Relevance to Utility: Process-Supervised Rewrite for RAG

Jaeyoung Kim[*†1], **Jongho Kim**[*2], **Seung-won Hwang**[‡2], **Seoho Song**[3], **Young-In Song**[3]
LG AI Research[1], Interdisciplinary Program in Artificial Intelligence, Seoul National University[2]
Naver Corp[3]
jae.young@lgresearch.ai
{jongh97, seungwonh}@snu.ac.kr
{song.seoho, song.youngin}@navercorp.com

Abstract

Retrieval-augmented generation systems often suffer from a gap between optimizing retrieval relevance and generative utility. With such a gap, retrieved documents may be topically relevant but still lack the content needed for effective reasoning during generation. While existing bridge modules attempt to rewrite the retrieved text for better generation, we show how they fail by not capturing "document utility". In this work, we propose R2U, with a key distinction of approximating true utility through joint observation of rewriting and answering in the reasoning process. To distill this observation reliably, R2U scales the process to isolate individual rewrites. We further construct utility-improvement supervision by measuring the generator's gain of the answer under the rewritten context, yielding signals for fine-tuning and preference optimization. We evaluate our method across multiple open-domain question-answering benchmarks. The empirical results demonstrate consistent improvements over strong bridging baselines.[1]

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Bai et al., 2023) have demonstrated strong generative capabilities across a wide range of tasks, yet they continue to struggle with queries that require knowledge beyond their parametric memory. Retrieval-augmented generation (RAG) addresses this limitation by retrieving external documents and conditioning generation on them. However, a persistent gap remains between retrieval relevance and generative utility: documents that are topically relevant do not necessarily support effective reasoning during answer generation (Ke et al., 2024).

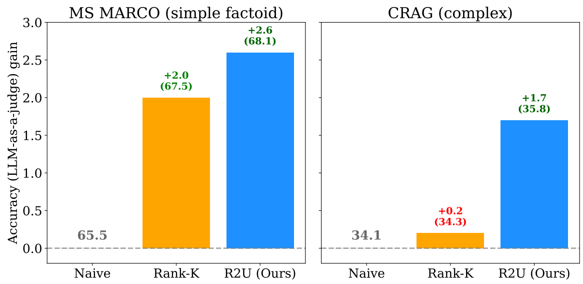

Figure 1: Accuracy gap between MS MARCO and CRAG across different bridging models. All models take the top-10 retrieved documents as input, rewrite them, and pass the results to the generator.

To bridge this gap, recent work has introduced intermediate modules between retrieval (R) and generation (G) that rewrite retrieved documents to better support downstream generation. Such **bridge** modules include document reranking (Pradeep et al., 2023; Yang et al., 2025; Xiao et al., 2023) constrained to reorder documents without modifying their content, and query-focused summarization (QFS) (Xu et al., 2023; Li et al., 2024; Yoon et al., 2024; Hwang et al., 2024; Edge et al., 2024; Chen et al., 2025) allowed to prune or enrich content selectively. We abstract all these reranking, summarization and pruning as **rewriting** of transformations from retrieved documents D into D', considering q as well.

Despite their reported success, we argue that existing rewriting approaches fail to reliably capture **document utility**, which we define as "the extent to which a document improves a generator's ability to produce the correct answer". As a result, rewriting models are particularly detrimental when the generation task requires reasoning. For example, Figure 1 shows Rank-K (Yang et al., 2025)[2] improves performance on MS MARCO (Nguyen et al., 2016), where the answer is simply extracted from the documents, while it suffers on

[*]Equal contribution. The order is decided by coin toss.
[†]This work was done while the author was a student at Seoul National University.
[‡]Corresponding author.
[1]Source code: https://github.com/ldilab/R2U

[2]Other baseline results are in Table 2, Rank-K shows the highest average performance among the baselines.

Findings of the Association for Computational Linguistics: ACL 2026, pages 30274–30293
July 2-7, 2026

CRAG (Yang et al., 2024), where generation requires beyond extraction.

We attribute this failure to two fundamental limitations. First, their rewriting process, independent of the reasoning process to derive the answer, favors shortcut documents that appear to support simple answer extraction. For complex queries, these shortcuts mislead generators by presenting superficial matches while discarding the evidence required for reasoning steps. Second, the binary answer for supervision is unreliable for reasoning-heavy tasks, as a rewrite might provide crucial partial evidence yet still result in an incorrect answer due to downstream generation errors. Binary supervision discards these high-utility yet failed attempts.

To address the limitations, we propose **R**elevance **to** **U**tility (R2U), a training framework that addresses both limitations: Rather than treating rewriting as a disjoint preprocessing step, we view it as part of the reasoning trajectory that leads to an answer. This motivates a joint observation of rewriting and generated answer, providing a more principled utility proxy for rewriting.

Building on this formulation, R2U distills utility-aligned supervision. First, we obtain process supervision from reasoning trajectories which include joint rewrite-answer step. Instead of naive generation, we scale the process to isolate individual rewrites, disentangling the utility of each document from the trajectories. Second, we improve the quality of supervision by explicitly measuring how rewritten documents change a downstream generator's likelihood of producing the correct answer. These signals are then used to train a lightweight rewriter via standard supervised fine-tuning and preference optimization.

We evaluate our approach on multiple complex question-answering tasks covering diverse query types and reasoning requirements. The results show that R2U improves the average F1 score by 6.8% over the naive RAG baseline and outperforms the best existing baselines by 5.6%, consistently enhancing performance across tasks.

2 On Bridging Objectives

In this section, we formalize the role of bridge modules in RAG system and identify two fundamental limitations in existing rewriting-based approaches. These limitations motivate the design choices of our training framework, which we address explicitly in Section 3.

2.1 Motivation: Existing Bridging Models

In a standard RAG system, a generator ϕ produces an answer a given a query q and a set of retrieved documents D:

$$a \sim P_\phi(a \mid q, D). \tag{1}$$

A line of recent work introduces a bridge module that rewrites or transforms the retrieved documents to improve their utility for generation. Such approaches include document reordering (Pradeep et al., 2023; Xiao et al., 2023), content filtering (Yoon et al., 2024; Li et al., 2024), and structured reconstruction (Chen et al., 2025). These methods can be summarized as:

$$a \sim P_\phi(a \mid q, D'), \text{ where } D' \sim P_\theta(D' \mid q, D), \tag{2}$$

where θ denotes a rewriting module and D' is the rewritten document set.

Training such bridge modules typically relies on result-level signals. In particular, rewritten documents are sampled and evaluated based on whether the generator produces the correct answer:

$$\max_\theta \mathbb{E}_{D' \sim P_\theta(D' \mid q, D)} \left[\mathbf{1}(a = a^*) \right], \quad \text{where } a \sim P_\phi(a \mid q, D'). \tag{3}$$

We refer to this paradigm as **answer-level supervision**, since documents are rewritten relevant to the query and judged solely by the final correctness of the generated answer. While effective in practice, this formulation suffers from two fundamental limitations that prevent it from reliably capturing **document utility improvement**, which we define as below.

Document utility improvement
The extent to which a rewritten document shifts the generator's distribution toward the correct answer

The first limitation is the disjoint approximation of the document utility. The objective of document rewriting is not merely to paraphrase the original content, but to generate documents that explicitly facilitate the reasoning process for answer generation. However, most existing approaches approximate this objective in a decoupled manner. Rewritten documents are first generated from $p(D' \mid q, D)$ and only afterward filtered using the answer. As a result, it is challenging to determine whether a rewritten document contributes to generating the answer or merely co-occurs with successful generation due to generator shortcuts or parametric knowledge.

