

KARE-RAG: Knowledge-Aware Refinement and Enhancement for RAG

Yongjian Li^{1,2,*}, HaoCheng Chu^{3,*}, Yukun Yan^{4,†}, Zhenghao Liu⁸,
Shi Yu⁴, Zheni Zeng⁴, Ruobing Wang⁹, Sen Song^{1,2,†},
Zhiyuan Liu^{4,5,6,7}, Maosong Sun^{4,5,6,7}

¹School of Biomedical Engineering, Tsinghua Medicine, Tsinghua University

²Tsinghua Laboratory of Brain and Intelligence ³School of Informatics, Xiamen University

⁴Department of Computer Science and Technology, Tsinghua University, Beijing

⁵Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing

⁶Institute for Artificial Intelligence, Tsinghua University, Beijing

⁷Jiangsu Collaborative Innovation Center for Language Ability, Jiangsu Normal University, Xuzhou

⁸School of Computer Science and Engineering, Northeastern University ⁹Institute of Information Engineering, Chinese Academy of Sciences

*Equal contribution. [†]Corresponding author. yj-li23@mails.tsinghua.edu.cn

Abstract

Retrieval-Augmented Generation (RAG) equips large language models with external knowledge and is central to knowledge-intensive tasks. As RAG systems enter real-world use, generators must reliably leverage retrieved evidence. Recent fine-tuning methods improve adaptation to RAG scenarios, but optimization remains challenging because retrieval may return incomplete, fragmented, noisy, or conflicting contexts. Complex tasks further require fine-grained evidence dependencies. These challenges make high-quality supervision costly and limit generalization.

We present KARE-RAG (Knowledge-Aware Refinement and Enhancement for RAG), a training-time scaffolded alignment framework. It uses structured knowledge representations as temporary scaffolds to expose evidence organization, support localized factual refinement, and construct fine-grained preference pairs. In our main implementation, an expert LLM refines a lightweight graph-structured evidence sketch. The generator is optimized with token-weighted Dense Direct Preference Optimization (DDPO), which focuses learning on edited scaffold regions. Scaffolds are used only for data construction and training supervision. At inference time, the model runs as standard Vanilla RAG without graph construction, extra retrieval, or latency overhead. Experiments show that KARE improves transfer across the evaluated QA and relation extraction datasets with limited training data, while leaving general capabilities largely unchanged. KARE can also complement existing RAG training objectives as an additional alignment stage.

1 Introduction

Retrieval-Augmented Generation (RAG) (Lewis et al., 2020; Shi et al., 2024) has become a cornerstone of knowledge-intensive tasks, enabling

large language models (LLMs) to combine parametric knowledge with real-time information retrieval. Despite its widespread adoption, RAG systems face inherent challenges in processing retrieved documents. Evidence is often fragmented across multiple sources, requiring models to piece together complete information from partial segments (Xu et al., 2024a). Retrieved documents may also present irrelevant or conflicting claims that require careful reconciliation (Wu et al., 2024; Gao et al., 2023; Yoran et al., 2023; Xu et al., 2024b; Longpre et al., 2021; Liu et al., 2024a), as even state-of-the-art retrieval systems cannot guarantee perfect relevance in returned documents.

Existing research has made significant progress by improving or adapting retrieval strategies (Jiang et al., 2023; Gao et al., 2023; Trivedi et al., 2023; Mao et al., 2024; Ma et al., 2023; Jeong et al., 2024) or developing post-retrieval evidence processing mechanisms (Yu et al., 2024b; Xu et al., 2023; Fang et al., 2024). Recent studies show that improving how the generation module processes and integrates multi-source knowledge is an equally important, complementary approach (Singh et al., 2021; Lin et al., 2024; Asai et al., 2024; Li et al., 2024).

A common strategy is to fine-tune the model on RAG-style instruction data or final-answer supervision (Lin et al., 2024; Asai et al., 2024). RA-DiT, for example, adapts RAG through dual instruction tuning and improves how the generator uses retrieved contexts (Lin et al., 2024). However, answer-level supervised fine-tuning (SFT) provides only coarse supervision for complex evidence use and may overfit to the training distribution. Preference-based alignment offers another route. DDR constructs metric-derived preference pairs and applies Direct Preference Optimization (DPO) (Rafailov et al., 2023) to optimize genera-

tion under retrieved contexts (Li et al., 2024). Such methods improve end-to-end behavior, but their supervision still falls mainly on final outputs. They provide limited guidance for intermediate evidence filtering, organization, and integration, where errors often arise in multi-hop reasoning or conflict resolution. Moreover, with intricate queries and noisy retrieval, sampling reliable positive examples can be difficult. This often requires large amounts of high-quality training data (Li et al., 2024), motivating more targeted supervision for how RAG generators process evidence.

To address these issues, we develop KARE-RAG (Knowledge-Aware Refinement and Enhancement for RAG), a scaffolded training strategy that teaches models to better utilize retrieved documents. We use KARE as a compact name in tables and experimental discussions. Figure 1 provides a schematic overview of the framework. Rather than supervising only final answers, KARE-RAG introduces temporary training scaffolds that make evidence organization explicit during data construction. These scaffolds act as editable intermediate workspaces. An advanced LLM refines localized factual errors while preserving the surrounding organization, yielding high-quality preference pairs that differ mainly in critical regions. For optimization, we adopt Dense Direct Preference Optimization (DDPO) (Yu et al., 2024a) with an SFT regularizer (Liu et al., 2024b) and assign token-level weights to edited scaffold regions. Crucially, the scaffolds are used only for alignment. During inference, the optimized model operates through a standard Vanilla RAG pipeline without additional computational overhead.

Empirically, KARE improves transfer across the evaluated QA and relation extraction datasets while retaining the standard Vanilla RAG inference pipeline. It obtains the best results on most cross-dataset metrics, showing that scaffolded alignment can improve evidence use across task formats. Its scaffold-based supervision can also be combined with existing RAG training objectives as a complementary alignment stage.

2 Related Work

Retrieval-Augmented Generation (RAG) improves knowledge-intensive generation by conditioning language models on external evidence rather than relying only on parametric memory (Lewis et al., 2020; Karpukhin et al., 2020; Guu et al., 2020). Its

effectiveness depends not only on retrieving relevant passages but also on whether the generator can use imperfect contexts. Retrieved evidence may be incomplete, irrelevant, conflicting, or difficult to use when useful information is dispersed across a long context (Yoran et al., 2023; Longpre et al., 2021; Liu et al., 2024a). Prior work addresses these issues through inference-time mechanisms such as context compression, note construction, and iterative retrieval with reasoning (Xu et al., 2023; Yu et al., 2024b; Trivedi et al., 2023; Xu et al., 2024a). These methods improve how evidence is selected or presented during generation and show why evidence organization matters when retrieval is imperfect.

Another line of work adapts RAG systems through supervised or preference-based training. RA-DiT uses dual instruction tuning to adapt both the language model and the retriever for retrieval-augmented language modeling (Lin et al., 2024). RAFT fine-tunes models for domain-specific open-book QA with relevant and distractor documents (Zhang et al., 2024). Self-RAG trains a model to retrieve, generate, and critique using self-reflection signals (Asai et al., 2024). DDR introduces differentiable data rewards to optimize RAG modules according to their effects on system-level performance (Li et al., 2024), and RPO incorporates retrieval relevance into preference optimization for robust RAG generation (Yan et al., 2025). These approaches provide strong ways to adapt RAG behavior, but their supervision is mainly tied to final answers, retrieval behavior, or task-level rewards.

Preference optimization provides a natural tool for learning from paired supervision. DPO directly optimizes a language model from preference pairs without training a separate reward model (Rafailov et al., 2023). Recent work further shows that finer-grained preference signals can be useful when correctness depends on localized errors. RLHF-V collects segment-level correction feedback for hallucination alignment and optimizes with dense direct preference signals (Yu et al., 2024a), while Step-DPO applies step-wise preferences to long-chain reasoning rather than judging only complete answers (Lai et al., 2024). This observation is especially relevant for RAG, where a small number of entities, attributes, or relations can determine whether an answer is grounded. It suggests the value of training signals that identify local evidence-use errors rather than judging the full output alone.

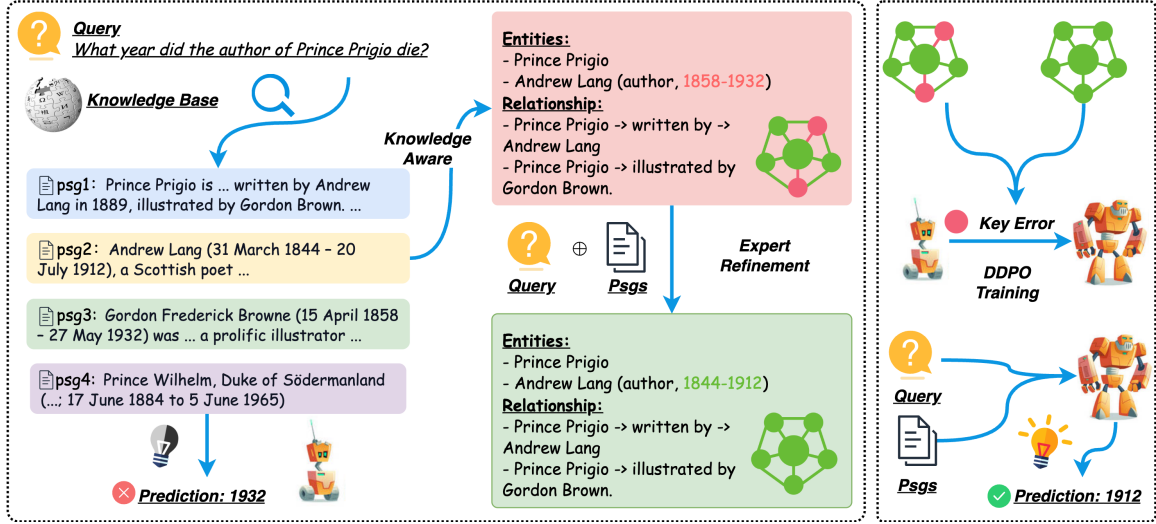


Figure 1: Overview of KARE-RAG. The left panel illustrates the data generation procedure. Retrieved passages can include sufficient evidence for the query, while the generator may still derive an incorrect response. KARE-RAG prompts the generator to construct a scaffold that localizes the evidence-use error. An expert model then revises the scaffold to form a DDPO preference pair. The right panel shows that the optimized generator returns to standard Vanilla RAG inference after training.

