electronics

MDPI

Article

RewriteGen: Autonomous Query Optimization for Retrieval-Augmented Large Language Models via Reinforcement Learning

Yixuan Zhao [1], Zihao Fan [1], Yingying Cao [1], Zhengjia Lyu [2] and Jingyuan Li [1,*]

[1] School of Computer and Artificial Intelligence, Beijing Technology and Business University, Beijing 102401, China
[2] iFlytek Co., Ltd., Courtyard No. 8, Dongbeiwang West Road, Haidian District, Beijing 100193, China
* Correspondence: li.jingyuan.jerry@btbu.edu.cn

Abstract

Large Language Models (LLMs) have achieved substantial progress in knowledge-intensive tasks, particularly through Retrieval-Augmented Generation (RAG) frameworks. However, existing RAG systems often suffer from performance degradation when input queries are misaligned with retrieval requirements, and effectively coordinating retrieval with reasoning remains challenging—especially for multi-hop questions requiring iterative retrieval steps. To address these challenges, we propose ReWriteGen, a unified framework that integrates query rewriting, retrieval augmentation, and complementary generation within a coordinated architecture, optimized using reinforcement learning (Group Relative Policy Optimization, GRPO) and Direct Preference Optimization (DPO). ReWriteGen introduces a retrieval-aware query rewriting mechanism to better align input queries with external knowledge. The framework optimizes retrieval-augmented answers without requiring supervised reasoning annotations.Our experiments show that ReWriteGen consistently outperforms traditional RAG baselines across three multi-hop QA benchmarks: HotpotQA, MuSiQue, and 2Wiki. On HotpotQA, ReWriteGen achieves improvements of 5.32 and 5.10 percentage points in EM and LLM-based evaluation, respectively, compared to the strongest baseline. Corresponding gains of 11.90 and 7.18 are observed on MuSiQue, and 15.45 and 18.60 on 2Wiki.ReWriteGen enhances the coordination between retrieval and reasoning in LLMs, delivering consistent performance gains while reducing reliance on supervised reasoning annotations and extensive task-specific engineering.

Keywords: query rewriting; retrieval-augmented generation; reinforcement learning; large language models; natural language processing

Check for updates

Academic Editor: Arkaitz Zubiaga

Received: 27 January 2026
Revised: 27 February 2026
Accepted: 27 February 2026
Published: 3 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Retrieval-Augmented Generation (RAG) has become a central paradigm for large language models (LLMs) in tasks such as open-domain question answering and knowledge-intensive reasoning. By integrating external retrieval modules, RAG allows models to effectively leverage external knowledge, especially for tasks that require multi-hop reasoning. However, traditional RAG frameworks, including the retrieve-then-read and Rewrite–Retrieve–Read approaches, often suffer from the problem of component decoupling. These systems treat query rewriting, document retrieval, and answer generation as isolated steps, which presents challenges in ensuring the optimal interaction between them. Specifically, the retrieval component may not align well with the user's original query,

and answer generation might not fully benefit from the retrieved knowledge, resulting in information loss, retrieval inaccuracies, and error propagation in multi-hop reasoning tasks.

In the retrieve-then-read framework, the query is directly passed to the retriever without much optimization, meaning the retrieved documents may not be perfectly aligned with the true intent of the query. Similarly, the Rewrite–Retrieve–Read framework improves the query formulation by introducing a rewriting step, but the overall system still relies on external retrievers and static models. This reliance on training data and pseudo-labels for query rewriting and the separation of the various components limits the flexibility and effectiveness of the system, especially when dealing with complex, knowledge-intensive tasks.

Reinforcement Learning (RL) offers a promising direction for enhancing RAG systems. Compared to conventional supervised learning with fixed training objectives, RL enables policy optimization through task-level reward signals, such as answer correctness, reasoning consistency, and format compliance. This allows the model to adapt its reasoning strategy dynamically based on feedback from the task environment. However, applying RL to RAG is non-trivial and presents two major challenges. First, designing a reward function that simultaneously encourages accurate generation while promoting effective retrieval alignment is difficult. Second, stabilizing the interaction between retrieval and generation during iterative optimization requires careful coordination to prevent error propagation.

To address these limitations, we propose ReWriteGen, a novel framework that integrates query rewriting, retrieval, and answer generation within a unified reasoning architecture. Unlike traditional RAG systems or the Rewrite–Retrieve–Read framework, which treat query rewriting and retrieval as separate modules, ReWriteGen optimizes the entire reasoning process, ensuring a smooth interaction between query rewriting, retrieval, and generation through RL.

The key innovation of ReWriteGen lies in its retrieval-aware query rewriting mechanism. Instead of directly issuing the original user query to the retriever, the model learns to refine and reformulate the query to better align with the knowledge distribution in the retrieval corpus. This process ensures that the retrieval results are more relevant to the query, which in turn improves the process of generating a downstream answer.

At the optimization level, ReWriteGen uses a modular reinforcement learning strategy. Specifically, Group Relative Policy Optimization (GRPO) (Guo et al. [1]) is employed to optimize the reasoning process as a whole. GRPO guides the model to improve its query rewriting and reasoning steps through task-level rewards. At the answer selection stage, Direct Preference Optimization (DPO) (Rafailov et al. [2]) refines the final answer selection, ensuring consistency and relevance between retrieval and generated responses.

By leveraging reinforcement learning to optimize query rewriting and integrate it with the reasoning chain, ReWriteGen enhances coordination between all stages of the reasoning process. This allows the model to autonomously improve its reasoning and retrieval processes without relying on labeled reasoning data or model distillation, as in the case of traditional frameworks.

Our main contributions are as follows:

- We propose ReWriteGen, a novel framework that integrates key components such as query rewriting, retrieval triggering, and answer generation, while utilizing DPO to optimize answer selection and GRPO to optimize the overall reasoning process. This framework enables autonomous optimization of the reasoning pipeline, eliminating the need for supervised reasoning traces, model distillation, or cold-start strategies.
- We introduce a unified reinforcement learning mechanism for ReWriteGen, utilizing GRPO to optimize the overall reasoning process and DPO to enhance the selection of the most relevant answer during reasoning.

- We conduct extensive experiments on open-domain question answering tasks, demonstrating the efficiency, broad applicability, and significant improvements of ReWriteGen over traditional RAG baselines. The results show that ReWriteGen exhibits across multiple multi-hop QA benchmarks, making it a promising solution for complex real-world applications.