The second limitation is the binary answer as a preference signal. Such binary supervision fails to differentiate rewrite quality, obscuring the degree to which a document enhances the generator's reasoning. It treats all correct answers equally, ignoring cases where a rewrite lowers the generator's confidence but still yields a correct prediction due to model robustness. Conversely, a rewritten document that substantially increases the probability of the correct answer may be discarded simply because it failed to cross the decision threshold for the final generation. Consequently, this binary criterion leads to the inclusion of noisy data or the exclusion of high-utility supervision signals.

3 Proposed Approach: R2U

To address the two limitations identified in Section 2.1, we first clarify the objective that a document rewriting framework should satisfy. Specifically, rewritten documents should be (i) jointly generated with the answer, and (ii) selected based on soft improvement of the downstream generation. In the following sections, we formalize this objective from a joint perspective (§3.1), and then describe how we operationalize it via scaled process supervision (§3.2) and utility improvement (§3.3).

3.1 Formulation of R2U: Joint Perspective

Our key distinction is to treat document rewriting as part of the reasoning process that produces the answer, rather than as an independent preprocessing step. Accordingly, we model generation as a joint distribution over rewritten documents D' and the answer a: $P_\phi(a, D' \mid q, D)$.

This joint formulation provides a principled proxy for utility-aligned rewriting. By the definition of conditional probability, the posterior distribution of D' given an answer a satisfies:

$$P_\phi(D' \mid q, D, a) = \frac{P_\phi(a, D' \mid q, D)}{P_\phi(a \mid q, D)} \propto P_\phi(a, D' \mid q, D). \tag{4}$$

Sampling D' from the LLM's joint reasoning traces approximates sampling from this posterior, yielding rewrites that are explicitly grounded in answer generation. As a result, rewriting supervision derived from joint trajectories is naturally aligned with document utility.

While this formulation specifies what constitutes utility-aligned rewriting, it does not yet specify how to reliably obtain such supervision or how to distinguish high-utility rewrites from low-utility ones. We address these challenges by distilling joint process supervision from LLMs in an identifiable manner (§3.2) and by selecting supervision signals based on measured utility improvement for a downstream generator (§3.3).

3.2 Scaled Process Supervision

Figure 2(a) illustrates our data collection process, where for each query–document set (q, D), we collect joint supervision traces $P(a, D' \mid q, D)$ by prompting the LLM to rewrite and generate the answer within a single autoregressive trajectory.

Empirical observation of joint generation We observe that sufficiently capable LLMs naturally generate such a joint structure in their reasoning traces. In manual inspection of reasoning traces from multiple LLMs[3], we find that rewritten documents frequently appear as intermediate steps for the answer generation, even without explicit prompting to do so, as shown in examples in Table 7. This empirical evidence suggests that large LLMs implicitly generate rewritten documents and answers in a single autoregressive process, allowing us to effectively sample documents D' from a joint distribution $P_{\text{LLM}}(a, D' \mid q, D)$.

Disentangling document-level utility via scaled observation A naive way to collect joint rewrite–answer traces is to prompt the LLM with the full retrieved document set D and let it rewrite all documents before producing an answer. However, when multiple rewritten documents $D' = \{d'_1, \ldots, d'_{|D|}\}$ are generated in a single trajectory and followed by a correct answer, it becomes difficult to disentangle which rewritten document actually contributed to the answer generation. For instance, a correct answer may arise because one document was rewritten effectively, while others were rewritten poorly or not at all.

[3]From chain-of-thought prompting (Wei et al., 2022), 10 queries per model on Llama-3.1-{70b, 405b}-Instruct, claude-3-5-sonnet, Deepseek-r1, Mistral-{large,large2}, reka-flash, and snowflake-arctic. The examples and more detailed results are in Appendix A.

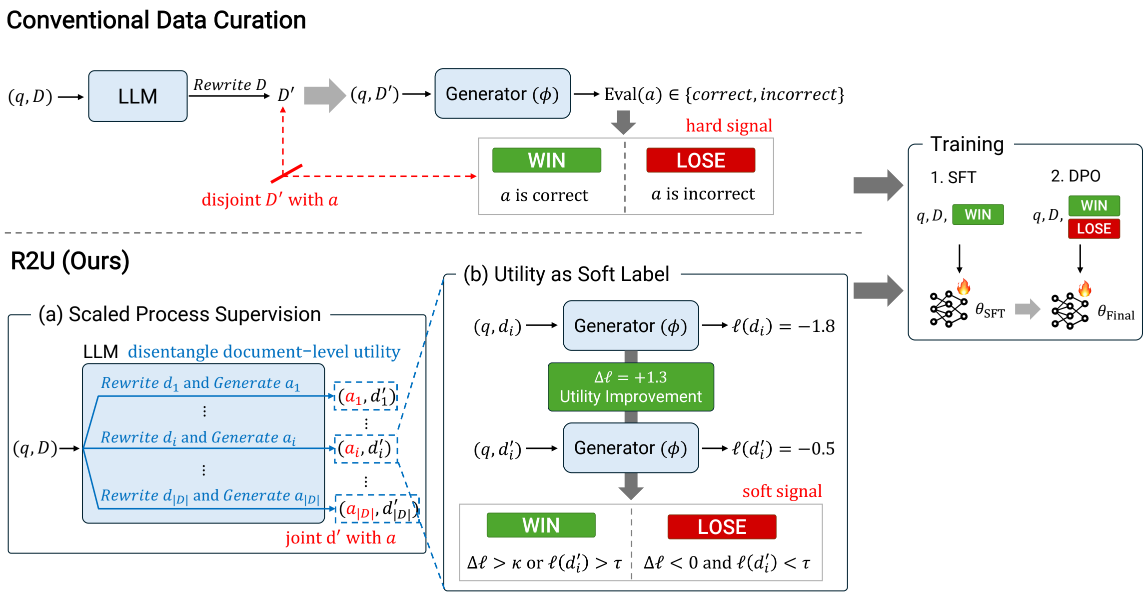

Figure 2: Overview of R2U. (a) Scaled Process Supervision: LLM rewrites each document while conditioning on the remaining documents, yielding (a, d') pairs suitable for distillation. (b) Utility as Soft Label: Rather than relying on hard binary signals, we evaluate d_i' by measuring their soft utility improvement, and then divide D into win and lose for training rewriter via standard SFT and DPO.

To resolve the challenge, we leverage inference scaling as a form of controlled observation, which scales the generation process to the document level to disentangle the supervision signals. The scaling is depicted with the blue color in Figure 2(a). Concretely, for each query–document set (q, D), we perform $|D|$ LLM calls, each targeting a single document d_i for rewriting, conditioned on the query and the remaining documents $D \setminus \{d_i\}$ as context. From scaled runs, it yields a set of isolated supervision traces $\{(d_i, d_i', a_i)\}_{i=1}^{|D|}$. This strategy preserves the benefits of a global context that captures inter-document relations (Burges, 2010), while ensuring that the specific utility of the target document is explicitly disentangled. Implementation details and prompts are provided in Appendix B.

3.3 Utility as Soft Label

To address the coarseness of answer-level supervision, we construct training signals based on **utility improvement** measured with respect to a downstream generator. Rather than labeling rewritten documents by hard signal, we quantify soft utility gain to guide preference tuning.

Data curation For a given query q, each retrieved document d, and its rewritten counterpart d', we evaluate the downstream generator ϕ by computing the average log-likelihood of the gold answer a^*:

$$\begin{aligned} \ell(d) &= \mathbb{E}_t \left[\log P_\phi(a_t^* \mid q, d, a_{<t}^*)\right], \\ \ell(d') &= \mathbb{E}_t \left[\log P_\phi(a_t^* \mid q, d', a_{<t}^*)\right]. \end{aligned} \tag{5}$$

We define the utility improvement of a rewritten document as $\Delta\ell = \ell(d') - \ell(d)$. A rewritten document is considered utility-improving if it increases the generator's likelihood of the correct answer.

