

5 Conclusion

We propose ARK, a fine-tuning framework for retrievers that uses curriculum learning to integrate KG knowledge through hard-negative generation. ARK constructs a compact knowledge subgraph using LLM-generated KG and PPR-based cohesive community search; selects positive examples via three-way alignment while preserving the base encoder's similarity signal; and applies a three-stage curriculum with augmented query retrieval that incrementally incorporates harder negative chunks. The framework requires no architectural modifications, which fits standard RAG pipelines. Experiments on UltraDomain and LongBench demonstrate consistent improvements in F1 score and pairwise win-rate.

Limitations

While our framework demonstrates strong effectiveness across diverse domains and tasks, it also has several limitations. First, our evaluation is constrained to publicly available benchmarks, which may not fully capture the diversity of real-world applications. In addition, while inference is KG-free, our training pipeline depends on an LLM-derived KG for hard-negative mining; noise in entity extraction or linking can affect the curriculum quality. We leave these aspects for future work.

References

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

Reid Andersen and Fan Chung. 2007. Detecting sharp drops in pagerank and a simplified local partitioning algorithm. In *International Conference on Theory and Applications of Models of Computation*, pages 1–12. Springer.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, and 1 others. 2022. Improving language models by retrieving from trillions of tokens. In *International conference on machine learning*, pages 2206–2240. PMLR.

Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu. 2024. Rq-rag: Learning to refine queries for retrieval augmented generation. *arXiv preprint arXiv:2404.00610*.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2023. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *Preprint*, arXiv:2309.07597.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, and 1 others. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. *arXiv preprint arXiv:2105.03011*.

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

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *Preprint*, arXiv:2212.10496.

Zirui Guo, Lianghao Xia, Yanhua Yu, Tu Ao, and Chao Huang. 2025. Lightrag: Simple and fast retrieval-augmented generation. *Preprint*, arXiv:2410.05779.

Bernal Jiménez Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. 2024. Hipporag: Neurobiologically inspired long-term memory for large language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Taher H Haveliwala and Sepandar D Kamvar. 2003. The second eigenvalue of the google matrix. Technical report, Citeseer.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. *arXiv preprint arXiv:2011.01060*.

Gautier Izacard and Edouard Grave. 2021. Distilling knowledge from reader to retriever for question answering. In *ICLR 2021-9th International Conference on Learning Representations*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, and 1 others. 2023. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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*, pages 6769–6781.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, and 1 others. 2021. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363.

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.

Tianle Li, Ge Zhang, Quy Duc Do, Xiang Yue, and Wenhu Chen. 2024. Long-context llms struggle with long in-context learning. *arXiv preprint arXiv:2404.02060*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Hongjin Qian, Peitian Zhang, Zheng Liu, Kelong Mao, and Zhicheng Dou. 2024. Memorag: Moving towards next-gen rag via memory-inspired knowledge discovery. *arXiv preprint arXiv:2409.05591*.

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.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. 2020. Contrastive learning with hard negative samples. *arXiv preprint arXiv:2010.04592*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D. Manning. 2024. RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval. *arXiv preprint*. ArXiv:2401.18059 [cs].

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*.

Saba Sturua, Isabelle Mohr, Mohammad Kalim Akram, Michael Günther, Bo Wang, Markus Krimmel, Feng Wang, Georgios Mastrapas, Andreas Koukounas, Andreas Koukounas, Nan Wang, and Han Xiao. 2024. jina-embeddings-v3: Multilingual embeddings with task lora. *Preprint*, arXiv:2409.10173.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

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.

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

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025a. Jasper and stella: distillation of sota embedding models. *Preprint*, arXiv:2412.19048.

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

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. 2024. Swift:a scalable lightweight infrastructure for fine-tuning. *Preprint*, arXiv:2408.05517.

Jiawei Zhou, Kai Wang, Jianwei Wang, Kunpeng Zhang, and Xuemin Lin. 2025. Comet: An interactive framework for efficient and effective community search via active learning. *INFORMS Journal on Computing*.

A KG-based Query Generation

The KG serves as a core component of our framework, providing structured semantic representations that enable both entity-level reasoning and query augmentation. In this section, we detail the KG-construction, hyperparameters, and prompts used for Entity Extraction, and Query Generation.

A.1 KG construction

Our KG construction pipeline transforms unstructured text into a structured graph of entities and relations. Using an LLM-based extraction process followed by embedding-driven augmentation, we ensure semantic consistency and connectivity between related concepts. The algorithm 2 outlines this process.