2. Related Work

2.1. Retrieval-Augmented Generation

Retrieval-Augmented Generation frameworks enhance language models by integrating external knowledge, addressing challenges such as factual inaccuracies in knowledge-intensive tasks like question answering and dialogue systems. Early retrieval methods employed lexical-based techniques, such as BM25 (Robertson et al. [3]), which ranks documents using term frequency and inverse document frequency. BM25's simplicity ensures robustness in scenarios with limited training data (Thakur et al. [4]), but its reliance on exact term matching limits its ability to capture semantic nuances in complex queries.

To overcome these shortcomings, neural-based retrieval methods have gained prominence. Dense Passage Retrieval (DPR) dual encoders (Karpukhin et al. [5]) utilize BERT to map queries and documents to semantic vector spaces, allowing efficient similarity search and surpassing BM25 in open-domain question answering tasks. Further refinements, such as ANCE (Xiong et al. [6]), leverage contrastive learning to enhance retrieval precision, particularly in data-rich environments. However, these methods often treat queries as static inputs, which can constrain their adaptability to varied user intents.

To boost the performance of Retrieval-Augmented Question Answering tasks, researchers have explored innovative methods to integrate retrieval and generation components seamlessly. One effective approach is RAG-end2end (Choi et al. [7]), which simultaneously trains a retriever (e.g., Dense Passage Retriever, DPR) and a generator (e.g., Colbertv2 (Santhanam et al. [8])) using a shared loss function, fostering enhanced collaboration between the two components. Another method involves dynamically refining retrieval results during answer generation by invoking the retriever multiple times. For example, Fusion-in-Decoder (Lewis et al. [9]) and REALM (He et al. [10]) identify the top-k most relevant document segments for a query, pass each segment along with the question to large language models to produce k responses, and then synthesize these into a cohesive final answer. Additionally, recent advancements leverage user feedback or the quality of generated outputs to fine-tune the parameters of both the retriever and generator, as demonstrated in frameworks like Trainable Rewrite–Retrieve–Read, which optimizes performance through iterative refinement. These strategies collectively enhance the accuracy and robustness of ReQA systems by ensuring tighter integration and continuous improvement of retrieval and generation processes.

2.2. Query Optimization for Large Language Models

Large Language Models (LLMs) exhibit remarkable capabilities in knowledge-intensive tasks, but their large scale or proprietary nature often restricts direct fine-tuning, necessitating alternative optimization strategies for retrieval-augmented frameworks. Prompt engineering has emerged as a key approach to leverage LLMs without modifying their parameters. For example, HyDE (Gao et al. [11]) generates hypothetical documents from the original query, focusing on embedding similarity between answers rather than queries, thus enhancing retrieval relevance. Similarly, ReAct (Yao et al. [12]) and Self-Ask (Press et al. [13]) integrate chain-of-thought prompting (Schaeffer et al. [14], Wei et al. [15]) with external API interactions, enabling LLMs to decompose complex queries into simpler

subquestions or interactive tasks. These methods often employ a least-to-most prompting strategy, breaking down problems systematically to improve reasoning accuracy.

Advanced prompting techniques further exploit LLMs' reasoning capabilities. Few-shot prompting (Brown et al. [16]) guides query generation with examples, while zero-shot prompting strategies (Asai et al. [17]), such as direct instructions, produce high-quality synthetic queries with models like GPT-3. UDAPDR (Saad-Falcon et al. [18]) combines prompting with reranker distillation to enable unsupervised domain adaptation, demonstrating the versatility of prompt-driven query optimization. Additionally, step-back abstraction (Zheng et al. [19]) encourages LLMs to reason about high-level concepts before addressing specific queries, improving performance in complex reasoning tasks. GENREAD (Yu et al. [20]) adopts a generate-then-read approach, using cluster-based prompting to produce contextual documents, thereby expanding knowledge coverage for retrieval-augmented tasks.

Query rewriting offers a complementary strategy by reformulating queries to align with retrieval and generation objectives. A rewrite–retrieve–read framework (Ma et al. [21]) employs Proximal Policy Optimization (PPO) (Schulman et al. [22]) to train a T5-based rewriter that optimizes queries for frozen retrievers and LLM generators, achieving robust improvements in question-answering tasks. In contrast, contextual adaptation focuses on refining retrieved documents. PRCA (Yang et al. [23]), a reward-driven contextual adapter that operates between the retriever and generator, refines document representations through reinforcement learning to enhance relevance.

Despite these advancements, most query optimization methods rely on modular pipelines, separating query rewriting, retrieval, and generation. This fragmentation can lead to inefficiencies, particularly in tasks requiring seamless integration. Unlike previous methods, our approach incorporates a rewriting operation before retrieval to enhance retrieval efficiency, integrating query rewriting, retrieval triggering, and answer generation into a single large model, optimized end-to-end with GRPO, achieving autonomous query reformulation, significantly reducing data dependency, and outperforming non-reinforcement learning methods in open-domain question-answering tasks.

3. Materials and Methods

In this section, we introduce the ReWriteGen framework, which integrates query rewriting, retrieval-augmented generation, reinforcement learning, and DPO into a unified optimization pipeline. Unlike conventional modular RAG systems that treat rewriting, retrieval, and generation as loosely coupled components, our approach emphasizes coordinated decision-making within a single large language model.

The framework operates through three interconnected stages:

1. Query rewriting for better retrieval alignment;
2. Evidence acquisition through retrieval and complementary generation;
3. Preference-based answer optimization.

By modeling the entire reasoning process as a reinforcement learning problem, the system autonomously refines queries and improves answer quality.

Figure 1 illustrates the overall architecture of ReWriteGen. The structural workflow is described in Section 3.1, while the optimization mechanisms, reward formulation, and preference learning strategies are detailed in subsequent subsections.