With this utility improvement, we construct supervision signals by filtering rewritten documents. Specifically, as depicted in Figure 2(b), we gather rewritten documents with $\Delta\ell > 0$ or with absolute likelihood $\ell(d')$ exceeding a minimum threshold τ and treat the gathered document set (D_w') as the positive example (*win*), while the remaining documents from the same query (D_l') are treated as the negative example (*lose*).

This criterion ensures that training focuses on rewritten documents that measurably improve the generator's answer distribution, rather than merely yielding correct answers.

Training We use the resulting supervision signals to train the student rewriter following standard practice. The training procedure consists of supervised fine-tuning (SFT) followed by direct preference optimization (DPO), as commonly adopted in prior works.

For SFT, we train the rewriter to generate rewritten documents conditioned on the query and retrieved document set, using only utility-improving

(D'_w) examples. Concretely, given a query q, retrieved documents D, and a utility-improving rewrite D'_w, we maximize the log-likelihood of the rewrite:

$$\mathcal{L}_{\text{SFT}}(\theta) = \mathbb{E}_{(q,D,D'_w)}\left[\log P_\theta(D'_w \mid q, D)\right] \quad (6)$$

where θ denotes the parameters of the student rewriter. This stage initializes the model to produce high-quality rewrites aligned with utility-improving behavior.

Starting from the SFT-trained model, we further apply direct preference optimization. For each query-documents pair, we construct preference pairs (D'_w, D'_l) from utility-improving (*win*) and non-improving (*lose*) rewrites. Following standard DPO formulations, we optimize the rewriter to assign a higher likelihood to the preferred rewrite:

$$\mathcal{L}_{\text{DPO}}(\theta) = -\mathbb{E}_{(q,D,D'_w,D'_l)}[\log\sigma(r_\theta(q,D,D'_w) - r_\theta(q,D,D'_l))] \quad (7)$$

where $\sigma(\cdot)$ denotes the sigmoid function. Following prior work (Rafailov et al., 2023), we simplify $r_\theta = \beta \log P_\theta(D'|q,D) - \beta \log P_{\theta_{\text{SFT}}}(D'|q,D)$, where θ_{SFT} denotes SFT-trained parameters.

At inference time, our trained rewriter model is called just once per query, taking the query and the retrieved document set as input and producing a rewritten document set.

4 Experiments

4.1 Experimental Setting

In this section, we describe the datasets, metrics, and implementation details in our experiments.

Datasets and Metrics Following evaluation conventions (Chirkova et al., 2025; Yoon et al., 2024), we evaluate R2U on both Wikipedia-based and web-scale question answering(QA) benchmarks to assess utility-oriented rewriting under different retrieval and reasoning conditions.

For wikipedia-based corpus, we consider multi-hop and ambiguity-focused QA datasets: **HotpotQA** (Yang et al., 2018), **2WikiMultihopQA (2WIKI)** (Ho et al., 2020), **MuSiQue** (Trivedi et al., 2022), and **AmbigQA** (Min et al., 2020). These datasets require reasoning over multiple pieces of evidence or resolving ambiguous queries by conditioning on appropriate contextual evidence. We report Exact Match (EM) and F1 scores, following standard QA evaluation (Hwang et al., 2024; Yoon et al., 2024).

For web-scaled corpus to assess practical effectiveness under web-scaled retrieval conditions, we additionally use web-based QA benchmarks. We include **MS MARCO** (Nguyen et al., 2016) for single-hop QA, evaluated on a randomly sampled subset of 500 queries following prior work (Lee et al., 2025; Back et al., 2021). For more challenging reasoning scenarios beyond extraction, we evaluate on **CRAG** (Yang et al., 2024), which covers diverse query types requiring synthesis and reasoning over web documents. For these datasets, we report Accuracy (ACC) using an LLM-based evaluation protocol with Llama-3.1-405B, following the dataset's evaluation setup (Yang et al., 2024)[4].

Following Li et al. (2024), we use the retrieval contexts provided by each dataset and fix the number of retrieved documents to top-10. We note that datasets such as MuSiQue and CRAG often do not include gold documents in the retrieved set, making them well-suited for evaluating utility-oriented rewriting beyond extraction.

Baselines For reranking-based baselines, we include **BGE-Reranker-Large** (Xiao et al., 2023), a cross-encoder pointwise reranker that scores documents independently, and **RankZephyr** (Pradeep et al., 2023), a listwise reranker distilled from GPT-4 that jointly models document interactions. We further include **Rank-K** (Yang et al., 2025), a recent strong reranker that incorporates explicit reasoning signals. These methods represent strong reranking-based bridges that improve utility through document ordering without modifying document content.

For summarization-based baselines, we compare against **ComPACT** (Yoon et al., 2024) and **Refiner** (Li et al., 2024), which distill LLM knowledge to generate query-focused abstractive summaries, and **EXIT** (Hwang et al., 2024), which improves generation by pruning irrelevant document content. These approaches explicitly rewrite document content to better align with the query.

For unified baseline, we include **Provence** (Chirkova et al., 2025), which unifies reranking and summarization into a single bridge model.

[4] We conducted a reliability analysis of LLM-as-a-judge evaluation in Appendix D.

Method	True Utility	Distill	Module Size	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
				EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	-	-	-	52.8	64.4	47.2	63.8	38.4	52.0	12.1	20.2	37.6	50.1
BGE-Reranker	✗	✗	560M	52.8	64.0	48.7	65.5	38.7	53.4	13.0	20.9	38.3	51.0
RankZephyr	✗	✓	7B	52.6	64.0	49.1	65.8	39.5	53.9	12.7	20.5	38.5	51.1
Rank-K	✗	✓	32B	53.5	64.8	48.8	65.5	40.0	54.3	12.6	20.4	38.7	51.3
Provence	▲	✓	430M	52.3	64.7	43.4	58.1	35.1	45.1	12.5	19.6	35.8	46.9
ComPACT	✗	✓	7B	24.0	30.2	47.8	63.7	43.8	53.3	10.6	16.7	31.6	41.0
Refiner	▲	✓	7B	31.1	37.6	47.5	63.2	43.6	53.3	12.0	18.7	33.6	43.2
EXIT	▲	✗	2B	49.2	60.0	38.5	51.5	32.7	41.1	8.5	14.4	32.2	41.8
R2U	✓	✓	3B	**54.5**	**67.2**	**49.4**	**68.2**	**54.4**	**65.7**	**16.0**	**26.3**	**43.6**	**56.9**

Table 1: Performance comparison on AmbigQA, HotpotQA, 2Wiki, and MuSiQue using Llama-3.1-8B-Instruct. Results are reported with EM and F1, with the last two columns showing the average EM and F1 across the datasets. The best performance for each target dataset is marked in bold.

Method	MS MARCO	CRAG	Average
Naive	65.5 ± 0.3	34.1 ± 0.3	49.8
BGE-Reranker	65.0 ± 0.7	34.2 ± 0.3	49.6
RankZephyr	66.1 ± 0.5	34.7 ± 0.2	50.4
Rank-K	67.5 ± 0.4	34.3 ± 0.2	50.9
Provence	65.5 ± 0.2	32.9 ± 0.2	49.2
ComPACT	55.4 ± 0.6	29.7 ± 0.3	42.6
Refiner	65.3 ± 0.2	29.3 ± 0.1	47.3
EXIT	57.0 ± 0.2	31.5 ± 0.2	44.3
R2U	**68.1** ± 0.1	**35.8** ± 0.3	**52.7**

Table 2: Performance comparison on MS MARCO and CRAG using Llama-3.1-8B-Instruct. Results are reported with ACC using LLM (Llama-3.1-405B) evaluation. The last column shows the average ACC across the two datasets. The best performance for each target dataset is marked in bold.