3 Method

In this section, we formalize Knowledge-Aware Refinement and Enhancement for RAG (KARE-RAG), shown in Figure 1. During training data construction, Knowledge-Aware Sampling asks the generator to build a scaffold from the query and retrieved documents before answer generation. The scaffold records how the generator selects and connects evidence, making it possible to locate errors in the intermediate organization. An expert-guided refinement pipeline then edits flawed scaffolds while preserving their overall format, producing preference pairs with localized factual differences. We train the generator with Dense Direct Preference Optimization (DDPO), where token-level weights are assigned to the edited regions. After training, the scaffold interface is removed. The model receives the same inputs as standard Vanilla RAG and generates answers directly. The following sections detail each component.

3.1 Knowledge-Aware Sampling

In a standard Retrieval-Augmented Generation (RAG) process, the LLM receives a query q and a set of retrieved documents $D = \{d_1, \dots, d_n\}$ to generate an answer. Retrieved documents may contain irrelevant, incomplete, or conflicting information, which can affect generation quality.

During training, Knowledge-Aware Sampling asks the generator to organize the query and re-

trieved documents into a temporary scaffold before producing the final answer. The scaffold records which evidence the model selects and how the selected evidence is connected. In our main implementation, it takes the form of a lightweight graph-style sketch. We denote this scaffold construction step as

$$s = \text{LLM}(\text{Instruct}_{\text{scaf}}, q \oplus D) \quad (1)$$

where $\oplus$ denotes concatenation, $\text{Instruct}_{\text{scaf}}$ is the scaffold construction instruction, and s denotes the resulting scaffold.

This scaffold provides the refinement stage with a concrete object to inspect. Compared with final-answer supervision, it exposes intermediate evidence organization and separates factual mistakes from unrelated wording differences. When the scaffold uses a structured format, its discrete units also make it easier to identify which entities, attributes, or relations require correction. This property is important for constructing preference pairs that differ in small factual regions rather than in broad surface form.

3.2 Scaffold Refinement

Large language models can often produce format-valid scaffolds, but these scaffolds may still contain localized factual inaccuracies when processing retrieved documents. Scaffold Refinement constructs preference pairs by correcting such errors while preserving the surrounding scaffold structure. For each

query, we generate an initial scaffold s^- and use it to produce an answer, retaining only cases where the answer is incorrect. We then verify whether the retrieved documents support the gold answer. If the evidence is sufficient, an expert model minimally edits s^- into s^+ under document-grounded constraints while preserving the scaffold format. The pair (s^+, s^-) is retained only when s^+ leads the generator to the correct answer. Appendix A.2 provides the full procedure.

3.3 Optimization Approach

The scaffold pairs produced by the refinement stage usually differ only in small factual regions. Standard DPO (Rafailov et al., 2023) treats all output tokens uniformly, so the signal from a corrected entity, attribute, or relation can be diluted by many unchanged scaffold tokens. We therefore adopt Dense Direct Preference Optimization (DDPO) (Yu et al., 2024a), which keeps the pairwise preference objective but computes scaffold scores with higher weights on edited tokens. Let x denote the training input and (s^+, s^-) denote the refined and flawed scaffold pair. The weighted DPO objective is

$$\mathcal{L}_{\text{DDPO}} = -\mathbb{E}_{(x, s^+, s^-)} [\log \sigma(\beta \Delta^w)], \quad (2)$$

where

$$\begin{aligned} \Delta^w(x, s^+, s^-) &= \ell_\theta^w(s^+ | x) - \ell_\theta^w(s^- | x) \\ &\quad - \ell_{\text{ref}}^w(s^+ | x) + \ell_{\text{ref}}^w(s^- | x). \end{aligned} \quad (3)$$

We compute the scaffold score with token-level weights:

$$\begin{aligned} \ell_\theta^w(s | x) &= \sum_{t=1}^T w_t \log p_\theta(s_t | x, s_{<t}), \\ w_t &= 1 + (\gamma - 1) \mathbf{1}[t \in C]. \end{aligned} \quad (4)$$

Here C is the set of modified token positions, and $\gamma > 1$ controls how strongly these tokens are emphasized. We obtain C by applying a sequence-level diff between s^- and s^+ , and then mapping the edited spans to token indices. This affects only training loss computation and introduces no inference overhead.

Because preference optimization can suffer from overoptimization (Liu et al., 2024b), we add an SFT regularizer on the refined scaffold to prevent reward collapse. The final loss is

$$\mathcal{L} = \mathcal{L}_{\text{DDPO}} - \alpha \sum_{t=1}^T \log p_\theta(s_t^+ | x, s_{<t}^+), \quad (5)$$

where α controls the strength of the regularization. This term anchors the policy to valid document-grounded scaffolds while DDPO focuses the preference signal on edited regions.

Although the optimization target is defined over scaffold outputs, KARE does not use scaffolds as a test-time interface. Scaffold generation serves as an auxiliary training target that exposes localized evidence-use corrections during alignment. These corrections update the same generator parameters used for standard RAG answering. Once alignment is complete, we remove the scaffold instruction and query the model with the standard RAG prompt. The model therefore receives the same input format as Vanilla RAG and directly generates an answer. The pipeline-transfer results in Appendix A.12 support that the learned behavior is not tied to a specific scaffold prompt.

4 Experimental Settings

In this section we first introduce our experimental settings, including datasets, evaluation metrics, and implementation details.

Dataset. In our experiments, we constructed training data from Musique (Trivedi et al., 2022), a challenging multi-hop QA dataset that requires models to identify and connect multiple pieces of evidence. This makes it well suited for constructing and refining training-time scaffolds that capture evidence organization. Using FlashRAG (Jin et al., 2025), we used Wikipedia as the document source and bge-large-en-v1.5 (Xiao et al., 2024) as the retrieval engine.

Using the process described in Section 3.2, we generated 2,401 training pairs from the 19,938 Musique entries. Appendix A.9 reports the full filtering funnel and the reasons for discarding examples. Musique is used for the main training pipeline, and its development set serves as the in-domain test set. To examine whether data construction depends on this source dataset, we also build KARE training data from Natural Questions and HotpotQA, with results reported in Appendix A.14. To evaluate out-of-distribution (OOD) performance, we incorporate Natural Questions (Kwiatkowski et al., 2019) and PopQA (Mallen et al., 2022) for single-hop retrieval, HotpotQA (Yang et al., 2018) for multi-hop reasoning, TruthfulQA (Lin et al., 2021) for truthfulness-oriented QA under generation-style answer matching, and Zero-shot RE (Levy et al., 2017) for slot-filling. We also include domain-shift

Method	In Domain		Out Of Domain									
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
Vanilla RAG	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
Realm	6.08	13.46	29.02	39.21	24.94	35.18	31.20	36.73	2.63	10.39	40.23	39.52
RetRobust	5.25	12.62	28.99	39.06	24.27	34.08	32.52	36.75	3.53	10.96	37.97	36.90
RAFT	6.04	15.07	27.92	41.63	24.96	38.71	35.82	44.39	9.25	24.49	<u>57.95</u>	<u>54.96</u>
RPO	6.37	13.34	28.03	40.84	28.25	40.21	32.87	42.69	4.15	15.27	53.50	49.52
RA-DiT	8.11	<u>17.83</u>	33.08	45.66	27.17	39.33	36.36	43.78	5.69	14.70	45.16	44.04
DDR	7.82	19.18	34.78	48.85	30.58	42.59	<u>40.63</u>	<u>45.90</u>	1.68	8.41	55.34	49.12
KARE	<u>8.02</u>	15.75	37.86	50.84	32.36	44.29	40.88	47.77	<u>7.42</u>	<u>17.76</u>	59.52	58.00

Table 1: Overall performance on Llama-3.1-8B-Instruct. The **best** result in each column is highlighted in bold, and the second-best result is underlined.

tests on RAGBench(Friel et al., 2024) domains covering legal, finance, and biomedical QA, with details reported in Appendix A.11. This diverse setup supports a systematic analysis of generalization across task formats and domain settings.

Evaluation. Following FlashRAG(Jin et al., 2025), we employ exact match (EM) and F1 score as evaluation metrics for question answering. For TruthfulQA generation evaluation, we use BLEU and ROUGE-1 scores to measure answer matching. For zero-shot relation extraction, we use F1 score and precision.