Algorithm 2: KG Construction

Input: Context `context`, Chunk size B and overlap size b, Selected generator `LLM`, Embedding model `EMB`, Similarity threshold τ

Output: Generated KG $G(V, E)$

```
// 1. Chunking and Extraction
chunks ← Chunk(context, B, b)                    ▷ Using token-based chunking
Entities, Rels ← {}, {}
foreach chunk in chunks do
    llm_output ← LLM(prompt, chunk)
    (extracted_ents, extracted_rels) ← ParseLLMOutput(llm_output)
    Entities ∪ (ent, chunk_id) for ent in extracted_ents
    Rels ∪ (tuple, chunk_id) for tuple in extracted_rels
end
G ← KG_generation(Entities, Rels)
                    ▷ Construct KG using LLM-generated entities and relations
// 2. Graph Augmentation
sim_matrix ← GetCosSimilarityMatrix(G.nodes)
                    ▷ Using embedding model to calculate cosine similarity
for i from 0 to sim_matrix.rows − 1 do
    for j from i + 1 to sim_matrix.cols − 1 do
        if sim_matrix[i, j] > τ then
            src ← ent[i]
            tgt ← ent[j]
            tuple ← (src, Rel_aug, tgt)
            G.InsertEdge(tuple, None)
        end
    end
end
return G
```

A.2 Hyperparmeter

Table 5 summarizes the key hyperparameters used in document processing, KG generation, and PPR retrieval. The default settings are empirically chosen to balance computational efficiency and representation quality.

Table 5: Key hyperparameters used in KG generation.

Category	Parameter	Default
Document Processing	LLM	Gemini-2.5 Flash
	B	512
	b	12
Graph Augmentation	EMB	BGE-m3
	τ	0.8
PPR	α	0.85
	ϵ	1e-4
	k	200

A.3 Entity Extraction Prompt

To build a coherent KG, we use a prompt that explicitly instruct the LLM to extract entities, their attributes, and relationships in a structured format.

Entity Extraction Prompt

Given a text document that is potentially relevant to this activity, your task is to identify all entities of specific types and the relationships among them.
Steps:
1. Identify Entity Types:
   ...
2. Extract Entities:
   ...
   Format each entity as:
   `("entity"<|>entity_name<|>entity_type<|>entity_description>)`
3. Identify Relationships:
   ...
   Format each relationship as:
   `("relationship" <|>source_entity<|>target_entity<|>relationship_description<|> relationship_strength>)`

...

Listing 1: A snippet of the prompt used for entity and relationship extraction. The full prompt provides detailed instructions and examples to the LLM.

A.4 Query Generation Prompt

Beyond KG construction, we employ a query generation process to expand the dataset and develop the curriculum learning pipeline. This prompt guides an LLM to craft semantically diverse yet answer-consistent questions by leveraging entity-level context from the KG.

Query Generation Prompt

You are an expert in creating complex and confusing questions for educational purposes. Your task is to generate 10 distinct and challenging questions based on an original question-answer pair and a set of related entities with their descriptions.
The goal is to formulate questions that are semantically different from the original but lead to the exact same answer. These new questions should be confusing by design, incorporating details from the provided entity descriptions to misdirect or challenge the user's understanding.
Input:
You will receive the following in JSON format:
- original_question: A straightforward question.
- answer: The correct and sole answer to the original question.
- entities: A list of objects, where each object contains:
  - name: The name of the entity.
  - type: The category of the entity.
  - descriptions: A list of strings, each describing a different aspect of the entity.

Task:
Generate 10 new questions (we'll call them "confusing questions").
Requirements for Confusing Questions:
1. Same Answer: Every generated question must have the exact same answer as the original question.
2. Incorporate Entities: Each question should subtly weave in information from the entities and their descriptions.
3. Variety: The questions should be diverse in their structure and focus. You should not include exact wording / entities in the original question / answer. For example, you can:
...
4. Clarity and Grammar: Despite being confusing, the questions must be grammatically correct and coherent.
Output Format:
Produce a single JSON object with one key, confusing_questions, which contains a list of 10 string questions.
...

Listing 2: A snippet of the prompt used for generating confusing questions. The LLM is instructed to use provided entities to create challenging reformulations of an original question.

B Alignment-based Finetuning

This section describes the alignment-based finetuning procedure, which enables the model to better evaluate the quality and relevance of retrieved chunks. The alignment module computes log-likelihood scores for given question–answer pairs relative to context passages, serving as a signal for measuring faithfulness and guiding downstream retrieval calibration. Model finetuning is performed using the MS-Swift library (used primarily for fine-tuning Qwen-series model).

B.1 Fintuning Pipeline

The complete training workflow of our alignment-based retriever optimization is summarized in Algorithm 3, which details the three-stage curriculum described in Section 3.3.