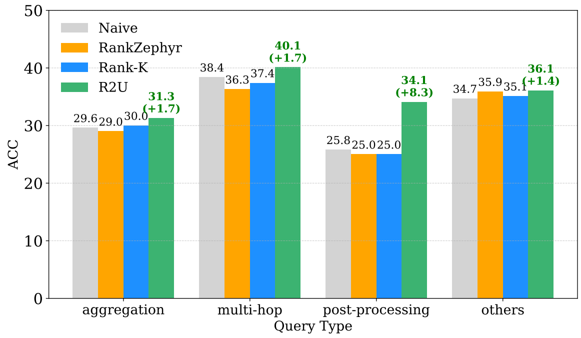

Figure 3: Comparison of ACC across various query types in CRAG. The value in parentheses indicates the difference relative to Naive RAG.

Implementation details We use Llama-3.3-70B-Instruct (AI@Meta, 2024) as the teacher model to generate joint rewrite–answer trajectories. For utility-based preference, we set $\kappa = 4.0$ and the utility threshold $\tau = 0.05$ to strictly divide the preference, following CoRAG (Wang et al., 2025). To prevent answer leakage, we strictly filter out queries for which the teacher can generate the correct answer without conditioning on retrieved documents.

For generating traces, we construct two training datasets from MS MARCO: 14k utility-improving rewrite examples for SFT, and 8k preference pairs for DPO. These datasets are derived by measuring the utility improvement of rewritten documents with respect to Llama-3.2-3B-Instruct as the downstream generator.

We train the student rewriter with Llama-3.2-3B-Instruct as the backbone. SFT is conducted using DeepSpeed ZeRO-3 (Rasley et al., 2020) on four NVIDIA A6000 GPUs for three epochs, followed by DPO for 500 steps with $\beta = 0.1$. Inference is performed with vLLM (Kwon et al., 2023), using a temperature of 0 for reproducibility.

4.2 Results

Research Questions To evaluate the effectiveness of our approach, we address the following research questions:

- RQ 1: Does R2U enhance the performance of the models in QA tasks?
- RQ 2: Does R2U generalize to diverse generators?
- RQ 3: Does each component of R2U contribute to overall effectiveness?

4.3 RQ 1: Does R2U enhance the performance of the models in QA tasks?

Overall Performance and True Utility Table 1 and 2 highlight the dominant performance of R2U compared to various baselines in question answering tasks. We separate the results to focus on different evaluation aspects: Table 1 assesses disambiguation or multi-hop reasoning ability on the Wikipedia corpus, and Table 2 evaluates performance on the web corpus to validate that our gains in document utility generalize to practical settings.

As shown in the column named "True Utility" in Table 1, baselines are marked with ✗ if they use a utility function without considering the answer (i.e., $P_\theta(d' \mid q, d)$ only) and with ▲ if they incorporate the answer supervision (Eq. 3). Overall, R2U achieves the highest scores, demonstrating particularly strong performance on disambiguation and multi-hop queries, where cross-document reasoning is required to answer the query. Compared to baselines, R2U outperforms NaiveRAG by absolute +6.8% gain and the strongest competitor by +5.6% in terms of average F1, supporting the hypothesis that aligning document utility with answer generation effectively guides reasoning over multiple sources. Table 2 further shows that R2U maintains superior performance on the web corpus, including MS MARCO and CRAG. This confirms that the benefits of our true utility function extend beyond structured Wikipedia datasets to more diverse, open-domain web content.

Comparison with Distilled Baselines While baselines such as RankZephyr (Pradeep et al., 2023) and Refiner (Li et al., 2024) also leverage knowledge distillation from large language models for document rewriting, their performance remains lower than that of R2U. This indicates that the observed gains primarily stem from aligning document utility, rather than from distillation alone. Notably, R2U achieves superior performance with a comparatively small 3B module, showing that our approach enables effective distillation even with limited model capacity.

Performance on Complex Query Types To further investigate the role of document utility in complex query resolution, we categorize CRAG queries into four types: aggregation, multi-hop, post-processing-heavy, and others (Yang et al., 2024). This categorization allows us to disentangle the varying levels of document interaction and reasoning required by different query types. We compare R2U against RankZephyr and Rank-K, which achieve the strongest overall performance among existing baselines on CRAG.

As shown in Figure 3, R2U attains the highest performance across all complex query types. While Rank-K also slightly improves performance on aggregation queries, this gain comes at the cost of accuracy on other types. In contrast, R2U demonstrates consistently superior performance, with particularly pronounced gains in scenarios demanding higher document utility. Specifically, R2U outperforms alternatives on aggregation and multi-hop queries, where the effective integration of multiple documents is crucial. The improvement is even more substantial for post-processing-heavy queries, which require both multi-document reasoning and significant post-processing.

Method	Sub-Answer Inclusion Rate (%)
Provence	43.8
ComPACT	39.6
Refiner	34.3
EXIT	58.4
R2U	**59.5**

Table 3: Intermediate evidence coverage on MuSiQue, measured by the proportion of gold sub-answer entities explicitly retained in the rewritten documents.

These findings highlight that by enhancing the true utility of retrieved documents through the bridge module, R2U enables the generator to more effectively leverage relevant evidence, outperforming baselines.

Intermediate evidence coverage To examine whether utility improvement reflects reasoning quality, we analyze intermediate evidence coverage on MuSiQue, where gold sub-answer entities are labeled. We measure the proportion of sub-answer entities explicitly retained in the rewritten documents. As shown in Table 3, R2U preserves the highest proportion among all methods. Since MuSiQue requires multi-hop reasoning over intermediate evidence, this result suggests that utility improvements are associated with preserving reasoning-relevant evidence rather than superficial lexical changes.

4.4 RQ 2: Does R2U generalize to diverse generators?

To address RQ2, we evaluate R2U with various generators, against the best-performing module of each type based on F1 score: Rank-K for reranking, Refiner for summarization, and Provence for combining both. We use Llama (Llama-3.2-{1B,3B}-Instruct, Llama-3.1-8B-Instruct) and Qwen (Qwen2.5-{0.5B,1B,3B,7B}-Instruct) Family (Bai et al., 2023) as generators.

The results in Figure 4 demonstrate the effectiveness and generalizability of our method over dif-

Method	MS MARCO	CRAG	AmbigQA	HotpotQA	2WIKI	MuSiQue
	ACC	ACC	F1	F1	F1	F1
Naive RAG	70.5	34.0	64.4	64.0	52.1	19.9
ComPACT (70B)	**70.7**	35.6	69.8	72.9	66.3	24.7
R2U (70B)	**70.7**	**37.2**	**71.8**	**73.7**	**67.8**	**37.1**

Table 4: Comparison of R2U with ComPACT using Llama-3.3-70B-Instruct as a rewriter and Llama-3.1-8B-Instruct as a generator. The best performance for each target dataset is marked in bold.

Method	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
R2U	**54.5**	**67.2**	**49.4**	**68.2**	**54.4**	**65.7**	**16.0**	**26.3**	**43.6**	**56.9**
w/o DPO	51.9	64.7	48.7	66.5	49.6	61.1	14.3	22.5	41.1	53.7
w/o DPO and soft label	53.6	66.4	48.7	66.8	45.8	59.4	12.7	20.9	40.2	53.4
w/o DPO and scaled obs.	44.3	53.1	38.4	51.8	28.6	36.5	9.1	15.2	30.1	39.2

Table 5: Ablation study. The best performance for each target dataset is marked in bold.

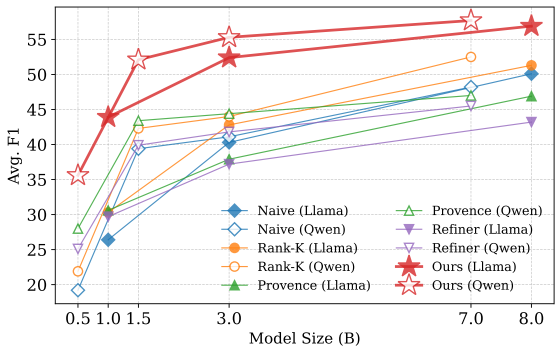

Figure 4: Comparison of average F1 scores across various model sizes using Llama (filled markers) and Qwen (hollow markers).