Baselines. We compare against baselines that cover the main alternatives to scaffolded alignment, without imposing a strict taxonomy. Vanilla RAG directly conditions the backbone model on retrieved documents without parameter updates. RA-DiT(Lin et al., 2024) and RAFT(Zhang et al., 2024) represent retrieval-conditioned supervised adaptation. RA-DiT instruction-tunes the model to use retrieved information, while RAFT fine-tunes on question-document-answer examples with relevant and distractor contexts for domain-specific RAG. DDR(Li et al., 2024) and RPO(Yan et al., 2025) represent preference or reward-based RAG optimization. DDR uses final-task rewards to align RAG modules, and RPO optimizes preferences tied to retrieval relevance and knowledge conflicts. We also include retrieval-context construction baselines. The REALM-style oracle-context variant(Guu et al., 2020) trains with answer-supporting evidence, while RetRobust(Yoran et al., 2023) trains with mixtures of relevant and irrelevant contexts. Since several original methods update retrievers or assume different RAG modules, all baselines are adapted to our fixed-retriever, generator-only setting using the recon-

structed dataset. The comparison concerns generator adaptation under shared inputs and training resources, not exact reproduction of each published system. Details are provided in Appendix A.3.

Implementation Details. We use Llama-3.1-8B-Instruct(Grattafiori et al., 2024) as the main backbone. We also evaluate Llama-3.2-3B-Instruct and Qwen2.5-14B-Instruct(Yang et al., 2024), with results reported in Appendix A.6. All methods retrieve five documents per query using the fixed retriever described above. Training is implemented with TRL(von Werra et al., 2020) and LoRA(Hu et al., 2022). We train for one epoch with a learning rate of $5e-5$. For preference optimization, we set $\beta = 0.1$ and use $\alpha = 0.1$ and $\gamma = 1.1$ in DDPO. All KARE scores are averaged over three runs. We use GPT-4o-mini as the default expert model for scaffold refinement. Scaffold construction is a one-time preprocessing step. With GPT-4o-mini, generating 2,401 valid pairs costs about \$20 and takes 8–10 hours. Code and prompts for data construction, training, and evaluation will be released at <https://github.com/thunlp/KARE-RAG>.

5 Evaluation Results

We compare KARE with the baseline models and then examine the effects of the scaffold and optimization design.

5.1 Main Results

Table 1 shows that KARE obtains the best results on NQ, HotpotQA, PopQA, and zero-shot relation extraction. These datasets cover single-hop QA, multi-hop QA, and relation extraction, so the gains are not confined to one task format. On TruthfulQA, KARE obtains the second-best BLEU and ROUGE-1 scores after RAFT.

Format	In Domain		Out Of Domain									
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
None (Vanilla RAG)	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
Summary	5.67	14.20	32.77	47.11	24.43	37.44	39.22	46.72	7.75	21.81	51.71	47.47
Keypoints	<u>7.36</u>	16.31	<u>34.73</u>	<u>49.06</u>	<u>30.07</u>	<u>42.93</u>	41.24	48.38	6.75	<u>19.46</u>	<u>55.01</u>	<u>51.74</u>
Graph (Ours)	8.02	<u>15.75</u>	37.86	50.84	32.36	44.29	<u>40.88</u>	<u>47.77</u>	<u>7.42</u>	17.76	59.52	58.00

Table 2: Ablation study on different training scaffold formats. We compare the impact of structuring retrieved information as Summaries, Keypoints, or Graphs (Ours) against the Vanilla RAG baseline. The **best** result in each column is highlighted in bold, and the second-best result is underlined.

Musique, the in-domain dataset, shows a different pattern. RA-DiT obtains the best EM and DDR the best F1, while KARE performs better on most cross-dataset metrics. The results indicate a trade-off between fitting the training distribution and transferring to other datasets.

Appendix A.4 reports controls that keep the training instances, expert access, filtering, backbone, and optimization settings fixed while replacing scaffold supervision with answer-level or free-form rationale targets. KARE performs better than these controls on all reported metrics.

Section 5.4 examines whether scaffolded alignment can be combined with another RAG training objective.

5.2 Scaffold Format Matters

To investigate how scaffold representation affects training, we compare three formats with different degrees of local structure. Summary consolidates retrieved information into a paragraph. Keypoints list salient evidence units line by line. Graphs represent entities and relations in a more explicit structure. All experiments use Llama-3.1-8B-Instruct, and the prompts are detailed in Appendix A.16.

Table 2 shows that KARE is not tied to a single scaffold format. Keypoints give the best PopQA scores, while graph-structured scaffolds give the best results on NQ, HotpotQA, and zero-shot RE and remain competitive elsewhere. We use the graph-structured scaffold as the default because it performs well on tasks that require multi-hop evidence or fine-grained relations. The framework itself does not require a graph scaffold.

5.3 Optimization Ablation

Table 3 isolates two parts of the optimization objective: token-level weighting in DDPO and the auxiliary SFT regularizer. It reports the same point estimates as the main experiments. Appendix A.5

gives the mean and standard deviation over three independent runs for the KARE variants.

The SFT baseline trains on the refined scaffolds from the same small, filtered set rather than on final answers. All variants use the direct-answer RAG prompt at test time. Its lower scores may arise from the lack of rejected-scaffold supervision, the mismatch between scaffold training and direct-answer inference, or both. This comparison does not show that SFT is generally harmful.

Removing token weighting slightly lowers most metrics, although Musique F1 is unchanged and TruthfulQA BLEU improves. Chosen and rejected scaffolds often differ in only a few factual spans, so a uniform preference loss also assigns weight to the many tokens shared by the pair. DDPO instead gives more weight to the edited spans. The three-run results in Appendix A.5 show the same broad pattern, with lower variation for the full objective on most metrics.

Removing the SFT term lowers most scores and increases run-to-run variation. The training curves in Appendix A.7 show that the DPO margin grows while the implicit rewards for both chosen and rejected responses fall, which is consistent with policy drift. The SFT term keeps the policy closer to the refined scaffold distribution and reduces this drift.

5.4 Exploring Hybrid Training

KARE can be combined with other RAG training objectives. We test this by starting from a DDR-trained model and applying KARE in a second LoRA fine-tuning stage without changing inference. Table 4 reports the resulting performance. This experiment shows that the two training objectives can be used together.

Optimization Method	In Domain				Out Of Domain							
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
None (Vanilla RAG)	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
SFT	4.88	10.46	33.62	44.10	23.69	32.77	35.88	40.73	3.12	9.63	47.50	45.05
KARE (DDPO)	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00
- w/o SFT Loss	6.54	13.92	33.08	46.74	27.51	40.43	37.18	44.34	6.34	18.54	54.36	51.92
- w/o Token Weight	7.73	15.75	37.40	50.57	32.26	43.83	40.27	47.39	7.64	17.00	58.72	57.10

Table 3: Ablation study on optimization approaches, including KARE (DDPO), SFT, and KARE variants without the SFT loss or token weighting. “None (Vanilla RAG)” denotes the baseline without training.

Method	In Domain				Out Of Domain							
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
Vanilla RAG	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
DDR	7.82	<u>19.18</u>	34.78	48.85	30.58	42.59	<u>40.63</u>	45.90	1.68	8.41	55.34	49.12
KARE	<u>8.02</u>	15.75	<u>37.86</u>	<u>50.84</u>	<u>32.36</u>	<u>44.29</u>	40.88	<u>47.77</u>	7.42	17.76	<u>59.52</u>	<u>58.00</u>
DDR + KARE	9.01	20.88	38.49	51.48	33.06	45.13	40.53	48.30	2.30	10.85	60.67	59.74

Table 4: Performance comparison of the hybrid training approach (DDR + KARE) against baselines on Llama-3.1-8B-Instruct. "DDR + KARE" denotes the model initialized with DDR weights and further fine-tuned using the KARE method. The **best** result in each column is highlighted in bold, and the second-best result is underlined.

5.5 Expert Model Diagnostics

We compare Qwen2.5-14B-Instruct, GPT-4o-mini, and GPT-4o as refinement experts in the same data construction pipeline. Qwen2.5-14B-Instruct produces 1,043 valid pairs, while GPT-4o-mini and GPT-4o each produce around 2,400. We also sample 1,043 pairs from the GPT-4o-mini data to control for the number of accepted pairs and average the downstream results over three training runs. Table 5 shows similar results for the matched Qwen2.5-14B and GPT-4o-mini sets on most tasks. Training on all GPT-4o-mini pairs gives better overall performance, which suggests that pair count matters more than expert identity in this comparison. GPT-4o and the full GPT-4o-mini setting give similar results, so we use GPT-4o-mini as the default expert because it costs less.

These results also reflect strict filtering in our data pipeline. A pair is retained only when the retrieved documents support the answer and the refined scaffold leads the generator to the correct output. This criterion reduces data volume, but keeps the final supervision concentrated on high-quality instances. Appendix A.9 reports the complete filtering funnel, failure breakdown, manual audit, and corruption analysis.

5.6 Generality Across Retrievers, Domains, and Pipelines

We test the same trained generator under different retrievers, domains, and inference pipelines.

KARE is trained with bge-large-en-v1.5 retrieval. At inference, we replace this retriever with e5-base-v2 or BM25 without retraining the generator. KARE improves over the no-training baseline with both alternative retrievers, although the absolute scores vary with retriever quality. Full results are reported in Appendix A.10.

We also evaluate domain shift on legal, finance, and biomedical tasks from RAGBench. Table 6 shows that KARE obtains the best result on all six adherence and completeness metrics and on eight of the nine metrics overall. RAFT has a higher CUAD F1. Appendix A.11 details the setup.

KARE also improves over the no-training baselines under Vanilla RAG, CoT, CoN, StructRAG, and IRCOT. These pipelines modify prompting or reasoning, while KARE modifies the generator through training, so the two can be used together. Full results are reported in Appendix A.12.