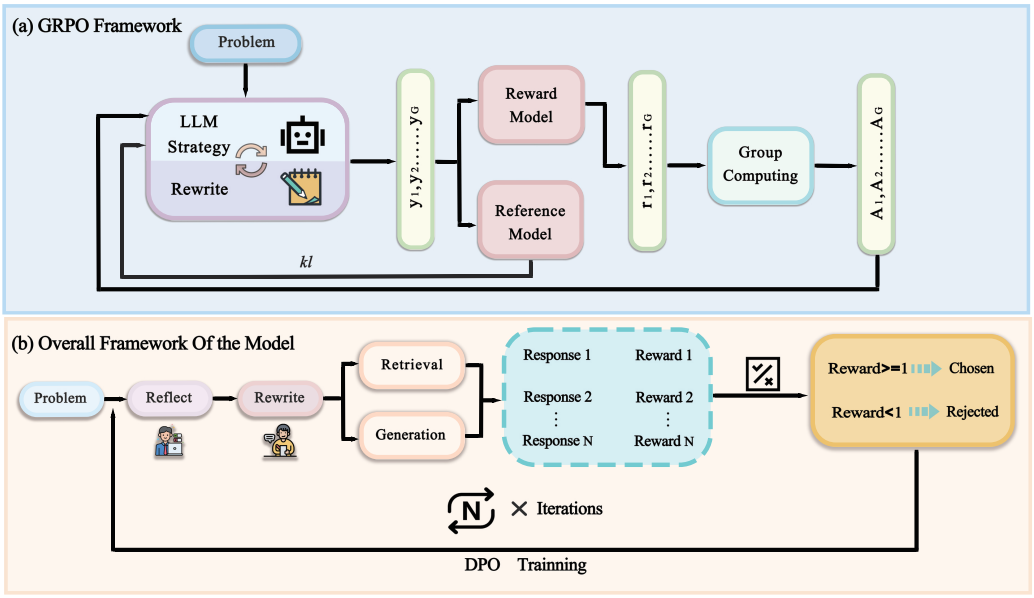


Figure 1. Overview of the ReWriteGen framework with GRPO algorithm integration.

3.1. Overview of the ReWriteGen Framework

ReWriteGen enhances retrieval-augmented reasoning by integrating query rewriting, retrieval triggering, and answer generation into a unified decision process. Rather than relying on independent modules, the framework enables the model to dynamically coordinate these components during inference and training.

Given an input query, the model first performs structured reflection and reformulates the query in a retrieval-aware manner. This rewriting step aims to reduce ambiguity and better align the query with the retrieval objective. The rewritten query is then used to retrieve relevant documents from the knowledge base.

In addition to retrieved evidence, the model may generate complementary content to compensate for potential gaps in retrieval. These two information channels—retrieved documents and generated references—are jointly considered when constructing candidate answers. This dual-source design improves robustness, particularly in multi-hop reasoning scenarios where retrieval alone may be insufficient.

To optimize this process, ReWriteGen employs GRPO for end-to-end learning and DPO for answer selection. GRPO models the reasoning workflow as a Markov Decision Process (MDP) and updates the policy based on reward feedback, enabling autonomous improvement of query rewriting and generation strategies. DPO further refines answer quality by comparing multiple candidate responses and promoting the most accurate one.

By unifying rewriting, retrieval, and generation within a reinforcement learning framework, ReWriteGen achieves iterative refinement during reasoning while maintaining structural consistency and answer accuracy.

To further clarify the complementary generation mechanism, we emphasize that this step is not performed in a random or unconstrained manner. The generation process is conditioned on both the original query and the retrieved evidence, and follows a structured prompting strategy designed to identify potential information gaps. Specifically, the model is prompted to analyze whether the retrieved documents sufficiently support multi-hop reasoning; if insufficient evidence is detected, the model generates supplementary contextual information to bridge missing reasoning steps.

To mitigate the potential introduction of noise, the generated content is not directly adopted as the final answer. Instead, it is incorporated as an auxiliary reasoning signal and subsequently evaluated through the reward function within the GRPO framework. Further-

more, DPO compares multiple candidate responses and promotes those that demonstrate higher factual consistency and contextual relevance. This design ensures that complementary generation improves reasoning completeness while minimizing noise propagation.

3.2. Reinforcement Learning

We model the query rewriting and retrieval-augmented generation process as a MDP. In the ReWriteGen framework, the model aims to optimize each decision step to maximize the quality of the generated answers. Specifically, the definitions of state, action, and reward are as follows:

- State (s_t): The model's state includes the input query, historical results, and the generated text sequence (such as <reflect> and <rewrite> tags). This information describes the interaction between the model and the external knowledge base, as well as the decisions made during the reasoning process.
- Action (a_t): At each time step t, the model chooses an action a_t, which is to generate the next token. This token could be regular text or control tags (such as <reflect> tag).
- Reward (r_t): The model's objective is to maximize the reward obtained. The reward function consists of the following components:
  - Format Reward: Ensures the generated text conforms to the expected format and includes all necessary tags.
  - Answer Reward: Evaluates the accuracy and relevance of the generated answer by comparing it with a ground-truth or expected answer.

Unlike Proximal Policy Optimization (PPO) [22], GRPO estimates the baseline from a group of rollouts, eliminating the need for a separate critic model, thereby reducing training complexity and enhancing policy stability. Given an existing policy $\pi_{\theta_{\text{old}}}$ and a reference policy $\pi_{\theta_{\text{ref}}}$, based on rollouts of G $\tau = \{y_i\}_{i=1}^{G} \sim \pi_{\theta_{\text{old}}}[\cdot|x]$ for each input $x \sim \mathcal{D}$, the objective of GRPO is to optimize the policy π_θ by maximizing:

$$\mathcal{J}(\theta) = \mathbb{E}_{x\sim\mathcal{D}}\left\{\mathbb{E}_{\tau\sim\pi_{\theta_{\text{old}}}[\cdot|x]}\left[\frac{1}{G}\sum_{i=1}^{G}\left[\min\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}A_i,\right.\right.\right.\right.$$
$$\left.\left.\left.\left.\text{clip}\left(\frac{\pi_\theta(y_i|x)}{\pi_{\theta_{\text{old}}}(y_i|x)}, 1-\epsilon, 1+\epsilon\right)A_i\right) - \beta \mathrm{D}_{\text{KL}}(\pi_\theta||\pi_{\theta_{\text{ref}}})\right]\right]\right\} \quad (1)$$

where $A_i = (r_i - \text{mean}(\{r_j\}_{j=1}^{G}))/\text{std}(\{r_j\}_{j=1}^{G})$ is the normalized advantage of the i-th rollout in the current group, ϵ is the clipping ratio, and β is the KL loss coefficient. A KL-divergence penalty prevents the policy from deviating significantly from the initial model. Through a single-stage reward mechanism (format learning and answer optimization), GRPO enables the model to autonomously learn query rewriting, achieving efficient retrieval and generation without requiring supervised reasoning traces.