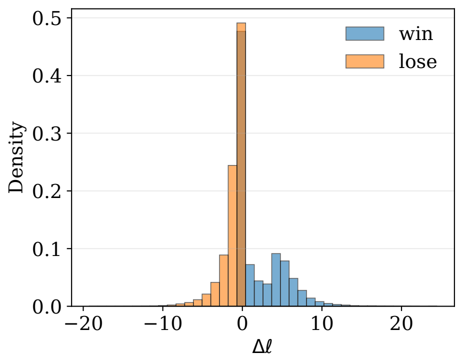

Figure 5: Distribution of utility improvement ($\Delta\ell$) for win and lose rewrites.

ferent sizes and types of generators. Notably, R2U enables Llama-1B and Qwen-1.5B to outperform their larger counterparts (Llama-3B and Qwen-3B) with other methods. These results highlight that smaller models that lack strong reasoning capabilities often gain more from the effectiveness of R2U. The full results are available in Table 13-18 in Appendix F.

4.5 RQ 3: Does each component of R2U contribute to overall effectiveness?

We conduct an ablation study to examine the contribution of each component of R2U: (i) joint process supervision, (ii) utility-based soft label, and (iii) scaled observation for disentangling document-level utility.

Effect of joint process supervision. Table 4 compares our joint rewrite–answer supervision against a disjoint rewriting baseline adapted from ComPACT, where documents are rewritten independently of answer generation. Using the same large rewriter (Llama-3.3-70B-Instruct), R2U consistently outperforms the disjoint baseline across all datasets. This confirms that coupling rewriting with answer generation is crucial for capturing document utility aligned with downstream reasoning, rather than merely producing query-relevant summaries.

Effect of training objectives (SFT vs. DPO). Table 5 shows that removing DPO (w/o DPO) leads to a clear performance drop, although the SFT-only variant still outperforms most baselines. This indicates that SFT on utility-improving rewrites already provides strong gains, while DPO further refines the model by sharpening distinctions between high- and low-utility rewrites. To enable a more controlled analysis, we first ablate DPO and conduct the remaining ablation studies on the SFT-trained model, thereby isolating the effect of each component.

Effect of utility-based soft label. To isolate the role of utility-based supervision selection, we compare R2U with a naive variant that trains on all rewrites generated by the teacher model without soft labeling (w/o soft label). In R2U (w/o DPO), rewrites with negative utility improvement ($\Delta\ell < 0$) are explicitly excluded from training, preventing counterproductive supervision, as shown in the left of Figure 5. In contrast, the w/o soft label variant indiscriminately learns from rewrites, including those that reduce the downstream generator's likelihood of the correct answer. Notably, in Table 5, this difference leads to weaker performance, since rewrites with $\Delta\ell < 0$ are treated as positive training signals in w/o soft label. These results indicate that soft label based on explicit utility improvement is critical for learning utility-aligned rewriting behavior.

Effect of scaled observation. Finally, removing scaled observation (w/o scaled obs.) causes the largest performance degradation. This variant collapses to naive joint rewriting, where document-level contributions are entangled within a single trajectory. The sharp drop in Table 5 demonstrates that isolating each document's contribution via scaled, controlled observations is essential for learning reliable document-level utility signals.

5 Related Works

5.1 Retrieval and Generation Gap in RAG

Retrieval modules (Robertson and Zaragoza, 2009) in RAG systems are commonly optimized using ranking metrics such as nDCG and MRR, which rely on relevance judgments based on expected human behavior (Järvelin and Kekäläinen, 2002). However, these metrics do not fully capture the needs of generative models consuming the retrieved context. For example, LLMs often exhibit positional biases distinct from humans, disproportionately focusing on tokens at the beginning and end of contexts while neglecting those in the middle (Liu et al., 2024). Furthermore, LLMs are sensitive to distracting passages (Amiraz et al., 2025; Cuconasu et al., 2024), and performance degradation from distracting passages becomes particularly severe when paired with strong retrievers (Jin et al., 2025). Consequently, a bridging model (Ke et al., 2024) is necessary to align retrieval outputs with the generative model's requirements, thereby reconciling human-based retrieval optimization with generation needs to improve the RAG performance.

5.2 Bridging the Gap

Optimizing relevance in the retriever may not align with what the generation needs. Bridging models aim to mediate this gap by re-ranking or rewriting retrieved content before it is passed to a generator.

One direction is rerankers, which reorder documents to reflect usefulness to the query better. Pointwise rerankers such as BGE (Xiao et al., 2023) score each document individually, while listwise approaches like RankZephyr (Pradeep et al., 2023) consider global ordering, often distilled from strong LLMs like GPT-4. Recent works such as Rank-K (Yang et al., 2025) go further by distilling reasoning-oriented models (e.g., DeepSeek-R1). However, the rerankings are performed independently of the generator's answer generation.

A second line of bridging work uses QFS, either selection-based or generation-based. Selection methods select salient segments directly from retrieved text, as EXIT (Hwang et al., 2024), which uses GPT-4o pseudo-labels to train sentence selectors but risks spurious success when reasoning shortcuts appear. Generation-based methods instead rewrite evidence with LLMs: RECOMP-Abst (Xu et al., 2023) distills GPT-3.5 summaries, while later works like CompAct (Yoon et al., 2024) and Refiner (Li et al., 2024) scale to stronger (GPT-4) or multiple LLMs. Some QFS methods also filter the training dataset based on answer correctness. (Hwang et al., 2024; Yoon et al., 2024; Chirkova et al., 2025).

Our work follows the generation-based direction but differs in that it exploits the reasoning traces of LLMs. Instead of distilling only summaries, we capture how the model reasons about the query to ensure that document rewriting yields higher utility.

6 Conclusion

In this work, we address the gap between retriever and generator in RAG. We propose to observe reasoning process as process supervision, then distill to a smaller model for practical deployment. Moreover, R2U enables more effective use of retrieved information for answer generation. Empirical results on diverse datasets confirm that our rewritten documents significantly improve performance over strong baselines.

Acknowledgement

This work was supported by Naver, Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2022-0-00077/RS-2022-II220077, AI Technology Development for Commonsense Extraction, Reasoning, and Inference from Heterogeneous Data), Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) [NO.RS-2021-II211343, Artificial Intelligence Graduate School Program (Seoul National University)].

7 Limitation

While R2U shows significant gains, generating process-supervised data relies on costly LLM reasoning traces. Due to this expense, we were unable to conduct data generation with larger teacher models beyond 70B. Moreover, our evaluation is focused on QA tasks, leaving broader RAG applications for future work. As with other distillation-based approaches, our method inherently depends on the quality of the teacher model. Though our utility-improvement data curation and potential shortcut filtering are designed to mitigate spurious supervision, fully eliminating all non-causal patterns inherited from the teacher remains challenging.

In this work, our goal is not to establish a new state-of-the-art system on benchmarks. We intentionally fix the retriever and generator across all methods so that the incremental gain can be attributed to document rewriting alone, viewed as a bridge between retrieval and generation. On the other side, many stronger systems on the leaderboard adopt different configurations, including more powerful retrievers, larger generators, and repeated retrieval or inference runs. Future work can apply our method on top of these stronger pipelines.

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*.

AI@Meta. 2024. Llama 3 model card.

Chen Amiraz, Florin Cuconasu, Simone Filice, and Zohar Karnin. 2025. The distracting effect: Understanding irrelevant passages in RAG. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 18228–18258, Vienna, Austria. Association for Computational Linguistics.