Appendix A.13 separately compares direct answering with explicit scaffold generation, LLMLingua-2 context compression, and controlled Self-RAG adaptations. These settings modify the inference procedure and are therefore kept separate

Refinement Expert	In Domain				Out Of Domain							
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
None (Vanilla RAG)	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
Qwen2.5-14B-Instruct	6.95	15.16	36.35	49.17	31.48	43.41	38.61	45.92	7.35	18.05	55.16	53.07
GPT-4o-mini (sampled)	7.36	15.23	36.13	49.21	31.49	43.58	38.28	46.13	7.27	17.45	56.12	54.16
GPT-4o-mini	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00
GPT-4o	7.82	15.98	37.70	50.51	32.75	44.36	40.37	47.44	7.59	17.32	59.73	58.42

Table 5: Impact of different expert models on KARE performance. Qwen2.5-14B-Instruct and the sampled GPT-4o-mini setting each use 1,043 accepted pairs; the full GPT-4o-mini setting uses 2,401 pairs. The sampled row is averaged over three training runs. All experiments use Llama-3.1-8B-Instruct as the generator.

Method	CUAD (Legal)			FinQA (Finance)			PubMedQA (Biomedical)		
	F1	Adh.	Comp.	F1	Adh.	Comp.	F1	Adh.	Comp.
Vanilla RAG	9.01	55.88	20.18	16.98	55.27	30.89	11.40	85.92	37.16
DDR	10.70	38.04	21.60	7.80	35.74	25.81	8.15	62.41	23.22
RAFT	12.19	45.49	17.98	6.75	33.13	28.14	17.86	77.96	44.80
KARE	11.56	56.41	22.92	19.62	56.09	32.57	18.26	86.20	46.37

Table 6: Domain-shift evaluation on three RAGBench domains. Adh. and Comp. denote adherence and completeness judged by GPT-4.1-mini. All methods use the same fixed-retriever, generator-only protocol.

Train Method	MMLU (Acc)	MMLU-Pro (Acc)
None	68.00	43.97
KARE	67.90	44.60

Table 7: General ability evaluation on MMLU and MMLU-Pro datasets using Llama-3.1-8B-Instruct.

from the main generator-adaptation comparison.

5.7 General Ability Diagnostics

Table 7 checks whether KARE affects general ability. Accuracy changes from 68.00 to 67.90 on MMLU(Hendrycks et al., 2021) and from 43.97 to 44.60 on MMLU-Pro(Wang et al., 2024). These small changes provide no evidence of a substantial loss of general ability on either benchmark.

6 Conclusion

This paper presents KARE-RAG, a structured-supervision framework for RAG. KARE builds localized preference pairs from temporary scaffolds, teaching valid evidence organization under retrieval noise. It adopts DDPO to learn from these localized corrections and improves transfer across the evaluated QA and relation extraction datasets while preserving standard RAG inference without extra runtime cost. Its scaffold-based supervision can also complement existing RAG training objectives as an additional alignment stage. These findings

suggest that fine-grained scaffolded supervision is promising for reliable knowledge-intensive generation.

7 Limitations

- **Retrieval coverage.** KARE improves how the generator uses retrieved evidence, but it still assumes sufficient retrieval coverage. If the retrieved documents do not contain answer-supporting evidence, the refinement stage cannot recover information that is absent from the context.
- **Scaffold design.** The scaffold design space remains only partially explored. We study summaries, keypoints, and graph-structured sketches, so the conclusions may not cover all possible scaffold formats or task-specific decomposition strategies.
- **Evaluation scope.** Our evaluation mainly focuses on English factual RAG. The results may not directly transfer to multilingual RAG or open-ended dialogue, where evidence ambiguity, user intent, and interaction history can be substantially different.
- **Expert-assisted data construction.** KARE relies on an expert model to refine scaffolds and filter valid pairs. Although this is a one-time preprocessing cost and does not affect

inference latency, the quality and volume of supervision can depend on expert capability and the scaffold length limit. Strict filtering also concentrates training on cases that can be corrected from the retrieved evidence.

Acknowledgments

This work was supported by the National Key Research and Development Program of China under Grant No. 2025ZD0215701, the National Natural Science Foundation of China under Grant Nos. T2341003 and U2336214, and the Beijing Natural Science Foundation under Grant No. L257019. We also thank the AI9Stars community for its support.

References

- Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. [Self-rag: Learning to retrieve, generate, and critique through self-reflection](#). In *The Twelfth International Conference on Learning Representations*.
- Jinyuan Fang, Zaiqiao Meng, and Craig MacDonald. 2024. [TRACE the evidence: Constructing knowledge-grounded reasoning chains for retrieval-augmented generation](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 8472–8494, Miami, Florida, USA. Association for Computational Linguistics.
- Robert Friel, Masha Belyi, and Atindriyo Sanyal. 2024. Ragbench: Explainable benchmark for retrieval-augmented generation systems. *arXiv preprint arXiv:2407.11005*.
- Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. [Retrieval-augmented generation for large language models: A survey](#). *arXiv preprint arXiv:2312.10997*.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.
- Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.
- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. [Measuring massive multitask language understanding](#). In *International Conference on Learning Representations*.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. [Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complexity](#). In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050, Mexico City, Mexico. Association for Computational Linguistics.
- Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*.
- Jiajie Jin, Yutao Zhu, Zhicheng Dou, Guanting Dong, Xinyu Yang, Chenghao Zhang, Tong Zhao, Zhao Yang, and Ji-Rong Wen. 2025. [FlashRAG: A modular toolkit for efficient retrieval-augmented generation research](#). In *Companion Proceedings of the ACM on Web Conference 2025*, pages 737–740. Association for Computing Machinery.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.
- Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. [Natural questions: a benchmark for question answering research](#). *Transactions of the Association for Computational Linguistics*, 7:452–466.
- Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xian-gru Peng, and Jiaya Jia. 2024. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms. *arXiv preprint arXiv:2406.18629*.
- Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. [Zero-shot relation extraction via reading comprehension](#). In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020.

- Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474.
- Xinze Li, Sen Mei, Zhenghao Liu, Yukun Yan, Shuo Wang, Shi Yu, Zheni Zeng, Hao Chen, Ge Yu, Zhiyuan Liu, Maosong Sun, and Chenyan Xiong. 2024. Rag-ddr: Optimizing retrieval-augmented generation using differentiable data rewards. *arXiv preprint arXiv:2410.13509*.
- Zhuoqun Li, Xuanang Chen, Haiyang Yu, Hongyu Lin, Yaojie Lu, Qiaoyu Tang, Fei Huang, Xianpei Han, Le Sun, and Yongbin Li. 2025. [StructRAG: Boosting knowledge intensive reasoning of LLMs via inference-time hybrid information structurization](#). In *The Thirteenth International Conference on Learning Representations*.
- Stephanie Lin, Jacob Hilton, and Owain Evans. 2021. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*.
- Xi Victoria Lin, Xilun Chen, Mingda Chen, Weijia Shi, Maria Lomeli, Richard James, Pedro Rodriguez, Jacob Kahn, Gergely Szilvassy, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. [RA-DIT: Retrieval-augmented dual instruction tuning](#). In *The Twelfth International Conference on Learning Representations*. OpenReview.net.
- Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024a. [Lost in the middle: How language models use long contexts](#). *Transactions of the Association for Computational Linguistics*, 12:157–173.
- Zhihan Liu, Miao Lu, Shenao Zhang, Boyi Liu, Hongyi Guo, Yingxiang Yang, Jose Blanchet, and Zhaoran Wang. 2024b. Provably mitigating overoptimization in rlhf: Your sft loss is implicitly an adversarial regularizer. *arXiv preprint arXiv:2405.16436*.
- Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. [Entity-based knowledge conflicts in question answering](#). In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7052–7063, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.
- Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. [Query rewriting in retrieval-augmented large language models](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.
- Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.
- Shengyu Mao, Yong Jiang, Boli Chen, Xiao Li, Peng Wang, Xinyu Wang, Pengjun Xie, Fei Huang, Hua-jun Chen, and Ningyu Zhang. 2024. [RaFe: Ranking feedback improves query rewriting for RAG](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 884–901, Miami, Florida, USA. Association for Computational Linguistics.
- Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. [LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 963–981. Association for Computational Linguistics.
- Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. [Direct preference optimization: Your language model is secretly a reward model](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 53728–53741. Curran Associates, Inc.
- Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. [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)*, pages 8371–8384, Mexico City, Mexico. Association for Computational Linguistics.
- Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-end training of multi-document reader and retriever for open-domain question answering. *Advances in Neural Information Processing Systems*, 34:25968–25981.
- 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.
- Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. [Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.
- Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Galouédec. 2020. [TRL: Transformers Reinforcement Learning](#).
- Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

- Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. 2024. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *arXiv preprint arXiv:2406.01574*.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. 2022. [Chain-of-thought prompting elicits reasoning in large language models](#). *Advances in neural information processing systems*, 35:24824–24837.
- Siye Wu, Jian Xie, Jiangjie Chen, Tinghui Zhu, Kai Zhang, and Yanghua Xiao. 2024. [How easily do irrelevant inputs skew the responses of large language models?](#) In *First Conference on Language Modeling*.
- Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muenighoff, Defu Lian, and Jian-Yun Nie. 2024. [C-pack: Packed resources for general chinese embeddings](#). In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '24*, pages 641–649, New York, NY, USA. Association for Computing Machinery.
- Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Re-comp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.
- Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2024a. Search-in-the-chain: Interactively enhancing large language models with search for knowledge-intensive tasks. In *Proceedings of the ACM on Web Conference 2024*, pages 1362–1373.
- Zhipeng Xu, Zhenghao Liu, Yibin Liu, Chenyan Xiong, Yukun Yan, Shuo Wang, Shi Yu, Zhiyuan Liu, and Ge Yu. 2024b. [ActiveRAG: Revealing the treasures of knowledge via active learning](#). *arXiv preprint arXiv:2402.13547*.
- Shi-Qi Yan, Quan Liu, and Zhen-Hua Ling. 2025. [RPO: Retrieval preference optimization for robust retrieval-augmented generation](#). In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5228–5240, Vienna, Austria. Association for Computational Linguistics.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. [HotpotQA: A dataset for diverse, explainable multi-hop question answering](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.
- Ori Yoran, Tomer Wolfson, Ori Ram, and Jonathan Berant. 2023. Making retrieval-augmented language models robust to irrelevant context. *arXiv preprint arXiv:2310.01558*.
- Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwen He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, and Tat-Seng Chua. 2024a. Rllm-v: Towards trustworthy mllms via behavior alignment from fine-grained correctional human feedback. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13807–13816.
- Wenhao Yu, Hongming Zhang, Xiaoman Pan, Peixin Cao, Kaixin Ma, Jian Li, Hongwei Wang, and Dong Yu. 2024b. [Chain-of-note: Enhancing robustness in retrieval-augmented language models](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 14672–14685, Miami, Florida, USA. Association for Computational Linguistics.
- Tianjun Zhang, Shishir G. Patil, Naman Jain, Sheng Shen, Matei Zaharia, Ion Stoica, and Joseph E. Gonzalez. 2024. [RAFT: Adapting language model to domain specific RAG](#). In *First Conference on Language Modeling*.