B.2 Alignment Prompt

The alignment prompt is designed to evaluate the faithfulness of model-generated answers relative to the provided context. In the forward alignment setting, the model is prompted to produce an answer strictly grounded in a provided context. Conversely, reverse alignment evaluates whether a given answer can be

Algorithm 3: Three-Stage Alignment-Based Finetuning (ARK)

Input: QA corpus $\{(q, a, \mathcal{T})\}$; pretrained generator p_θ; retriever $\mathrm{sim}(\cdot,\cdot)$; weights $(\lambda_f, \lambda_b, \lambda_v)$; temperature τ; augmented query sets $\mathcal{Q}_L^{\mathsf{aug}}, \mathcal{Q}_S^{\mathsf{aug}}$.

Output: Fine-tuned retriever parameters.

Stage 0: Alignment Scoring
foreach $(q, a, \mathcal{T})$ **do**
  foreach $t \in \mathcal{T}$ **do**
    $S_f = \frac{1}{|a|}\sum_i \log p_\theta(a_i|q, t, a_{<i})$
    $S_b = \frac{1}{|q|}\sum_j \log p_\theta(q_j|a, t, q_{<j})$
    $S_v = \mathrm{sim}(q, t)$
    $S(t) = \lambda_f S_f + \lambda_b S_b + \lambda_v S_v$
  end
  $\mathcal{T}^+ = \mathrm{TopM}_{t\in\mathcal{T}}\, S(t)$
end

Stage 1: Initial Answer-Centric Alignment
foreach *mini-batch* $\mathcal{B}$ **do**
  foreach $(q, t^+) \in \mathcal{B}$ **do**
    $s(q, t) = \mathrm{sim}(q, t)/\tau$
    $\mathcal{L}_{\text{Stage1}}(q) = -\log \frac{e^{s(q,t^+)}}{\sum_{t'\in\text{Batch}} e^{s(q,t')}}$
  end
  Update retriever by $\nabla \frac{1}{|\mathcal{B}|}\sum \mathcal{L}_{\text{Stage1}}$
end

Stage 2: Coarse Alignment with $\mathcal{Q}_L^{\mathsf{aug}}$
foreach $q' \in \mathcal{Q}_L^{\mathsf{aug}}$ **do**
  Retrieve Top-K chunks and form $\mathcal{T}^-_{\mathsf{hard}_L}(q')$
end
foreach *mini-batch* $\mathcal{B}$ **do**
  $\mathcal{L}_{\text{Stage2}}(q) = -\log \frac{e^{s(q,t^+)}}{\sum_{t\in\mathcal{C}(q)} e^{s(q,t)}}, \mathcal{C}(q) = \{t^+\} \cup \mathcal{T}^-_{\mathsf{hard}_L}$
  Update retriever
end

Stage 3: Fine-Grained Alignment with $\mathcal{Q}_S^{\mathsf{aug}}$
foreach $q'' \in \mathcal{Q}_S^{\mathsf{aug}}$ **do**
  Retrieve Top-K chunks and form $\mathcal{T}^-_{\mathsf{hard}_S}(q'')$
end
foreach *mini-batch* $\mathcal{B}$ **do**
  $\mathcal{L}_{\text{Stage3}}(q) = -\log \frac{e^{s(q,t^+)}}{\sum_{t\in\mathcal{C}'(q)} e^{s(q,t)}}, \mathcal{C}'(q) = \{t^+\} \cup \mathcal{T}^-_{\mathsf{hard}_S}$
  Update retriever
end
return Fine-tuned retriever

justified by its corresponding context. Together, these two directions enable a bidirectional evaluation of model reliability through log-likelihood estimation, which is later integrated into the retrieval scoring and finetuning stages.

Forward Alignment Prompt

You are given a context and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the context, write ünanswerable. If the question is a yes/no question, answer ÿes, ño, or ünanswerable. Do not provide any explanation.
Context: {chunk}
Question: {question}
Answer:

Listing 3: The forward prompt used to retrieve log-likelihood.

B.3 Hyperparameter

Table 6 summarizes the key hyperparameters used during alignment-based finetuning. The process employs **full-parameter tuning** on the embedding model to ensure that the learned representations are optimally aligned with the retrieval task objectives. The configuration is chosen by default and not through grid-search since the fine-tuning stage is relatively stable.

Table 6: Training Hyperparameters

Parameter	Default
`Epochs`	10
`batch size`	2
`gradient accumulation steps`	8
`Learning rate`	6e-6
`Loss`	infonce

C Inference

This section describes the inference components of our system. The inference module generates final answers conditioned on those retrieved ones.

C.1 Inference Prompt

The inference prompt serves as the main instruction template for generating final answers from retrieved text chunks. It is intentionally concise and task-oriented, ensuring that responses are direct, factual, and free from unnecessary reasoning chains.