Seohyun Back, Akhil Kedia, Sai Chetan Chinthakindi, Haejun Lee, and Jaegul Choo. 2021. Learning to generate questions by learning to recover answer-containing sentences. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1516–1529, Online. Association for Computational Linguistics.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, and 1 others. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Boyu Chen, Zirui Guo, Zidan Yang, Yuluo Chen, Junze Chen, Zhenghao Liu, Chuan Shi, and Cheng Yang. 2025. Pathrag: Pruning graph-based retrieval augmented generation with relational paths. *arXiv preprint arXiv:2502.14902*.

Nadezhda Chirkova, Thibault Formal, Vassilina Nikoulina, and Stéphane Clinchant. 2025. Provence: efficient and robust context pruning for retrieval-augmented generation. *arXiv preprint arXiv:2501.16214*.

Florin Cuconasu, Giovanni Trappolini, Federico Siciliano, Simone Filice, Cesare Campagnano, Yoelle Maarek, Nicola Tonellotto, and Fabrizio Silvestri. 2024. The power of noise: Redefining retrieval for rag systems. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 719–729.

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

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6609–6625.

Taeho Hwang, Sukmin Cho, Soyeong Jeong, Hoyun Song, SeungYoon Han, and Jong C Park. 2024. Exit: Context-aware extractive compression for enhancing retrieval-augmented generation. *arXiv preprint arXiv:2412.12559*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Bowen Jin, Jinsung Yoon, Jiawei Han, and Sercan O Arik. 2025. Long-context llms meet rag: Overcoming challenges for long inputs in rag. In *The Thirteenth International Conference on Learning Representations*.

Zixuan Ke, Weize Kong, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. 2024. Bridging the preference gap between retrievers and llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10438–10451.

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

Youngwon Lee, Seung-won Hwang, Daniel F Campos, Filip Graliński, Zhewei Yao, and Yuxiong He. 2025. Inference scaling for bridging retrieval and augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 7324–7339, Albuquerque, New Mexico. Association for Computational Linguistics.

Zhonghao Li, Xuming Hu, Aiwei Liu, Kening Zheng, Sirui Huang, and Hui Xiong. 2024. Refiner: Restructure retrieval content efficiently to advance question-answering capabilities. *arXiv preprint arXiv:2406.11357*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12.

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

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

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

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. *Advances in neural information processing systems*, 36:53728–53741.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA. Association for Computing Machinery.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

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

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. 2025. Chain-of-retrieval augmented generation. *CoRR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. 2023. C-pack: Packaged resources to advance general chinese embedding. *Preprint*, arXiv:2309.07597.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *arXiv preprint arXiv:2310.04408*.

Eugene Yang, Andrew Yates, Kathryn Ricci, Orion Weller, Vivek Chari, Benjamin Van Durme, and Dawn Lawrie. 2025. Rank-k: Test-time reasoning for listwise reranking. *arXiv preprint arXiv:2505.14432*.

Xiao Yang, Kai Sun, Hao Xin, Yushi Sun, Nikita Bhalla, Xiangsen Chen, Sajal Choudhary, Rongze Gui, Ziran Jiang, Ziyu Jiang, and 1 others. 2024. Crag-comprehensive rag benchmark. *Advances in Neural Information Processing Systems*, 37:10470–10490.

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.

Chanwoong Yoon, Taewhoo Lee, Hyeon Hwang, Minbyul Jeong, and Jaewoo Kang. 2024. Compact: Compressing retrieved documents actively for question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21424–21439.

Appendix

A Details and Examples of LLM's Reasoning Process

We conducted a systematic inspection across 8 model families with 10 MS MARCO examples, finding that this rewriting behavior occurred in 95% of cases. We elicit reasoning traces with the following prompt:

```
Think step by step to use the provided
documents to answer a user's question.
Question: . . .
Documents:
Document 1: . . .
Document 2: . . .
. . .
```

A step is annotated as a rewrite when the trace explicitly cites one or more source documents (e.g., "from Document 1, 2, ..." or "Document 1 supports ...") A case satisfies the prerequisite if at least one such rewrite appears before the final answer. Table 7 presents representative cases, illustrating how robust LLMs produce d' tailored to the target answer under its reasoning trace.

We expanded this manual inspection to quantify how frequently LLMs naturally perform rewriting across three distinct datasets, where we sampled 50 examples and evaluated them manually. As shown in Table 6, LLMs exhibit this behavior in nearly 100% of cases across all tasks.

Model	CRAG	HotpotQA	MS MARCO
Llama-3.1-70B	98%	100%	100%
Mistral-Large2	98%	98%	98%

Table 6: Percentage of inherent rewriting across datasets.

B Prompts for R2U

Table 8 provides the full CoT prompt template for generating scaled process supervision. We design a revised CoT prompt template to observe the desired document distribution from the reasoning process. To ensure better supervision quality, we have made several design choices: (1) Separate the document rewriting step from the reasoning and answering step, and discard the latter step to prevent leakage. (2) Restrict rewrites to retrieved content. (3) Encourage cross-document interaction. (4) Allow skipping irrelevant documents.

For each query with a set of retrieved documents D (typically $|D| = 10$), we executed $|D|$ separate generation passes. In the i-th pass (where $i \in \{1, \ldots, |D|\}$), we designated the current document d_i as the {document} and the remaining set $D \setminus \{d_i\}$ as the {other documents}. Every document is rewritten exactly once as the primary target while being conditioned on the global context of all other documents. We then aggregated these individual traces to form the final supervision set $\{(d_i, d'_i, a_i)\}_{i=1}^{|D|}$ for the query.

Table 9 presents the R2U prompt template for document rewriting during both training and inference. The template takes a query and the full set of retrieved documents as input and rewrites them in a single forward pass.

C Details of Computational Cost

The computational cost of LLM for R2U is incurred only once during offline supervision data construction, while its inference-time cost is identical to that of standard bridge models.

For dataset construction, we perform 14k queries with 10 retrieved documents each, resulting in 140k LLM calls. Each call produces an average of 100.84 output tokens, yielding approximately 14 million generated tokens in total. In contrast, the strongest baseline, Rank-K, reports 100k LLM calls with an average of 2,408.85 output tokens per call, resulting in approximately 240 million generated tokens. Despite using more LLM calls, R2U requires roughly $6\times$ fewer generated tokens, as each call rewrites only a single document rather than the entire document set.

Moreover, Rank-K relies on DeepSeek-R1, a larger and more computationally expensive model, whereas R2U uses Llama-3.3-70B-Instruct as the teacher. As a result, the total cost of constructing supervision data for R2U is substantially lower than that of Rank-K.

D Analysis of LLM Evaluation

To quantify the reliability of LLM evaluation, we conducted a human verification study on 100 sampled instances (50 from MS MARCO, 50 from CRAG), where three of the authors independently annotated the model outputs. The results demonstrate that Llama-3.1-405B achieves the highest alignment with human judgment (Cohen's Kappa: 0.960). We compared this against other strong evaluators, including Claude-3.5-Sonnet (0.900) and

Mistral-Large2 (0.742). The significantly higher agreement of Llama-3.1-405B confirms it is the most reliable proxy for human evaluation in this context, justifying its usage as an evaluator.

To definitively rule out evaluator bias, we also conducted the cross-model evaluation using Claude-3.5-Sonnet as an independent judge. The results in Table 11 confirm that R2U consistently outperforms the baselines under the different evaluators, demonstrating that our improvements are robust and not an artifact of self-preference bias.

E Case Study

The results in Table 12 show the qualitative analysis of R2U's intermediate evidence coverage on MusiQue. The rewritten documents restructure the evidence into reasoning-aligned statements, explicitly linking intermediate entities and reducing the need for implicit cross-document inference.