A Appendix

A.1 LLM Employment

In the preparation of this paper, we employed large language models (LLMs) exclusively to assist with language-related aspects of the writing process. Specifically, LLMs were used to refine grammatical structures, facilitate translation of certain text segments, and provide final polishing to improve the clarity and readability of the manuscript. No substantial intellectual, methodological, or analytical input was contributed by LLMs, and all scientific content, results, interpretations, and conclusions remain entirely our own.

A.2 Scaffold Refinement Algorithm

This appendix expands on the filtering and editing process summarized in Section 3.2. For each query-document set, the base generator first produces a scaffold s^- through Knowledge-Aware Sampling, and the same generator uses this scaffold to answer the query. We retain only examples in which the generated answer is incorrect, because these cases expose a concrete scaffold-level failure that can be

Algorithm 1 Scaffold Refinement

```
1: Generate initial scaffold  $s^-$  using  $\text{LLM}_{\text{Gen}}$ 
   with Knowledge-Aware Sampling
2: if  $s^-$  produces incorrect answer  $y_{\text{err}} \neq y_{\text{gnd}}$ 
   then
3:   Check document adequacy with
      $\text{LLM}_{\text{Exp}}(q \oplus D \oplus y_{\text{gnd}})$ 
4:   if Document  $D$  is sufficient then
5:     Revise the scaffold with an advanced
       model  $\text{LLM}_{\text{Exp}}$ 
       
$$s^+ \leftarrow \text{LLM}_{\text{Exp}}(\text{Instruct}_{\text{revise}},$$

       
$$q \oplus D \oplus y_{\text{gnd}} \oplus s^-)$$

6:   Generate final answer  $y_{\text{Gen}}$  using  $s^+$  with
      $\text{LLM}_{\text{Gen}}$ 
7:   while  $y_{\text{Gen}} \neq y_{\text{gnd}} \ \& \ \text{iter} < \text{max\_iter}$  do
8:     Further revise  $s^+$  using error analysis
9:     Generate  $y_{\text{Gen}}$  with  $\text{LLM}_{\text{Gen}}$ 
10:  end while
11:  if  $y_{\text{Gen}} = y_{\text{gnd}}$  then
12:    Add  $(s^+, s^-)$  as a contrastive DDPO
      training sample pair
13:  else
14:    Discard the sample
15:  end if
16: end if
17: end if
```

corrected. We then ask the expert model to judge whether the retrieved documents D contain enough evidence to support the gold answer y_{gnd} . If the evidence is insufficient, the example is discarded rather than repaired, since any correction would require information outside the retrieved documents.

For evidence-sufficient cases, LLM_{Exp} receives the query, retrieved documents, gold answer, and flawed scaffold. It is instructed to minimally revise s^- while preserving the scaffold format and most of the surrounding structure. The revision should only add, remove, or modify content supported by the retrieved documents. The refined scaffold s^+ is then passed back to LLM_{Gen} for answer generation. We accept (s^+, s^-) only when s^+ leads to the correct answer, which keeps the preference pair grounded in the retrieved evidence and aligned with the factual error that caused the original failure.

A.3 Additional Baseline Settings

The baseline results are controlled adaptations rather than exact reproductions of each complete

system. We use official code or prompts when available and otherwise follow the procedure described in the paper. We adapt each method to the same Llama backbone, retrieved documents, top-five retrieval, generator-only optimization, and training budget. The published results are not directly comparable because the original systems may use different backbones, retrievers, corpora, or inference modules.

We construct the RA-DiT(Lin et al., 2024) and DDR(Li et al., 2024) training data from Musique. RA-DiT uses the instruction prompts from the paper and five retrieved documents per query. For DDR, we sample several responses at different temperatures from the same retrieved context. We label a response positive if either F1 or EM exceeds its threshold and negative only if both metrics fall below their thresholds. We discard questions without both labels, then pair the highest-scoring positive response with the lowest-scoring negative response for DPO training.

The REALM-style oracle-context variant and RetRobust share the same backbone and training setup: Llama-3.1-8B-Instruct, the fixed bge-large-v1.5 retriever, and five retrieved documents per question. Only the generator is updated. We train each model with TRL(von Werra et al., 2020) and LoRA(Hu et al., 2022) for one epoch at a learning rate of 5×10^{-5} .

RAFT(Zhang et al., 2024) is trained on MuSiQue examples containing answer-supporting and distractor documents, with targets that include a short rationale followed by the answer. RPO(Yan et al., 2025) forms preference pairs from relevant and irrelevant responses under the same retrieved contexts and uses $\beta = 0.1$. For the REALM-style oracle-context baseline(Guu et al., 2020), we follow the official implementation’s data-construction procedure to build examples with answer-supporting evidence, then fine-tune the generator under the shared configuration above. RetRobust(Yoran et al., 2023) follows its official data-construction procedure by mixing relevant and irrelevant documents, while retaining the same generator-only training setup. These results should be read as controlled adaptations under a common experimental setting, not as exact reproductions of the original complete systems.

A.4 Matched-Budget Supervision Controls

We compare KARE with three alternatives using the same 2,401 instances, retrieved contexts, GPT-

4o-mini access, gold-answer verification and filtering, backbone, and training configuration. Answer-DPO uses expert-corrected final answers without scaffolds. Rationale-DPO uses expert-generated free-form rationales and answers without a structured scaffold. KARE-Answer-Only follows the KARE refinement pipeline but optimizes only the final-answer pairs derived from the refined and flawed scaffolds. Full KARE optimizes the scaffold preference pairs.

As shown in Table 8, Full KARE performs better than the three answer-level controls on all reported metrics under these matched conditions.

A.5 Optimization Variance on the Main Backbone

Table 9 reports mean and standard deviation over three independent runs for the KARE variants on Llama-3.1-8B-Instruct. These statistics supplement the point estimates in Table 3. Given the small number of runs, we treat the comparison as descriptive and do not make a statistical significance claim.

A.6 Supplementary Experiments Across Model Sizes

This section reports supplementary experiments on Llama-3.2-3B-Instruct and Qwen2.5-14B-Instruct. We include both the main-comparison results and the optimization ablations to show whether the observations in the main text remain consistent across model sizes.

Table 11 extends the optimization ablation in Table 3 to the same two additional backbones. The results follow the same broad pattern as the results for the 8B model. DDPO generally improves over SFT, while removing token weighting or the SFT term weakens the overall profile. These trends suggest that the optimization design is not specific to the main backbone.

A.7 Training Curves

Figure 2 compares the full DDPO objective with the variant that removes the SFT term. Without SFT regularization, the DPO margin increases, but the implicit rewards for both chosen and rejected responses decrease. This indicates that the model can improve relative preference separation while drifting away from the refined scaffold distribution. With the SFT term, the margin grows more moderately and the reward for chosen responses remains better anchored. These trends support the main-text

observation that SFT regularization helps stabilize preference optimization.

A.8 Computational Efficiency

Data generation uses GPT-4o-mini for targeted refinement. Filtering and refining 2,401 valid training pairs from the original 19,938 entries costs about \$20 and takes 8–10 hours, depending on API latency. This is a one-time preprocessing cost and does not affect later model use.

We train parameter-efficient LoRA adapters with DeepSpeed-Zero3 on a single A800-40G GPU. The objective introduces no additional computational complexity over other DPO-based methods such as DDR. Training takes less than one hour for 3B-parameter models and less than four hours for 14B-parameter models. Scaffolds are used only for training supervision, so inference has the same latency and computational cost as Vanilla RAG.

A.9 Scaffold Data Quality Analysis

Table 12 traces the 19,938 Musique training examples through the construction pipeline. KARE retains cases in which the retrieved documents contain enough evidence but the base generator answers incorrectly. Gold answers are used only during offline construction to check evidence adequacy and refinement success, and are not available during inference. Most examples are removed because the retrieved documents do not contain enough evidence. The final set contains 2,401 preference pairs.

Stage	Count	Ratio
Original examples	19,938	100.0%
Base generator correct	3,735	18.7%
Insufficient retrieved evidence	13,216	66.3%
Refinement/verification failure	586	2.9%
Final preference pairs	2,401	12.0%

Table 12: Filtering statistics for the default Musique and GPT-4o-mini data-construction pipeline. Ratios are relative to the original training set.