Inference Prompt

You are given a scientific article and a question. Answer the question as concisely as you can, using a single phrase or sentence if possible. If the question cannot be answered based on the information in the article, write "unanswerable". If the question is a yes/no question, answer "yes", "no", or "unanswerable". Do not provide any explanation.
Context: {chunk}
Question: {question}
Answer:

Listing 4: The general prompt used to generate answers from the retrieved context.

D Win-rate Evaluation

This section presents the evaluation protocol and results used to compare the performance of different retrieval and reasoning models. We adopt an LLM-based evaluator that systematically measures pairwise model performance through criteria grounded in faithfulness and conciseness.

D.1 Win-rate Prompt

To ensure consistent and interpretable evaluation, we employ a structured prompt that directs an LLM to act as a neutral expert judge. The evaluator receives a ground truth reference, a question, and two candidate answers. It then performs a two-stage comparison: first applying a disqualification rule to detect unsupported answers, and subsequently assessing faithfulness (support from the ground truth), conciseness (brevity without loss of correctness) and overall winner.

Win-rate Prompt

You are an expert evaluator. Your task is to rigorously assess two answers to a specific question, based on a provided Ground Truth. You will use two criteria: Faithfulness and Conciseness.
...
Evaluation Rules:
Disqualification Rule (Primary Check):
First, check if either answer explicitly states that the Ground Truth document does not contain enough information or evidence to answer the Question.
...
Evaluation Criteria (Secondary Check):
Faithfulness: The degree to which the answer is exclusively and accurately supported by the provided Ground Truth document.
Conciseness: The degree to which the answer avoids mentioning excessive entities or relationships that are not essential for answering the Question.
...
Output Format:
Output your complete evaluation in the following JSON format.

```
{
  "Faithfulness": {"Winner": "[Answer 1, Answer 2, Tie, or None]", "Explanation": ...},
  "Conciseness": {"Winner": "[Answer 1, Answer 2, Tie, or None]", "Explanation": ...},
  "Overall Winner": {"Winner": "[Answer 1, Answer 2, Tie, or None]", "Explanation": ...}
}
```

Listing 5: A snippet of the prompt for LLM-based evaluation. The prompt defines a strict set of rules, including a disqualification rule, and requires a structured JSON output.

D.2 Additional Results

Table 7 extends win-rate comparisons to Faithfullness and Conciseness.

Table 7: **Additional win-rate results**. winrate $= 1/|q| \times \Sigma_q[1_{\text{Answer 1}}/(1_{\text{Answer 1}} + 1_{\text{Answer 2}})]$. Cell shading indicates relative win rates compared to ARK.

Criteria	Models	LongBench					UltraDomain				
		nar	qas	mus	2wiki	hot	bio	fic	music	tech	phil
Faithfulness	Full	83.10	58.00	75.00	64.44	71.11	100.00	100.00	94.12	100.00	100.00
	Qwen3-embedding	59.57	54.17	63.16	61.11	77.42	100.00	92.31	94.74	78.57	58.82
	BGE-M3	62.26	55.10	63.41	58.14	56.76	70.59	91.67	64.71	76.92	63.16
	Stella-v5	68.52	84.62	64.29	40.00	61.76	72.22	66.67	70.59	76.92	89.47
	Jina-emb-v3	60.78	62.50	55.88	53.85	45.45	77.78	66.67	66.67	60.00	55.56
	GraphRAG	89.13	88.75	77.27	73.33	76.74	100.00	100.00	90.00	95.00	100.00
	LightRAG	76.40	91.86	76.71	76.32	89.41	100.00	70.00	100.00	100.00	100.00
	HippoRAG	69.01	65.00	46.03	40.91	60.87	77.78	37.50	50.00	60.00	70.00
	MemoRAG	79.17	78.79	68.33	57.38	65.38	93.33	93.75	84.21	87.50	88.24
Conciseness	Full	90.24	52.31	81.71	60.94	75.00	94.74	100.00	83.33	94.74	100.00
	Qwen3-embedding	55.17	50.00	65.31	57.69	62.79	90.00	78.57	73.68	76.92	44.44
	BGE-M3	55.88	46.97	48.94	56.90	51.92	70.59	69.23	38.89	66.67	55.00
	Stella-v5	72.46	66.67	61.22	60.00	71.11	72.22	62.50	61.11	71.43	78.95
	Jina-emb-v3	62.50	50.00	55.10	61.82	50.00	66.67	53.85	60.00	58.82	40.00
	GraphRAG	95.83	96.63	95.74	95.29	97.78	100.00	100.00	90.00	90.00	100.00
	LightRAG	98.94	96.70	96.34	94.19	97.85	100.00	100.00	95.00	100.00	100.00
	HippoRAG	79.22	45.45	48.57	17.33	48.15	33.33	22.22	50.00	60.00	40.00
	MemoRAG	86.75	82.19	72.37	67.09	70.42	86.67	94.12	84.21	81.25	76.47