F Performance of R2U with Different Model Sizes

Tables 13–18 provide the per-dataset results, including both EM and F1, corresponding to the average F1 trends shown in Figure 4. The results consistently show that R2U outperforms the baselines across all datasets and model sizes.

G Ai Assistants In Research Or Writing

We used ChatGPT to shorten the script to fit within the page limit.

H Use Or Create Scientific Artifacts

We used existing scientific artifacts for research purposes, and the use of existing artifacts was consistent with their intended applications.

Used Datasets The datasets we used are in English and are anonymized and protected from offensive content.

- MS MARCO dataset (Nguyen et al., 2016): non-commercial research purposes only
- CRAG dataset (Yang et al., 2024): CC BY-NC 4.0 license
- AmbigQA dataset (Min et al., 2020): CC BY-SA 3.0 license
- HotpotQA dataset (Yang et al., 2018): CC BY-SA 4.0 license
- 2WIKI (Ho et al., 2020): Apache 2.0 license
- MusiQue (Trivedi et al., 2022): CC BY 4.0 license

Used Models

- Qwen-2.5 (Bai et al., 2023): Apache license 2.0
- Llama-3 (AI@Meta, 2024): custom commercial license

Question: does xpress bet charge to deposit money in your account
Claude-3.5-sonnet: Let me analyze this step by step: First, I'll look for specific information about deposit fees in the documents. From **Document 7**: "They also give you the ability to wager for free online, make free deposits, and have free withdrawals." From **Document 9**: "No membership, monthly, or per-wager fees" However, there is one exception noted in **Document 2:** When using certain payment methods like PayNearMe, there may be a convenience fee (normally $3.99) Looking at deposit methods available (from Documents 4 and 9): ...
Deepseek-r1: <think> Okay, let me try to figure out if Xpressbet charges a fee for depositing money into an account. The user is asking specifically about deposit fees. First, I'll go through each document provided to see if any mention of fees is made. Let's start with **Document 1**. It talks about the minimum deposit amounts ($10 in most states, $25 in Massachusetts) but doesn't mention any fees. It does mention a bonus when a friend deposits and wagers, but that's not related to fees. **Document 2** is more relevant. The first step says, "Tell the cashier how much you want to deposit... they will collect your funds plus the convenience fee (normally $3.99)." So here, there's a convenience fee of $3.99 when depositing via a cashier using PayNearMe, I think. That's a direct mention of a fee for deposit. **Document 3** repeats the PayNearMe steps but doesn't mention the fee again. **Document 4** lists deposit methods like Xpressfund, Green Dot, PayNearMe, etc., but doesn't state fees. **Document 5** is similar to 2 and 3, again about PayNearMe steps without mentioning the fee here. **Document 6** is about mobile PayCode setup; again no fee mentioned. **Document 7** is really important. It says, "make free deposits, and have free withdrawals." So this directly states that deposits are free. But wait, there's a conflict here with **Document 2** which mentions a $3.99 fee. Maybe the fee depends on the deposit method? Looking at **Document 7** again: "They also give you the ability to wager for free online, make free deposits, and have free withdrawals." So according to this, deposits are free. But Document 2's convenience fee is for a specific method, PayNearMe via cashier. So maybe some methods have fees, others don't. **Document 9** repeats that there are no per-wager fees and lists deposit methods again without mentioning fees, except for the possible ones in Document 2. Document 8 and 10 talk about other services (U.S. Bank Mobile app, Western Union) which might not be directly related to Xpressbet's own fees. So putting it all together: ...
llama-3.1-405b: To answer the user's question, we need to find information about Xpressbet's deposit fees. Let's analyze the documents: **Document 2** mentions a convenience fee of $3.99 when depositing funds into an Xpressbet account using a cashier. **Document 7** states that Xpressbet does not charge per wager fees, and also mentions that deposits are free. **Document 9** also mentions that there are no membership, monthly, or per-wager fees, and lists various deposit methods, but does not mention any fees associated with depositing funds. Based on the information provided, it appears that ...

Table 7: An example from the MS MARCO development set where LLMs naturally perform rewriting during their reasoning process. Models were provided with all 10 passages to generate the answer.

CoT prompt template for generating scaled process supervision

You are given a user query, a set of other documents, and one target document. Your task is to answer the query based solely on the target document, while using the other documents to help clarify and improve the target document. To do this, you will first rewrite the target document to make it more accurate, complete, and easier to reason with in the context of the query.
Step 1: Rewrite the **target document** using the other documents so that it is accurate, non-redundant, and maximally informative in light of the query.
Step 2: Use the rewritten document to answer the query. Think step by step and explain your reasoning.

[Step 1 Guidelines]
1. Relevance Check
- If parts of the target document are not relevant to the query or not logically connected to it, remove them without explanation.
- Partial relevant information is acceptable. The document does not need to be sufficient to fully answer the query.
- If the document does not contain any content relevant to the query, return exactly: NO REWRITE

2. Integration
- If information in the target document appears similar to what is in the other documents, do not overwrite or remove information—highlight subtle differences or unique nuances.
- If the target document contradicts the other documents, preserve the target document's version.
- When rewriting, preserve original terminology from the target document or reuse vocabulary from the query wherever possible, rather than paraphrasing unnecessarily.
- Enrich or clarify the target version using the other documents if helpful, but maintain its perspective as primary.

3. Focus & Prioritization
- Emphasize information that most directly supports answering the query.
- Less essential or tangential content may be shortened, but not removed if it offers a unique nuance or framing.

[IMPORTANT RULES]
- Do **not** answer the query directly in Step 1.
- Do **not** mention the existence of other documents.
- Do **not** explain your changes or reference the task.
- Do **not** include statements like "target document does not provide" or "according to other documents."
- Return only the rewritten version of the target document with no commentary or formatting.

[Query]
{query}
[Other Documents]
{other documents}
[Target Document]
{document}

You must follow the output format exactly as specified below.
Output format:
Step 1. Document Rewriting: <your rewritten version of the target document>
Step 2. Answer: <your answer to the query, along with explanation based only on the rewritten document>

Table 8: CoT prompt template for generating scaled process supervision. It observes the desired document distribution from the reasoning process.

R2U prompt template for document rewriting
You are a helpful assistant. Your job is to analyze the documents below and rewrite only the parts that help clarify or refine the information in relation to the question. List each relevant document to better support answering the question. Do not include unrelated documents. Question: {query} Documents: {documents}

Table 9: R2U prompt template for document rewriting.

Dataset	F1 = 1 (Correct)	F1 = 0 (Incorrect)
HotpotQA	-0.1403	-0.3784
2Wiki	-0.1400	-0.2982

Table 10: Average log-likelihood under rewritten contexts for correct and incorrect answers.

Method	MS MARCO	CRAG	Average
Naive	58.6	34.1	46.4
BGE-Reranker	59.0	34.3	46.7
RankZephyr	59.8	34.2	47.0
Rank-K	59.4	34.1	46.8
Provence	58.4	32.4	45.4
ComPACT	50.0	29.4	39.7
Refiner	58.8	30.7	44.8
EXIT	50.8	31.7	41.3
R2U	**62.4**	**35.1**	**48.8**

Table 11: Performance comparison on MS MARCO and CRAG, using Claude-3.5-Sonnet as an LLM evaluator. The best performance for each target dataset is marked in bold.