Of the 586 refinement or verification failures, 573 result from truncation at the 256-token scaffold limit, 7 are formatting errors, 4 contain an insufficient evidence chain, and 2 still lead to an incorrect answer after refinement. We tested a 512-token limit, but it yielded few additional valid pairs while increasing data-construction and training costs. We therefore retain the 256-token limit. This breakdown is specific to the default Musique and GPT-

Method	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	R-1	F1	Prec.
Vanilla RAG	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
Answer-DPO	6.79	15.24	30.33	44.84	27.59	40.39	33.97	44.12	3.13	11.42	51.13	48.18
Rationale-DPO	7.11	15.18	35.50	48.45	29.10	40.44	37.32	45.77	3.99	9.81	53.05	52.89
KARE-Answer-Only	6.74	14.00	30.30	44.41	27.08	39.93	33.83	43.77	3.15	11.46	50.68	47.60
Full KARE	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00

Table 8: Matched-budget supervision controls. All trained variants use the same instances, retrieved contexts, expert access, verification and filtering, backbone, and training configuration.

Optimization Method	In Domain				Out Of Domain							
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
KARE (DDPO)	8.02±0.23	15.75±0.35	37.86±0.48	50.84±0.61	32.36±0.24	44.29±0.24	40.88±0.33	47.77±0.20	7.42±0.69	17.76±2.59	59.52±0.50	58.00±0.68
- w/o SFT Loss	6.54±3.60	13.92±4.25	33.08±12.74	46.74±12.00	27.51±10.45	40.43±9.77	37.18±10.91	44.34±9.01	6.34±0.99	18.54±2.71	54.36±12.23	51.92±14.71
- w/o Token Weight	7.73±0.35	15.75±0.15	37.40±1.68	50.57±1.55	32.26±1.33	43.83±0.99	40.27±0.64	47.39±0.41	7.64±0.74	17.00±3.58	58.72±1.25	57.10±1.55

Table 9: Optimization variance on Llama-3.1-8B-Instruct. Values are mean $\pm$ standard deviation over three independent runs.

4o-mini pipeline and may differ for other expert models or source datasets.

We manually inspect 100 sampled refined instances. A scaffold is valid if its corrected evidence structure is sufficient to derive the gold answer. Of the 100 samples, 97 are fully valid, 2 are partially valid, and 1 is invalid. The two partially valid samples capture the main entity or reasoning chain but lack the detail needed for the final answer. We also check whether the retrieved documents fully support each chosen scaffold and find two unsupported cases among the 97 initially valid samples. Under this stricter criterion, 95 of the 100 sampled instances remain fully valid.

We next evaluate whether data quality affects training. Starting from the refined training set, we randomly corrupt 15% and 30% of the scaffolds by deleting answer-supporting edges, replacing edges or relations, and adding distractor evidence.

Table 13 shows that the clean KARE setting achieves the strongest results. Performance decreases under 15% corruption and drops further under 30% corruption. Although the corrupted variants still remain above Vanilla RAG, the consistent degradation indicates that scaffold quality materially affects the training signal. These results support the need for a strict refinement and filtering pipeline when constructing KARE training data.

A.10 Transfer Across Retrieval Backends

KARE is trained using data constructed with bge-large-en-v1.5 retrieval. To examine whether the learned generator depends on this retrieval backend, we evaluate the same trained model with e5-

base-v2(Wang et al., 2022) and BM25 at inference time. No additional training is performed for these retrieval settings.

As shown in Table 14, KARE improves over the no-training baseline under all three retrieval backends. The gains persist when the inference retriever is changed to e5-base-v2 or BM25, although the absolute scores depend on retrieval quality. This result suggests that the training signal is not limited to the bge-large-en-v1.5 retrieval distribution. It can also improve evidence use under unseen retrieval backends.

A.11 Domain Shift on RAGBench

We further evaluate domain shift using three RAGBench domains(Friel et al., 2024). CUAD covers legal QA, FinQA covers financial QA, and PubMedQA covers biomedical QA. We use the same KARE-trained Llama-3.1-8B-Instruct model from the main experiments, without domain-specific fine-tuning. We report F1 together with adherence and completeness, where the latter two metrics are judged by GPT-4.1-mini.

Table 6 in the main paper compares KARE with Vanilla RAG, DDR, and RAFT under this protocol. KARE obtains the best result on all six adherence and completeness metrics and on eight of nine metrics overall; RAFT obtains the best CUAD F1. These results are limited to the three evaluated domains and this fixed-retriever, generator-only setup.

Method	In Domain		Out Of Domain									
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
<i>Llama-3.2-3B-Instruct</i>												
Vanilla RAG	4.47	9.89	33.20	45.59	27.89	37.75	34.32	41.60	6.36	13.78	48.70	47.40
Realm	4.59	11.42	29.30	39.37	23.44	33.40	28.41	33.31	2.34	9.36	23.04	22.66
RetRobust	5.42	12.32	29.21	39.53	22.90	32.73	29.18	34.28	3.79	10.16	22.37	21.55
RA-DiT	5.46	13.16	26.25	37.08	21.74	31.74	33.01	37.76	3.06	10.24	32.05	31.82
DDR	6.00	16.18	28.93	45.31	22.88	<u>38.04</u>	33.21	<u>42.91</u>	2.83	10.09	52.55	50.76
KARE	<u>5.59</u>	<u>13.28</u>	33.21	46.56	<u>27.62</u>	39.59	38.29	45.52	7.58	17.75	<u>51.94</u>	<u>49.03</u>
<i>Qwen2.5-14B-Instruct</i>												
Vanilla RAG	6.66	14.58	32.67	47.01	30.64	42.31	36.38	43.98	6.90	18.15	53.26	49.27
Realm	7.94	15.33	<u>34.37</u>	<u>48.08</u>	31.48	43.78	38.14	44.55	1.49	8.50	51.46	<u>51.41</u>
RetRobust	7.41	15.10	33.36	43.62	31.02	41.58	31.83	36.12	1.88	8.81	43.08	42.67
RA-DiT	<u>9.06</u>	<u>18.47</u>	33.36	43.62	31.02	41.58	37.32	44.91	1.59	9.63	<u>54.36</u>	53.91
DDR	9.68	19.84	32.62	45.85	<u>32.05</u>	<u>44.65</u>	41.08	<u>45.87</u>	1.14	8.30	52.08	50.13
KARE	8.32	17.48	36.19	49.96	33.96	46.45	<u>38.58</u>	46.09	8.80	20.39	54.47	50.55

Table 10: Additional main-comparison results on Llama-3.2-3B-Instruct and Qwen2.5-14B-Instruct. The main text focuses on Llama-3.1-8B-Instruct, while these results provide complementary evidence across model scales. The **best** result in each model block is highlighted in bold, and the second-best result is underlined.

Optimization Method	In Domain		Out Of Domain									
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
<i>Llama-3.2-3B-Instruct</i>												
None (Vanilla RAG)	4.47	9.89	33.20	45.59	27.89	37.75	34.32	41.60	6.36	13.78	48.70	47.40
SFT	2.73	7.29	20.06	30.46	21.73	31.35	29.52	35.80	5.96	14.02	25.32	22.83
KARE (DDPO)	5.59	13.28	33.21	46.56	27.62	39.59	38.29	45.52	7.58	17.75	51.94	49.03
- w/o SFT Loss	4.10	13.99	22.32	38.37	23.13	37.50	31.91	41.57	7.96	20.64	49.36	44.64
- w/o Token Weight	4.63	13.97	26.40	41.95	24.27	38.53	33.32	42.87	7.97	19.16	51.20	48.32
<i>Qwen2.5-14B-Instruct</i>												
None (Vanilla RAG)	6.66	14.58	32.67	47.01	30.64	42.31	36.38	43.98	6.90	18.15	53.26	49.27
SFT	6.08	14.12	32.23	45.70	28.32	40.56	36.57	43.86	8.32	20.46	51.72	49.27
KARE (DDPO)	8.32	17.48	36.19	49.96	33.96	46.45	38.58	46.09	8.80	20.39	54.47	50.55
- w/o SFT Loss	6.29	15.70	25.18	41.25	25.18	41.25	33.42	42.03	8.54	21.95	52.10	47.22
- w/o Token Weight	7.57	16.58	34.99	49.00	32.84	45.47	38.12	45.65	9.00	20.17	54.07	50.13

Table 11: Ablation analysis of optimization methods across varying model scales. We compare the SFT baseline, the DDPO method, and its ablation variants (w/o SFT Loss, w/o Token Weighting) on Llama-3.2-3B-Instruct and Qwen2.5-14B-Instruct.

Training Setting	In Domain		Out Of Domain									
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
None (Vanilla RAG)	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
KARE	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00
+ 15% corrupted scaffolds	7.53	15.17	37.15	50.02	32.33	43.72	39.41	46.84	7.09	17.62	58.99	56.68
+ 30% corrupted scaffolds	6.83	14.71	36.53	49.37	31.47	43.27	38.52	46.27	6.83	16.94	58.44	55.19

Table 13: Effect of scaffold data corruption during training. Corruption is applied to a random subset of training scaffolds by deleting answer-supporting edges, replacing edges or relations, and adding distractor evidence.

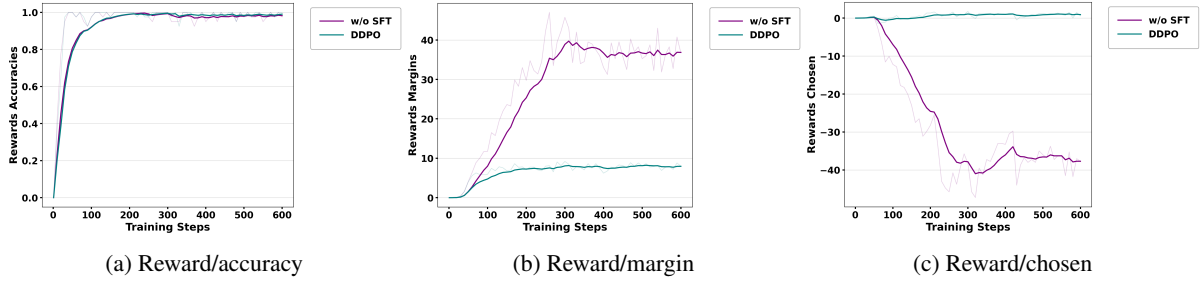


Figure 2: Training curves with or without the SFT loss.