3.3. Prompt Template Design

To guide the Qwen2.5-3B model in producing structured rollouts with query rewriting, we design a concise prompt template. The template ensures the model adheres to the required format, including reasoning, query rewriting, and final answer presentation. As shown in Figure 2, the prompt format is provided.

This prompt serves as the system prompt for the Qwen2.5-3B model, ensuring that query rewriting is performed before reasoning and that retrieved information is concisely summarized to inform subsequent reasoning steps.

Prompt Format

You are a helper who can solve a given problem step by step using Wikipedia search tools and self-generated references. Given a problem, please answer it by following these steps: 1. Provide the reasoning process of the query from a rewrite perspective and include it within the <reflect></reflect> tag.
2. Place the rewritten query within the <rewrite></rewrite> tag; the rewritten query will be used for subsequent searches.
3. Place the retrieved results within the <retrieve_result></retrieve_result> tag and include the generated references within the <gen></gen> tag.
4. Place the results generated by each rollout within the <result> tag and end with the </result> tag.
5. Place the final answer within the <answer></answer> tag.

Figure 2. Prompt Format.

3.4. Reward Model

In this study, we define the reward function used in our ReWriteGen framework to evaluate and guide the optimization process during reinforcement learning (RL). The reward function is composed of two key components: the format reward and the answer reward.

3.4.1. Format Reward

The format reward ensures that the generated text adheres to the required structure, which is essential for maintaining the coherence of the query rewriting and answer generation process. For each step, the model is awarded a small positive reward if it correctly includes the necessary tags, such as `<reflect>`, `<rewrite>`, `<retrieve_result>`, `<gen>`, `<result>`, and `<answer>`. These tags must appear in pairs, meaning that each opening tag (e.g., `<reflect>`) must be correctly matched with its corresponding closing tag (e.g., `</reflect>`). This reward reinforces the importance of maintaining the correct structure in the generated responses. If the required tags are missing or incorrectly paired, the model receives a penalty of zero for the format reward.

$$R_{\text{format}} = \begin{cases} 0.1, & \text{if all tags are correctly paired} \\ 0, & \text{otherwise} \end{cases}$$

3.4.2. Answer Reward

The answer reward evaluates the accuracy of the generated answer by comparing it with the ground truth or the expected response. This component is essential for ensuring that the model is providing high-quality, relevant answers to the input queries. The model is awarded a higher reward if its answer matches the expected answer (based on Exact Match or other evaluation metrics). If the answer is incorrect or irrelevant, the model will receive a lower reward, guiding it to improve over time.

The answer reward is integrated with the format reward to form a combined reward function, which is calculated as follows:

$$R = \begin{cases} \max(\text{Acc.} + R_{\text{format}}, \text{Acc.}) & \text{if Format is Good and Acc.} > 0 \\ 0.5 & \text{if Format is Good and Acc.} = 0 \\ 0 & \text{otherwise} \end{cases}$$

In this combined reward function, Acc. represents the accuracy of the answer, indicating how well the generated response matches the ground truth. The reward is maximized when the format is correct and the accuracy is greater than zero. If the format is correct

but the accuracy is zero, the model still receives a moderate reward of 0.5 to encourage further improvement. However, if either the format or the answer is completely incorrect, the model receives a reward of zero, signaling the need for improvement.

The answer reward plays a crucial role in guiding the model's behavior. If the answer is inaccurate, the model receives a lower reward, which encourages it to make decisions about whether to generate an answer or rely on retrieval. This mechanism enables the model to make better decisions between generating answers and retrieving relevant information, improving both answer relevance and overall performance.

This reward function is key in ensuring that the model not only generates accurate answers but also adapts to the task's needs by choosing the most appropriate method (generation vs retrieval) based on feedback from the task at hand.

3.5. Direct Preference Optimization (DPO) for Answer Generation

In addition to the GRPO optimization, we introduce DPO to further enhance the model's answer generation process. DPO operates by comparing multiple candidate answers generated from both retrieval and generation, enabling it to autonomously choose the final answer and optimize the query-answer generation process. The optimization objective for DPO is formulated as follows:

$$L_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l)\sim D}\left[\log \sigma\left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)}\right) - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)}\right] \quad (2)$$

where y_w represents the preferred answer (the correct or optimal answer), and y_l represents the less preferred answer (the incorrect or suboptimal answer). The function σ is the sigmoid function, and β is a hyperparameter that controls the strength of the preference signal. Additionally, π_θ denotes the current policy for generating answers, while π_{ref} is the reference model used for comparing responses.

This function encourages the model to assign higher probabilities to the preferred answer y_w and lower probabilities to the less preferred answer y_l. It ensures that the model is increasingly confident in selecting the optimal answer through continuous learning from preference-based data.

4. Experiments

This section outlines the experimental setup and results for evaluating our proposed query rewriting framework, which leverages RL to enhance retrieval-augmented generation in LLMs. The experiments are organized into five subsections: dataset description, baseline methods, evaluation metrics, implementation details, and main results.

4.1. Datasets

We evaluated our framework on three datasets: HotpotQA, MuSiQue, and 2Wiki, a synthetic dataset we constructed to test multi-hop reasoning. HotpotQA contains 90.4 k training and 7.4 k test samples requiring two-hop reasoning, MuSiQue comprises 19.9 k training and 2.4 k test samples with two to four-hop questions, and 2Wiki includes 10 k synthetic samples designed for two to three-hop reasoning tasks.

4.2. Baselines

To assess the effectiveness of our query rewriting approach, we compare it against several established methods, focusing on retrieval-augmented frameworks and query optimization techniques. The baselines include: (1) Naive Generation, where the LLM generates answers without retrieval; (2) Standard RAG, which concatenates retrieved documents with the input question; (3) Rewrite–Retrieve–Read, a framework that employs a

frozen LLM to rewrite queries before retrieval; (4) Iter-RetGen, which iteratively synergizes retrieval and generation; (5) Self-Ask, which generates its own search queries to retrieve information for answering questions; and (6) IRCoT, which interleaves retrieval with chain-of-thought reasoning.