Query	Intermediate evidence	Original documents	Rewritten documents
In which state is Vera Barbosa's place of birth located?	Vila Franca de Xira, Lisbon District	Vila Franca de Xira is a municipality in the Lisbon District in Portugal. The population in 2011 was 136,886, [...] Vera Barbosa (born 13 January 1989 in Vila Franca de Xira) is a Portuguese track and field athlete [...]	Vera Barbosa was born in Vila Franca de Xira. [...] **Vila Franca de Xira is a municipality in the Lisbon District in Portugal, where Vera Barbosa was born.** [...]
What is the seat of the county where Van Hook Township is located?	Mountrail County, Stanley	[...] The Mountrail County Courthouse in Stanley, North Dakota was built in 1914 and served Mountrail County as its [...] Van Hook Township is a township in Mountrail County in the U.S. state of North Dakota.	[...] **Van Hook Township is a township in Mountrail County in the U.S. state of North Dakota.** At the time of the 2000 Census, its population was 42, and estimated to be 41 as of 2009. **The county seat for Mountrail County is Stanley.** [...]

Table 12: Case study examples showing how rewriting surfaces intermediate evidence needed for multi-hop reasoning.

Method	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	33.0	41.7	20.1	30.3	19.8	26.6	2.9	7.1	19.0	26.4
BGE-Reranker	32.2	40.6	24.1	35.6	20.8	27.8	3.7	8.0	20.2	28.0
RankZephyr	33.6	42.3	24.3	35.9	21.9	29.1	3.3	7.7	20.8	28.8
Rank-K	35.3	43.0	25.9	37.1	25.1	32.0	4.5	8.9	22.7	30.3
Provence	38.8	47.8	23.3	33.2	25.5	32.1	5.4	9.4	23.3	30.6
ComPACT	19.1	25.3	30.2	41.6	32.1	39.0	5.5	9.8	21.7	28.9
Refiner	22.5	29.2	29.5	40.6	30.7	37.4	6.9	11.6	22.4	29.7
EXIT	28.3	37.7	14.9	25.9	13.2	22.2	1.9	6.2	14.6	23.0
R2U	**50.1**	**60.3**	**35.3**	**49.4**	**38.6**	**47.4**	**11.4**	**18.6**	**33.9**	**43.9**

Table 13: Results on Llama3.2-1B-Instruct with EM and F1 scores.

Method	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	39.1	52.4	39.0	55.3	26.5	41.1	6.3	12.5	27.7	40.3
BGE-Reranker	39.1	52.5	42.4	59.4	30.7	45.0	7.1	13.7	29.8	42.7
RankZephyr	39.3	52.8	42.8	59.7	31.1	45.3	6.5	12.8	29.9	42.7
Rank-K	39.1	52.8	42.9	60.1	31.0	44.9	7.0	13.5	30.0	42.8
Provence	43.2	56.4	35.3	49.4	25.0	35.8	5.1	10.0	27.2	37.9
ComPACT	19.9	26.7	41.6	58.0	32.7	45.1	4.8	10.2	24.8	35.0
Refiner	27.6	36.8	40.5	56.4	30.3	43.3	6.4	12.2	26.2	37.2
EXIT	40.4	51.5	29.1	41.6	17.8	27.2	3.9	9.1	22.8	32.4
R2U	**47.5**	**61.5**	**45.9**	**64.7**	**46.1**	**59.4**	**15.0**	**24.0**	**38.6**	**52.4**

Table 14: Results on Llama3.2-3B-Instruct with EM and F1 scores.

Method	AmbigQA		HotpotQA		2WIKI		MusiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	8.9	22.6	9.9	22.7	9.8	22.6	3.0	9.0	7.9	19.2
BGE-Reranker	8.7	22.6	13.9	28.7	10.8	25.4	4.0	10.6	9.4	21.8
RankZephyr	8.9	22.7	14.0	28.7	11.5	25.9	4.3	11.0	9.7	22.1
Rank-K	8.2	22.1	14.1	29.3	11.7	25.3	4.2	10.8	9.6	21.9
Provence	24.4	38.0	15.8	30.2	17.0	30.9	6.6	12.9	16.0	28.0
ComPACT	9.3	19.4	19.1	35.6	19.4	34.4	5.6	12.2	13.4	25.4
Refiner	12.0	22.2	17.7	34.0	17.3	31.5	5.7	12.6	13.2	25.1
EXIT	13.4	28.0	11.2	25.1	10.0	23.5	4.1	10.2	9.7	21.7
R2U	**24.9**	**41.8**	**22.9**	**41.1**	**22.7**	**40.2**	**9.8**	**19.1**	**20.1**	**35.6**

Table 15: Results on Qwen2.5-0.5B-Instruct with EM and F1 scores.

Method	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	40.7	50.6	37.2	50.8	30.9	38.3	10.1	18.0	29.7	39.4
BGE-Reranker	39.9	50.2	42.7	57.4	35.2	43.5	10.7	18.9	32.1	42.5
RankZephyr	40.9	50.9	42.0	56.5	35.8	43.7	11.1	19.0	32.5	42.5
Rank-K	40.6	50.8	43.1	57.3	34.9	42.4	10.7	18.8	32.3	42.3
Provence	46.4	58.3	38.6	52.8	34.9	42.4	12.2	20.1	33.0	43.4
ComPACT	20.4	29.3	43.8	59.2	39.7	47.8	9.7	16.8	28.4	38.3
Refiner	25.2	35.0	43.1	58.4	38.2	46.0	12.2	20.3	29.7	39.9
EXIT	41.7	52.0	33.8	46.5	30.8	37.5	10.0	17.6	29.1	38.4
R2U	**48.8**	**61.6**	**46.5**	**64.0**	**47.0**	**57.1**	**15.8**	**25.7**	**39.5**	**52.1**

Table 16: Results on Qwen2.5-1.5B-Instruct with EM and F1 scores.

Method	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	43.8	54.5	37.2	51.4	34.8	42.3	9.4	16.3	31.3	41.1
BGE-Reranker	44.9	55.4	44.0	58.9	41.6	49.7	11.2	18.4	35.4	45.6
RankZephyr	44.1	54.8	43.1	58.0	40.3	48.3	10.7	17.7	34.6	44.7
Rank-K	44.2	55.1	42.0	57.2	38.6	46.3	10.3	17.4	33.8	44.0
Provence	47.5	58.6	40.2	54.3	37.6	44.6	12.6	20.0	34.5	44.4
ComPACT	22.0	29.4	45.8	60.8	43.6	51.5	10.5	17.3	30.5	39.8
Refiner	29.1	37.7	44.7	59.7	41.8	49.7	12.9	20.2	32.1	41.8
EXIT	42.4	53.6	35.6	48.3	33.1	40.0	9.4	15.9	30.1	39.5
R2U	**49.9**	**62.5**	**49.7**	**66.8**	**54.8**	**64.3**	**17.8**	**27.5**	**43.1**	**55.3**

Table 17: Results on Qwen2.5-3B-Instruct with EM and F1 scores.

Method	AmbigQA		HotpotQA		2WIKI		MuSiQue		Average	
	EM	F1	EM	F1	EM	F1	EM	F1	EM	F1
Naive	49.1	60.8	46.6	62.1	39.4	48.7	12.4	21.2	36.9	48.2
BGE-Reranker	49.6	61.5	49.8	66.1	45.7	54.7	13.8	22.6	39.7	51.2
RankZephyr	49.5	61.2	50.6	66.7	44.7	54.0	14.3	22.8	39.8	51.2
Rank-K	49.4	60.9	51.0	67.5	49.2	59.1	13.8	22.5	40.9	52.5
Provence	48.9	61.6	43.3	58.0	39.0	47.1	13.0	21.3	36.1	47.0
ComPACT	24.9	33.5	49.0	64.6	45.1	53.4	10.4	18.0	32.4	42.4
Refiner	33.3	43.9	47.6	63.2	45.2	53.1	13.3	21.8	34.9	45.5
EXIT	46.7	57.8	40.1	53.8	36.5	44.4	11.5	20.0	33.7	44.0
R2U	**53.8**	**66.2**	**51.9**	**69.6**	**57.3**	**67.2**	**17.4**	**27.9**	**45.1**	**57.7**

Table 18: Results on Qwen2.5-7B-Instruct with EM and F1 scores.