Train Method	In Domain		Out Of Domain									
	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
<i>bge-large-en-v1.5</i>												
None	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
KARE	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00
<i>e5-base-v2</i>												
None	4.85	11.40	32.88	45.90	27.52	39.36	37.31	47.29	4.63	17.20	56.78	53.50
KARE	6.73	15.15	34.02	46.75	29.86	43.49	42.88	50.83	6.58	21.20	61.33	60.97
<i>BM25</i>												
None	2.85	7.56	16.54	23.93	19.81	28.86	15.60	20.04	7.77	22.27	38.18	36.01
KARE	3.73	9.28	18.98	27.86	22.15	31.64	17.99	22.36	11.27	26.42	42.95	43.41

Table 14: Retriever transfer results on Llama-3.1-8B-Instruct. KARE is trained with bge-large-en-v1.5 retrieval and evaluated with bge-large-en-v1.5, e5-base-v2, and BM25 retrieval at inference time.

A.12 Generalization Across Different Generation Pipelines

We evaluate KARE with five generation pipelines: Vanilla RAG, Chain-of-Thought(Wei et al., 2022), Chain-of-Note(Yu et al., 2024b), StructRAG(Li et al., 2025), and IRCOT(Trivedi et al., 2023). Each pipeline is compared with its no-training baseline. IRCOT uses the same task metrics as the other pipelines rather than a separate accuracy-only report.

Table 15 shows that KARE improves every reported metric under each of the five pipelines. The gains therefore do not depend on the Vanilla RAG prompt used in the main comparison.

A.13 Additional Inference-Time Comparisons

The main experiments use one-pass direct-answer inference so that KARE retains the same interface as Vanilla RAG. This section examines three alternatives that change the inference procedure: explicit scaffold generation, context compression, and Self-RAG-style reflection. We report them separately because they do not share the main comparison’s inference protocol.

Explicit scaffold inference. Although KARE is trained on scaffold pairs, it does not generate a scaffold at test time by default. We also evaluate a two-step setting in which the trained model first constructs a scaffold and then answers from it. Table 16 shows that explicit scaffold generation improves MuSiQue and HotpotQA F1 but lowers the other four reported metrics. It also requires separate scaffold and answer generation steps. We therefore retain direct answering as the default and treat explicit scaffold inference as a task-dependent option for compositional QA.

Context compression. We use LLMingua-2 (Pan et al., 2024) to compress the retrieved contexts before generation, with target retained-context ratios of 50% and 30%. Table 17 compares compression with both Vanilla RAG and KARE. Relative to uncompressed Vanilla RAG, LLMingua-2 lowers seven of the twelve metrics at 50% and eleven at 30%. Under the same compressed inputs, KARE exceeds the corresponding Vanilla setting on all twelve metrics at 50% and eleven at 30%. These results do not support context compression as a substitute for generator training in this setting,

Inference Pipeline	Train Method	In Domain		Out Of Domain									
		Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
		EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
Vanilla	None	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
	KARE	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00
CoT	None	8.52	16.40	35.63	48.48	31.11	42.45	37.71	44.35	3.41	10.27	51.26	48.61
	KARE	10.34	19.28	37.22	50.28	34.40	46.82	40.38	46.96	4.51	11.35	54.44	52.41
CoN	None	6.83	14.47	35.06	48.61	29.05	40.70	36.06	43.32	5.81	11.96	45.60	42.80
	KARE	9.18	18.44	37.55	50.73	33.11	45.34	39.36	46.30	7.14	12.83	49.41	48.04
StructRAG	None	3.97	10.30	26.04	37.94	18.27	29.96	30.83	30.59	9.75	25.44	42.43	38.93
	KARE	8.07	16.69	29.14	40.86	21.27	34.13	33.50	39.00	11.32	27.91	45.56	42.44
IRCoT	None	9.97	18.58	34.13	46.63	36.10	49.36	38.42	46.97	4.80	15.81	54.95	51.71
	KARE	13.90	23.14	35.18	48.89	38.24	51.86	41.26	48.85	6.61	20.47	58.66	55.56

Table 15: Generalization across inference pipelines on Llama-3.1-8B-Instruct. We compare no additional training with KARE under Vanilla RAG, Chain-of-Thought (CoT), Chain-of-Note (CoN), StructRAG, and IRCOT.

Inference Setting	MuSiQue F1	NQ F1	HotpotQA F1	PopQA F1	TQA R-1	ZSRE F1
Direct KARE	15.75	50.84	44.29	47.77	17.76	59.52
KARE, Scaffold-to-Answer	20.53	49.36	46.01	47.55	12.92	37.12

Table 16: Direct-answer and explicit Scaffold-to-Answer inference with the same KARE-trained Llama-3.1-8B-Instruct generator. TQA R-1 denotes TruthfulQA ROUGE-1, and ZSRE denotes zero-shot relation extraction.

but show that KARE can be used with compressed contexts.

Self-RAG adaptations. Native Self-RAG (Asai et al., 2024) uses adaptive retrieval and reflection-token decoding, which cannot be fully preserved in our fixed-retriever protocol. We therefore report two controlled adaptations rather than a reproduction of the complete system. Zero-shot Self-RAG scores candidates using prompted relevance, support, and utility labels without training reflection tokens. Self-RAG-LoRA trains the Llama-3.1-8B-Instruct backbone on all 145,533 decontaminated public examples and uses learned reflection-token probabilities for candidate scoring. Table 18 shows that Self-RAG-LoRA obtains the best NQ EM and TruthfulQA ROUGE-1, while KARE obtains the best result on the other ten metrics. These results describe the controlled fixed-retriever setting and should not be read as a comparison with native Self-RAG under its original inference pipeline.

A.14 Impact of Training Dataset Selection

We selected Musique as the primary training dataset because it emphasizes compositional multi-hop reasoning and fragmented evidence. These properties make it well suited for constructing scaffolds that test evidence organization under retrieval noise.

To verify that KARE is not dependent on the

specific distribution of the Musique dataset, we conduct additional experiments using HotpotQA and Natural Questions (NQ) as training sources. This appendix uses Qwen2.5-14B-Instruct as the refinement expert for all source datasets, including the Musique row in Table 19. Under this common-expert setting, we generate 8,500 training pairs from HotpotQA and 9,008 pairs from NQ.

Since the main experiments use GPT-4o-mini as the default expert, the absolute scores in this appendix are not directly comparable to the main GPT-4o-mini setting. The comparison examines whether KARE remains effective with different source datasets under a shared refinement expert. Because the number of accepted pairs differs across datasets, it is not intended to isolate source-dataset effects under a matched data budget. As shown in Table 19, models trained on different source datasets improve over Vanilla RAG across multiple benchmarks, suggesting that KARE is not tied to a single training distribution.

A.15 Case Study of Knowledge Refinement

To provide a qualitative analysis of the refinement mechanism, we present a detailed case study from the Musique dataset in Table 20. In this multi-hop reasoning scenario, the query asks for the creation year of the region containing Hertfordshire, which requires the model to filter out high-interference noise from retrieved documents. As observed in

Method	In Domain		Out Of Domain									
	MuSiQue		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	R-1	F1	Prec.
Vanilla RAG	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
+ LLMingua-2 (50%)	5.71	12.63	31.50	44.26	28.12	40.11	34.84	42.20	6.17	16.24	55.14	53.24
+ LLMingua-2 (30%)	4.51	11.16	30.08	41.88	24.17	34.90	31.31	37.09	6.93	14.90	44.16	43.76
KARE	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00
+ LLMingua-2 (50%)	6.29	14.65	33.35	45.39	29.63	41.96	38.05	44.71	7.65	16.59	59.56	57.63
+ LLMingua-2 (30%)	5.54	13.04	32.69	44.03	26.10	37.67	34.91	40.47	7.70	14.41	49.87	49.13

Table 17: Context-compression results on Llama-3.1-8B-Instruct. Percentages denote the target fraction of the retrieved context retained by LLMingua-2. KARE uses its default direct-answer inference in all rows.

Method	In Domain		Out Of Domain									
	MuSiQue		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	R-1	F1	Prec.
Vanilla RAG	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
Zero-shot Self-RAG	5.83	11.61	33.13	42.79	28.83	38.02	33.12	38.10	5.83	11.46	42.57	42.22
Self-RAG-LoRA (145,533)	1.12	6.13	40.89	48.58	8.85	19.62	2.50	17.42	5.81	25.86	46.25	44.90
KARE	8.02	15.75	37.86	50.84	32.36	44.29	40.88	47.77	7.42	17.76	59.52	58.00

Table 18: Controlled Self-RAG adaptations on Llama-3.1-8B-Instruct with the same fixed retriever used in the KARE experiments. The Self-RAG-LoRA row uses the full decontaminated public training set.

the negative sample (s^-), the base model successfully identifies relevant entities but fails to extract the specific temporal attribute associated with the “East of England” region, instead hallucinating an incorrect answer (“1963”) derived from a distractor document about “Greater London.” The expert refinement pipeline detects this logical gap and explicitly injects the missing evidence (created_in: 1994) into the scaffold. This correction establishes a complete reasoning chain from the entity to the ground truth, demonstrating KARE’s capability to mitigate retrieval noise and enhance factual fidelity

through scaffolded refinement.