4.3. Evaluation Metrics and Protocol

We adopt two primary metrics to evaluate answer correctness: Exact Match (EM) and LLM-as-a-Judge (LJ). EM measures whether the predicted answer exactly matches the ground truth after standard normalization (lowercasing, punctuation removal, and whitespace normalization), providing a strict and objective evaluation criterion. However, for multi-hop question answering tasks, large language models often generate explanatory responses rather than concise spans, making strict lexical matching potentially restrictive.

To better capture semantic equivalence, we employ LLM-as-a-Judge (LJ) evaluation using GPT-4. Instead of relying solely on conventional metrics such as F1 or BLEU—which may underestimate correctness when answers are semantically equivalent but lexically different—GPT-4 evaluates the alignment between the predicted answer and the ground truth in terms of factual consistency and final answer consistency.

GPT-4 was used as the primary automatic evaluator under a structured evaluation prompt. The complete assessment template is provided in Table 1, for reproducibility. During evaluation, the decoding temperature of GPT-4 was set to 0 to ensure deterministic outputs. All models were evaluated under identical prompts and evaluation settings to ensure fairness and consistency.

In addition, limited manual inspection was conducted on randomly selected samples to verify general consistency between GPT-4 judgments and expected answer correctness. The inspection confirmed that GPT-4 evaluations were largely aligned with human judgment in clearly correct and clearly incorrect cases.

Table 1. Template for GPT-4 Assessment.

Category	Description
Prompt	You are an advanced evaluation assistant. Based on the provided question and the reference answer, determine if the predicted answer accurately addresses the question.
Question: Predicted Answer: Golden Answer:	

4.4. Implementation Details

We train and evaluate our framework on Qwen2.5-3B. The reinforcement learning (RL) framework is built on verl, a versatile library that supports efficient policy optimization for large language models. Training is conducted on the HotpotQA training set, comprising 90.4 k samples, for two epochs.

The retrieval environment is implemented using FlashRAG (Jin et al. [24]), a standard toolkit for RAG research. We employ E5-base-v2 as the retriever, with Wikipedia data from December 2018 serving as the knowledge base. All corpus indexing and embedding preprocessing are handled by FlashRAG. During both training and evaluation rollouts, we retrieve the top-5 documents for each query. Baseline methods are implemented under the same FlashRAG framework to ensure fair comparison.

Training is performed on 8 NVIDIA A100 (40 GB) GPUs with full-parameter optimization and gradient checkpointing enabled. The detailed experimental hyperparameter settings are summarized in Table 2.

Table 2. Experimental Parameter Settings.

Parameter	Value
Learning Rate	1×10^{-6}
Batch Size	256
Number of Epochs	2
Number of Rollouts	5
Rollout Temperature	1.0
KL Loss Coefficient	0.001
Clip Ratio	0.2

4.5. Main Results

We evaluate the performance of RewriteGen against several baseline models across three datasets: HotpotQA, MuSiQue, and 2Wiki. The results are summarized in Table 3, using Exact Match (EM) and LLM as evaluation metrics. The results show that query rewriting plays a critical role in enhancing the ability of large language models to handle complex query understanding and retrieval tasks. Specifically, the rewriting step enables the LLM to better eliminate query ambiguity and provide contextualization, thus improving the clarity of intent and relevance of retrieved information. Compared to prior rewriting models, RewriteGen achieves significant performance gains without requiring fine-tuning or cold-start procedures, underscoring its efficiency and robustness. Quantitatively, the framework achieves an improvement of 5.32 and 5.10 in Exact Match (EM) and LLM-based Evaluation (LLM) on the HotpotQA dataset, 11.90 and 7.18 on the MuSiQue dataset, and 15.45 and 18.60 on the 2Wiki dataset, relative to the current best baseline, reflecting its superior ability to generate precise and contextually appropriate query reformulations. Furthermore, our framework exhibits a learned capacity for multi-hop rewriting inference, effectively capturing dependencies across sequential query reformulations. This capability is highly generalizable, enabling RewriteGen to adapt to diverse query patterns and domains without task-specific retraining. These results establish RewriteGen as a state-of-the-art solution for query rewriting, offering both practical efficacy and theoretical insight into LLM-driven query optimization.

Table 3. Performance comparison of RewriteGen and baselines on multi-hop QA benchmarks. Best results are in **bold**.

	HotpotQA		MuSiQue		2Wiki	
Model	**EM**	**LLM**	**EM**	**LLM**	**EM**	**LLM**
Naive Generation	16.20	30.64	3.76	10.68	25.76	27.87
Standard RAG	28.75	49.59	6.21	12.78	23.78	29.52
Rewrite–Retrieve–Read	34.38	53.80	7.90	13.64	30.67	28.20
Iter-RetGen	31.36	52.22	6.69	6.69	24.92	27.92
Self-Ask	32.56	53.43	7.49	7.43	23.35	28.12
IRCoT	28.33	50.06	6.99	6.99	19.57	21.57
RewriteGen-Qwen2.5-3B (Our)	**39.70**	**58.90**	**19.80**	**20.82**	**46.12**	**48.12**

As shown in Figure 3, the scaling relationships of reward, response length, KL loss, and validation score are illustrated in subfigures (a), (b), (c), and (d), respectively. All metrics, including reward score, KL penalty, and validation score, steadily increase with the training steps, highlighting the stability of the RL training method.

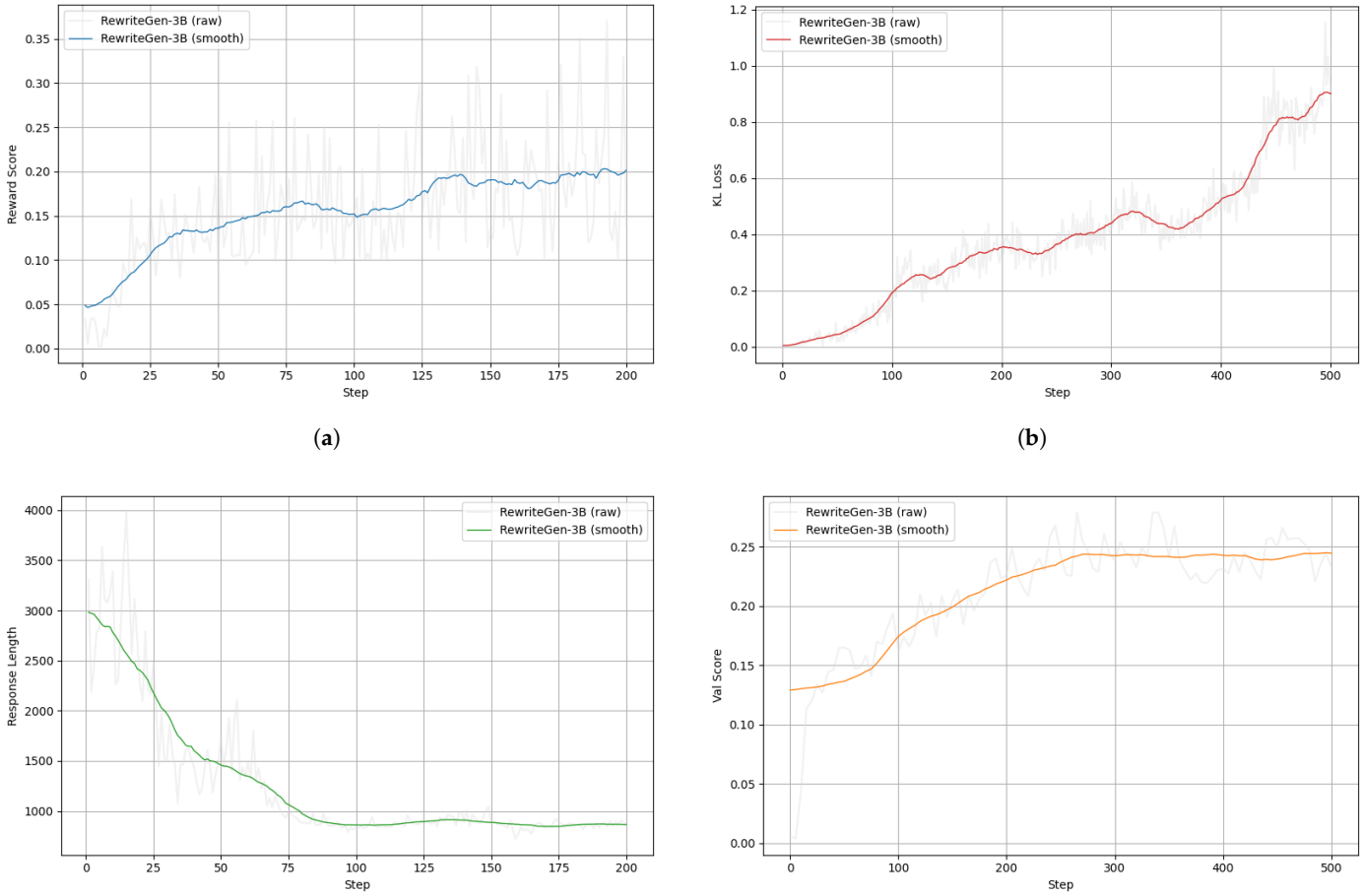


Figure 3. (**a**) The scaling relationship of reward, response length. (**b**) The scaling relationship of KL loss. (**c**) The scaling relationship of response length. (**d**) The scaling relationship of validation score.

Response length stability. As shown in Figure 3c, while the response length does not exhibit a clear inflection point (e.g., an "aha moment"), it gradually stabilizes over training, suggesting that the model's responses converge toward optimized patterns as training progresses.

Number of Rewrite Operations. Figure 4 presents the average number of rewrite labels during the model training process. We compute the average number of query rewrite operations per rollout during training. As shown in the figure, the frequency of rewrite operations exhibits steady growth throughout training, indicating that the model learns to iteratively refine queries for complex multi-hop questions, enhancing retrieval relevance.

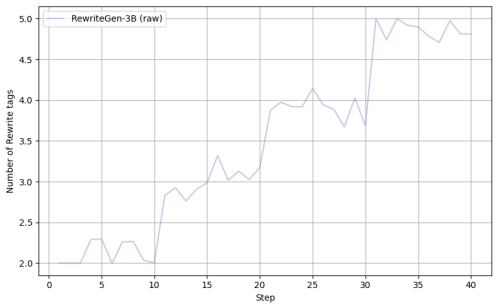


Figure 4. Average number of rewrite operations.

5. Ablation Study

To understand the contribution of each component in the ReWriteGen framework, we performed a detailed ablation study. This study aims to assess the impact of removing certain key elements, such as Reinforcement Learning (RL), the Reward mechanism, and Direct Preference Optimization (DPO), from the model. By systematically removing these components, we evaluate how their absence influences the overall performance of the framework.

We compare the performance of the full ReWriteGen model with several reduced versions, each excluding one of the key components. The models were evaluated on multi-hop QA benchmarks, including HotpotQA, MuSiQue, and 2Wiki, with the Qwen2.5-3B model as the base for all experiments. The results of these ablation experiments are summarized in Table 4.

Table 4. Ablation study of RewriteGen. The bold text indicates key models, red text represents the difference between the model's results and the baseline, and the blue background highlights the results of the proposed RewriteGen-Qwen2.5-3B.

	HotpotQA		MuSiQue		2Wiki	
Model	**EM**	**LJ**	**EM**	**LJ**	**EM**	**LJ**
RewriteGen (No RL)	36.50 (−3.20)	56.80 (−2.10)	19.20 (−0.60)	20.50 (−0.32)	44.90 (−1.22)	47.60 (−0.52)
RewriteGen (No Reward)	37.00 (−2.70)	57.20 (−1.70)	19.40 (−0.40)	20.34 (−0.48)	45.10 (−1.02)	47.80 (−0.32)
RewriteGen (No DPO)	38.00 (−1.70)	58.00 (−0.90)	19.10 (−0.70)	20.60 (−0.22)	45.16 (−0.96)	47.00 (−1.12)
RewriteGen-Qwen2.5-3B	**39.70**	**58.90**	**19.80**	**20.82**	**46.12**	**48.12**

Impact of Reinforcement Learning. When the RL phase is removed from the model, performance drops significantly across all datasets. Specifically, the Exact Match (EM) and LLM scores for ReWriteGen (No RL) show a reduction of 3.20 and 2.10 for HotpotQA, respectively. The lack of RL results in a failure to optimize the iterative query rewriting process effectively, leading to less accurate retrieval-augmented answers. Without RL, the model is unable to learn from the feedback provided by the task, thus impacting its ability to adapt queries dynamically based on previous responses. Reinforcement Learning is essential for enabling autonomous learning, allowing the model to adapt its queries and answers during the reasoning process, thereby enhancing the overall accuracy of the output.