A.16 Experiment Prompts

This section presents all prompts used in our experiments. For Vanilla RAG, we use the standard prompt template provided by FlashRAG(Jin et al., 2025) without modification. Table 21 contains the prompts for Knowledge-Aware generation with a Knowledge Graph as the intermediate representation. Tables 22 and 23 contain the prompts for Keypoints and Summary, respectively. Table 24 presents the prompts used during Knowledge Graph refinement.

Training Dataset	Musique		NQ		HotpotQA		PopQA		TruthfulQA		Zero-shot RE	
	EM	F1	EM	F1	EM	F1	EM	F1	BLEU	Rouge-1	F1	Precision
None (Vanilla RAG)	6.00	12.37	34.64	48.30	29.52	40.66	35.43	44.10	5.43	15.03	51.62	49.30
Natural Questions	5.88	12.73	37.36	49.97	30.44	41.20	38.48	45.71	7.42	17.80	59.57	57.79
HotpotQA	6.37	13.80	37.95	50.72	32.13	42.75	39.59	46.36	7.68	15.85	59.27	57.60
Musique	6.95	15.16	36.35	49.17	31.48	43.41	38.61	45.92	7.35	18.05	55.16	53.07

Table 19: Performance comparison of KARE when trained on different datasets (Natural Questions, HotpotQA, and Musique). Llama-3.1-8B-Instruct serves as the generator backbone, while Qwen2.5-14B-Instruct is employed as the expert model for training data refinement.

Case Study: Refining Knowledge Structure for Multi-hop Reasoning	
Query: <i>What year saw the creation of the region where the county of Hertfordshire is located?</i>	
Step 1: Retrieval Context (Selected Snippets) Doc 1 (Gold): "East of England... It was created in 1994 ... It includes the ceremonial counties of ... Hertfordshire ..." Doc 5 (Distractor): "Greater London... was first established... between 1963 and 1986 ..."	
Step 2: Negative Sample (s^-) – Generated by Base Model The model successfully identifies entities but fails to extract the creation year , leading to a reasoning dead-end. • Entity: Hertfordshire (Attributes: county, location: East of England) • Entity: East of England (Attributes: region, population, NUTS classification) [Missing Attribute: Creation Date] • Entity: Greater London (Attributes: established: 1963) ← <i>Distractor Information</i>	
Prediction from s^-: 1963 (Incorrect)	
Step 3: Refined Positive Sample (s^+) – After Expert Refinement The refinement process locates the missing bridge in Doc 1 and structurally inserts the key temporal attribute. • Entity: Hertfordshire (Attributes: county, location: East of England) • Entity: East of England – Attributes: region, population: 5,847,000, created_in: 1994, includes: Hertfordshire	
Prediction from s^+: 1994 (Correct)	

Table 20: A comparison between the initial negative knowledge organization and the refined structure. The refinement explicitly captures the "created_in" attribute that was previously missed, enabling the model to ignore the distractor date (1963) and answer correctly.

Role	Content
Stage 1: Knowledge-Aware Extraction	
System Prompt	You are a helpful AI assistant distinctively skilled at extracting crucial information from documents helpful for answering a given question. For a given question, focus on identifying the key entities, their attributes, and their relationships that are directly relevant to generating an accurate answer. Only include entities and attributes that are crucial for understanding and forming the answer, and avoid unnecessary details. Your response must include the following keys and strictly adhere to the exact structure without any additional text before or after the keys: Entities: - [Entity 1] (Attributes: [Attribute 1, Attribute 2, ...]) - [Entity 2] (Attributes: [Attribute 1, Attribute 2, ...]) ... Relationships: 1. [Entity 1] -> [Relationship] -> [Entity 2] 2. ... Important Note: 1. Do not include any extra commentary or unnecessary details in the response. 2. Closely follow the structure provided above and ensure that the response is concise and directly addresses the question. 3. Don't provide the answer to the question. Instead, focus on extracting the key entities, attributes, and relationships that are essential for answering the question accurately.
User Prompt	Question: {question} Documents: {reference} Knowledge Graph:
Stage 2: Chain-of-Thought Reasoning	
System Prompt	You are a helpful AI assistant good at reasoning on the knowledge graph to answer a given question. For a given question and relevant knowledge graph, provide the step-by-step reasoning process to derive the answer. Make sure the reasoning steps are logical, coherent, and directly related to the question and the information in the knowledge graph.
User Prompt	Question: {question} Knowledge Graph: {s} Reasoning Steps:
Stage 3: Final Generation	
System Prompt	You are a helpful AI assistant good at generating final answers based on the reasoning steps provided for a given question. Important Notes: 1. Make sure the final answer is accurate, concise, and directly addresses the question. 2. Only give me the answer and do not output any other words.
User Prompt	Question: {question} Reasoning Steps: {y_{CoT}} Answer:

Table 21: Detailed Prompts for Knowledge-Aware Generation (Graph Format).

Role	Content
Stage 1: Keypoints Extraction	
System Prompt	You are a helpful AI assistant good at extracting crucial information from documents which are helpful for answering a given question. For a given question, please thoroughly analyze the list of documents (given as strings) and extract the most relevant key points that directly answer the question. Ensure that each key point is directly supported by evidence found within the documents, and avoid unnecessary details. Your response must include the following keys and strictly adhere to the exact structure without any additional text before or after the keys: Key Points: 1. [Key Point 1] 2. [Key Point 2] ... Important Note: 1. Do not include any extra commentary or unnecessary details in the response. 2. Closely follow the structure provided above and ensure that the response is concise and directly addresses the question. 3. Don't provide the answer to the question. Instead, focus on extracting the key points that are essential for answering the question accurately.
User Prompt	Question: {question} Documents: {reference} KeyPoints:
Stage 2: Chain-of-Thought Reasoning	
System Prompt	You are a helpful AI assistant that is good at doing reasoning on the keypoints to answer a given question. For a given question and relevant keypoints, provide the step-by-step reasoning process to derive the answer. Make sure the reasoning steps are logical, coherent, and directly related to the question and the information in the keypoints.
User Prompt	Question: {question} Keypoints: {y_{Key}} Reasoning Steps:
Stage 3: Final Generation	
System Prompt	You are a helpful AI assistant that is good at generating final answers based on the reasoning steps provided for a given question. Important Notes: 1. Make sure the final answer is accurate, concise, and directly addresses the question. 2. Only give me the answer and do not output any other words.
User Prompt	Question: {question} Reasoning Steps: {y_{CoT}} Answer:

Table 22: Detailed Prompts for Knowledge-Aware Generation (Keypoints Format).

Role	Content
Stage 1: Summary Note Extraction	
System Prompt	You are a helpful AI assistant that is good at extracting crucial information from documents which are helpful for answering a given question. For a given question, you need to extract a note that is directly relevant to generating an accurate answer. Important Note: 1. Don't directly provide the answer to the question. Instead, focus on extracting the relevant information that is essential for answering the question accurately and formulating a note. 2. The length of the response is limited, so make sure to include only the most relevant information.
User Prompt	Question: {question} Documents: {reference} Note:
Stage 2: Chain-of-Thought Reasoning	
System Prompt	You are a helpful AI assistant that is good at doing reasoning on the note to answer a given question. For a given question with relevant note, provide the step-by-step reasoning process to derive the answer. Make sure the reasoning steps are logical, coherent, and directly related to the question and the information in the note.
User Prompt	Question: {question} Note: {y_{Note}} Reasoning Steps:
Stage 3: Final Generation	
System Prompt	You are a helpful AI assistant that is good at generating final answers based on the reasoning steps provided for a given question. Important Notes: 1. Make sure the final answer is accurate, concise, and directly addresses the question. 2. Only give me the answer and do not output any other words.
User Prompt	Question: {question} Reasoning Steps: {y_{CoT}} Answer:

Table 23: Detailed Prompts for Knowledge-Aware Generation (Summary Format).

Role	Content
Module: Check Answerability	
System Prompt	You are a helpful AI assistant very good at judging whether the answer can be derived from the documents for a given question. You will be given a question, a set of documents, and golden answers. Please determine whether any of the golden answers can be derived from the documents for the given question. If any of the golden answers can be derived from the documents, the judgement should be "True". If none of the golden answers can be derived from the documents, the judgement should be "False". Do not output any explanation, only output the judgement.
User Prompt	Question: {question} Documents: {reference} Golden Answers: {golden_answers} Judgement:
Module: Knowledge Graph Refinement	
System Prompt	You are a professional AI assistant very good at refining the flawed knowledge graph. The knowledge graph is extracted from the documents for a given question so that it can be helpful for answering the question. But the current knowledge graph may contain incorrect information, redundant information, or lack critical information, making it impossible to deduce the correct answer for the given question. You will be given a question, a set of documents, the flawed knowledge graph and golden answers. Your task is to add, remove, or modify the content in the knowledge graph to make it accurate and relevant to answering the question. Make sure that the refined knowledge graph's content is both relevant to answering the question and entirely derived from the provided document. Important Notes: 1. The new knowledge graph should be refined based on the flawed knowledge graph, don't start from scratch. 2. The refined knowledge graph should be of the same format as the flawed knowledge graph. 3. Do not directly add the golden answers to the knowledge graph, the refined knowledge graph should be derived from the documents. 4. Do not output the explanation of your changes, only output the refined knowledge graph!
User Prompt	Question: {question} Documents: {reference} Flawed Knowledge Graph: {s⁻} Golden Answers: {golden_answers} Refined Knowledge Graph:

Table 24: Prompts used for Data Construction and Refinement.