Impact of the Reward Mechanism. Removing the Reward mechanism also leads to performance degradation, although less pronounced compared to removing RL. The Exact Match score drops by 2.70 on HotpotQA, and the LLM score decreases by 1.70. Without the reward function, the model is unable to effectively assess the quality of its answers or its query rewriting process. The reward mechanism, which evaluates the relevance and format of the answers, is critical for guiding the model towards high-quality and contextually appropriate responses. This highlights the critical role of structured feedback in guiding the model toward generating high-quality, relevant answers.

Impact of Direct Preference Optimization. The removal of DPO results in the smallest drop in performance compared to RL and the reward mechanism. However, there is still a noticeable decrease in Exact Match (1.70) for HotpotQA and LLM (0.90) for HotpotQA. DPO plays a crucial role in selecting the best possible answer by comparing multiple candidate answers from both retrieval results and generated content. The absence of DPO reduces the model's ability to autonomously fine-tune its answer selection, relying instead on simpler mechanisms to choose answers that may not always be the most relevant. While the absence of DPO results in a smaller performance decrease compared to RL and the reward mechanism, its role in comparing multiple candidate answers and selecting the

most appropriate one remains significant. Without DPO, the model's answer selection process becomes less refined, often leading to suboptimal answers.

Discussion of Ablation Results. The ablation study clearly demonstrates the importance of each component in ReWriteGen. Each part, including RL, the reward mechanism, and DPO, plays a vital role in optimizing the model's performance. The full model, which integrates all three components, consistently outperforms the reduced versions, demonstrating the effectiveness of our approach for query rewriting and retrieval-augmented generation. Reinforcement Learning enables autonomous learning, allowing the model to adapt queries and answers during the reasoning process, while the reward mechanism ensures that responses are accurate and contextually appropriate. DPO, though leading to a smaller performance drop, ensures that the most relevant and accurate answers are selected. These results also emphasize the importance of an end-to-end optimization approach that integrates query rewriting, retrieval, and answer generation seamlessly, ensuring superior performance on complex multi-hop question answering tasks.

6. Case Study Analysis

As shown in Table 5, the advantages of using this step-by-step method are clear. By breaking down the task into manageable steps—reflection, rewriting, retrieval, generation, and summarization—the approach ensures that each stage of the problem-solving process is handled thoughtfully and effectively. This process enhances the accuracy of the solution, reduces ambiguity, and provides a clear trail of reasoning that can be reviewed and understood.

Table 5. Case Study.

Question: Roger O. Egeberg was Assistant Secretary for Health and Scientific Affairs during the administration of a president that served during what years? **Answer:** 1969 until 1974
<reflect> We know that the role of Assistant Secretary for Health and Scientific Affairs was likely appointed under a U.S. president. Given the health-related nature of the role, this is probably tied to significant health reforms or policies during the administration of a specific president. I will start by searching for the years when Roger O. Egeberg held this position and determine which president was in office during that time. </reflect>
<rewrite> When did Roger O. Egeberg serve as Assistant Secretary for Health and Scientific Affairs? </rewrite>
<retrieve_result> Roger O. Egeberg served as Assistant Secretary for Health and Scientific Affairs under President Richard Nixon from 1969 to 1974. </retrieve_result>
<reflect> Now that I know the years (1969–1974), I should confirm whether there were any significant health policies or initiatives during this time to provide a complete answer. What was the president's role in health affairs during these years? </reflect>
<gen> During Nixon's presidency, Egeberg played a crucial role in overseeing health and scientific affairs, contributing to major public health initiatives, including efforts to combat air pollution, support for the development of vaccines, and the advancement of scientific research. </gen>
<result> Roger O. Egeberg served during a time when Nixon's administration enacted major health reforms, including the establishment of the EPA and NIOSH. These policies reflect the significant health reforms tied to his tenure.</result>
<answer> 1969 until 1974 **<**/answer>

This approach also demonstrates flexibility, as the model can be adapted to handle a wide range of questions and contexts. In addition, it allows for continuous learning, as each step can be refined based on the feedback from the previous stages. As a result, the system is able to autonomously generate and refine answers, making it more efficient and adaptable over time.

By adopting this approach, complex questions can be tackled in a structured manner, ensuring that the final answers are both accurate and well-informed. This method is

particularly effective in handling multi-hop reasoning tasks, where multiple pieces of information need to be gathered and synthesized. Overall, this step-by-step approach provides a robust framework for solving complex problems with clarity and precision.

7. Conclusions

Our experiments demonstrate that ReWriteGen significantly enhances the query rewriting capabilities of large language models (LLMs) through reinforcement learning (RL). The results across multiple benchmarks, including HotpotQA, MuSiQue, and 2Wiki, show that ReWriteGen outperforms traditional baselines, underscoring its effectiveness in handling complex multi-hop question answering tasks. The framework's ability to autonomously refine queries and integrate retrieved knowledge allows it to not only generate more accurate answers but also improve the efficiency of the reasoning process. The core innovation of ReWriteGen lies in its seamless integration of query rewriting, retrieval, and answer generation. By introducing an iterative, reinforcement-learning-based optimization process, the model learns to progressively improve its query formulation, which in turn enhances the relevance and quality of the retrieved information. This end-to-end optimization enables the model to tackle more nuanced and intricate questions, adapting to a variety of task-specific requirements without the need for additional fine-tuning or external supervision. Furthermore, the case study highlights the model's capacity for structured reasoning and iterative query reformulation, demonstrating the system's ability to handle dynamic reasoning processes. ReWriteGen naturally elicits advanced capabilities such as reflection during training, allowing the model to revisit and refine previous answers or queries based on new insights or data, thus improving the overall task performance.

These results validate the efficacy of our approach, showcasing ReWriteGen as a promising solution for advancing retrieval-augmented generation in complex, knowledge-intensive tasks. The framework not only significantly improves retrieval relevance and generation accuracy but also demonstrates scalability across different datasets and domains, suggesting its broad applicability to a variety of real-world problems. By reducing the dependency on task-specific fine-tuning and supervised data, ReWriteGen paves the way for more generalizable and efficient large language models that can seamlessly integrate retrieval and reasoning for complex question answering and other knowledge-intensive applications.

However, there are some limitations to this study. First, although the model performs well across multiple datasets, it is still largely dependent on the quality of the retrieval step. If the retrieved documents are not highly relevant, the performance of ReWriteGen can degrade, highlighting the challenge of integrating retrieval with reasoning in a fully autonomous manner. Second, while we have demonstrated the ability of ReWriteGen to improve accuracy, the framework's performance can still be influenced by the specific characteristics of the datasets and domains it is applied to. For example, when dealing with highly specialized domains or queries with sparse information, the model might struggle to generate effective query rewrites and optimize answer generation. Finally, the computational cost of reinforcement learning, especially in large-scale applications, remains a challenge. Although we have used efficient policy optimization techniques, the scalability of this approach for industrial-scale deployment needs further investigation.

In future work, we plan to address these limitations by exploring ways to improve retrieval quality, particularly in domain-specific settings. We also intend to optimize the reinforcement learning pipeline to reduce computational costs, making the framework more efficient for large-scale deployment. Additionally, incorporating user feedback during the learning process could enhance the model's adaptability, allowing it to perform better in dynamic, real-world environments.

Author Contributions: Conceptualization, Y.Z.; methodology, Y.Z.; software, Y.C.; validation, Y.Z., Z.F. and Y.C.; formal analysis, Y.Z.; investigation, Y.Z.; resources, Y.C.; data curation, Y.C.; writing—original draft preparation, Y.Z.; writing—review and editing, Y.Z., Z.F. and Y.C.; visualization, Y.C.; supervision, Z.L. and J.L.; project administration, Y.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The data supporting the reported results are not publicly available.

Conflicts of Interest: Author Zhengjia Lyu was employed by the company iFlytek Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Guo, D.; Yang, D.; Zhang, H.; Song, J.; Zhang, R.; Xu, R.; Zhu, Q.; Ma, S.; Wang, P.; Bi, X.; et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv* **2025**, arXiv:2501.12948.
2. Rafailov, R.; Sharma, A.; Mitchell, E.; Manning, C.D.; Ermon, S.; Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 53728–53741.
3. Robertson, S.; Zaragoza, H. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.* **2009**, *3*, 333–389. [CrossRef]
4. Thakur, N.; Reimers, N.; Rücklé, A.; Srivastava, A.; Gurevych, I. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv* **2021**, arXiv:2104.08663. [CrossRef]
5. Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.S.; Wu, L.; Edunov, S.; Chen, D.; Yih, W.-t. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the EMNLP (1), Online, 16–20 November 2020; pp. 6769–6781.
6. Xiong, L.; Xiong, C.; Li, Y.; Tang, K.F.; Liu, J.; Bennett, P.; Ahmed, J.; Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. *arXiv* **2020**, arXiv:2007.00808. [CrossRef]
7. Choi, Y.; Na, C.; Kim, H.; Lee, J.H. READSUM: Retrieval-augmented adaptive transformer for source code summarization. *IEEE Access* **2023**, *11*, 51155–51165. [CrossRef]
8. Santhanam, K.; Khattab, O.; Saad-Falcon, J.; Potts, C.; Zaharia, M. Colbertv2: Effective and efficient retrieval via lightweight late interaction. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA, 10–15 July 2022; pp. 3715–3734.
9. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
10. He, Z.; Zhong, Z.; Cai, T.; Lee, J.; He, D. Rest: Retrieval-based speculative decoding. In Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), Mexico City, Mexico, 16–21 June 2024; pp. 1582–1595.
11. Gao, L.; Ma, X.; Lin, J.; Callan, J. Precise zero-shot dense retrieval without relevance labels. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 1762–1777.
12. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. React: Synergizing reasoning and acting in language models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.
13. Press, O.; Zhang, M.; Min, S.; Schmidt, L.; Smith, N.A.; Lewis, M. Measuring and narrowing the compositionality gap in language models. In Proceedings of the Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, 6–10 December 2023; pp. 5687–5711.
14. Schaeffer, R.; Miranda, B.; Koyejo, S. Are emergent abilities of large language models a mirage? *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 55565–55581.
15. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D.; et al. Chain-of-thought prompting elicits reasoning in large language models. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24824–24837.
16. Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 1877–1901.
17. Asai, A.; Schick, T.; Lewis, P.; Chen, X.; Izacard, G.; Riedel, S.; Hajishirzi, H.; Yih, W.t. Task-aware retrieval with instructions. In Proceedings of the Findings of the Association for Computational Linguistics: ACL 2023, Toronto, ON, Canada, 9–14 July 2023; pp. 3650–3675.

18. Saad-Falcon, J.; Khattab, O.; Santhanam, K.; Florian, R.; Franz, M.; Roukos, S.; Sil, A.; Sultan, M.; Potts, C. UDAPDR: Unsupervised domain adaptation via LLM prompting and distillation of rerankers. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 11265–11279.
19. Zheng, H.S.; Mishra, S.; Chen, X.; Cheng, H.T.; Chi, E.H.; Le, Q.V.; Zhou, D. Take a step back: Evoking reasoning via abstraction in large language models. *arXiv* **2023**, arXiv:2310.06117.
20. Yu, W.; Iter, D.; Wang, S.; Xu, Y.; Ju, M.; Sanyal, S.; Zhu, C.; Zeng, M.; Jiang, M. Generate rather than retrieve: Large language models are strong context generators. *arXiv* **2022**, arXiv:2209.10063.
21. Ma, X.; Gong, Y.; He, P.; Zhao, H.; Duan, N. Query rewriting in retrieval-augmented large language models. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 5303–5315.
22. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]
23. Yang, H.; Li, Z.; Zhang, Y.; Wang, J.; Cheng, N.; Li, M.; Xiao, J. PRCA: Fitting black-box large language models for retrieval question answering via pluggable reward-driven contextual adapter. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, Singapore, 6–10 December 2023; pp. 5364–5375.
24. Jin, J.; Zhu, Y.; Dou, Z.; Dong, G.; Yang, X.; Zhang, C.; Zhao, T.; Yang, Z.; Wen, J.R. Flashrag: A modular toolkit for efficient retrieval-augmented generation research. In Proceedings of the Companion Proceedings of the ACM on Web Conference 2025, Sydney, Australia, 28 April 2025; pp. 737–740.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.