Query Optimization for Parametric Knowledge Refinement in Retrieval-Augmented Large Language Models

Anonymous ACL submission

Abstract

We introduce the *Extract-Refine-Retrieve-Read* (ERRR) framework, a novel approach designed to bridge the pre-retrieval information gap in Retrieval-Augmented Generation (RAG) systems through query optimization tailored to meet the specific knowledge requirements of Large Language Models (LLMs). Unlike conventional query optimization techniques used in RAG, the ERRR framework begins by extracting parametric knowledge from LLMs, followed by using a specialized query optimizer for refining these queries. This process ensures the retrieval of only the most pertinent information essential for generating accurate responses. Moreover, to enhance flexibility and reduce computational costs, we propose a trainable scheme for our pipeline that utilizes a smaller, tunable model as the query optimizer, which is refined through knowledge distillation from a larger teacher model. Our evaluations on various question-answering (QA) datasets and with different retrieval systems show that ERRR consistently outperforms existing baselines, proving to be a versatile and cost-effective module for improving the utility and accuracy of RAG systems.

1 Introduction

The field of natural language processing (NLP) has witnessed transformative advancements in recent years, largely driven by the advent of Large Language Models (LLMs). These models, trained on vast corpora, have demonstrated exceptional capabilities in understanding human text and generating high-quality responses (Kaplan et al., 2020; Clark et al., 2022). They have also proven practical and scalable for various downstream NLP tasks, such as conversational response generation, text summarization, and content recommendation, even in few-shot or zero-shot settings (Wu et al., 2023). Despite their strengths, a key limitation of LLMs lies in their reliance on static training data, which makes them struggle with dynamic or less commonly known information outside their initial training scope. This limitation often leads to outdated, inaccurate, or entirely fabricated responses—a phenomenon commonly referred to as “hallucination” (Lee et al., 2018).

To address this issue, Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) has emerged as a promising approach to enhance the functionality and reliability of LLMs. By integrating external knowledge sources through retrieval systems, RAG enables LLMs to augment user queries with relevant, up-to-date information. This augmentation allows LLMs to generate more contextually accurate and relevant responses. For instance, in a conversational setting where a user queries an LLM like ChatGPT (Ouyang et al., 2022) for the latest news, RAG retrieves pertinent articles to supplement the static pre-trained knowledge of the model, thereby mitigating the information gap.

While retrieval augmentation has proven effective in mitigating hallucinations, it introduces its own set of challenges. A prominent issue in Retrieval-Augmented Generation (RAG) systems is the **pre-retrieval gap**—a mismatch between the information retrieved using the original user query and the specific knowledge required to generate optimal responses (Gao et al., 2024). For instance, consider a document collection containing three passages, labeled Passage A, B, and C, each containing unique knowledge components x, y, and z, respectively. Although all three passages include keywords associated with Knowledge z—the user’s intended target—a poorly formulated query may lead to retrieving Passage A or B instead of the ideal Passage C. This misalignment restricts the LLM reader’s ability to generate accurate responses, making the pre-retrieval gap a critical barrier to achieving optimal text generation in RAG systems.

To bridge the pre-retrieval gap, the *Rewrite-*

Retrieve-Read (RRR) framework (Ma et al., 2023) introduced query rewriting as a mechanism to optimize user queries and improve their alignment with retrieval systems. However, RRR and similar methods (Zheng et al., 2024; Gao et al., 2024) primarily focus on rephrasing or broadening queries, which helps expand the search scope but fails to address the specific knowledge requirements of the LLM reader. Additionally, recent self-prompting techniques (Li et al., 2022; Wang et al., 2023) have explored using chain-of-thought (CoT) prompts and pseudo-QA pairs to enhance LLM reasoning capabilities by eliciting internal parametric knowledge. While these approaches effectively improve the internal reasoning and explanation capabilities of LLMs for tasks like multi-hop reasoning and open-domain QA, they lack mechanisms for aligning external retrieval queries with the LLM's knowledge gaps, making them insufficient for resolving the pre-retrieval gap in Retrieval-Augmented Generation (RAG) systems.

To this end, we propose *Extract-Refine-Retrieve-Read* (ERRR), a simple but effective framework designed for retrieval augmentation systems. The ERRR framework is crafted to bridge the pre-retrieval information gap through tailored query optimization and aims to resolve the inherent limitations of RRR by enabling retrieval based on the specific information needs of the LLM reader. Specifically, it initiates by extracting parametric knowledge from LLMs and employs a specialized query optimizer which refines user queries. This refinement either complements or validates the extracted parametric knowledge, ensuring that only essential information is retrieved for generating accurate responses, and minimizing the retrieval of extraneous information that could degrade the quality of the output.

In addition to its innovative query optimization process, ERRR introduces a trainable scheme to enhance efficiency and adaptability. Recognizing the constraints posed by black-box systems like ChatGPT (Ouyang et al., 2022), which are accessible only through inference APIs, ERRR incorporates a smaller, tunable language model as the query optimizer. This trainable component reduces computational costs while offering greater flexibility to customize the retrieval process for diverse queries and knowledge sources. By combining precision in addressing pre-retrieval gaps with cost-effective adaptability, ERRR provides a robust solution for improving retrieval augmentation in LLM-driven systems.

We evaluate ERRR on multiple question-answering (QA) datasets, including HotpotQA (Yang et al., 2018), AmbigNQ (Min et al., 2020), and PopQA (Mallen et al., 2022), using both web-based (e.g., Brave Search Engine) and local retrieval systems (e.g., Dense Passage Retrieval (Karpukhin et al., 2020)). Across all tested datasets and retrieval configurations, ERRR consistently outperforms baseline frameworks, such as RRR, in terms of retrieval accuracy and response quality. These results highlight ERRR's versatility and effectiveness in diverse settings.

In summary, our key contributions are as follows: (i) We propose *Extract-Refine-Retrieve-Read* (ERRR), a novel framework that optimizes queries to bridge the pre-retrieval gap and enhance RAG systems. (ii) We demonstrate ERRR's adaptability across different datasets, retrieval systems, and settings, establishing its robustness and versatility. (iii) We introduce a trainable ERRR scheme that reduces computational costs while maintaining high performance, making it suitable for real-world applications.

2 Related work

2.1 Retrieval-Augmented Generation

The integration of retrieval modules to access relevant contextual knowledge has played a crucial role in enhancing Large Language Models (LLMs) in recent years. Initially designed for early sequence-to-sequence models, the Retrieval-Augmented Generation (RAG) framework proposed by Piktus et al. (Lewis et al., 2020) has gained substantial traction in the era of LLM. This approach has diversified into a broad array of methods, with ongoing efforts aimed at further augmenting its capabilities. Earlier exploration primarily focused on improving key components, such as upgrading to more powerful pre-trained language models like BERT (Devlin et al., 2019) as readers or employing advanced dense retrievers for retrieval tasks (Karpukhin et al., 2020). These retrievers encode documents and inputs into dense vectors, facilitating retrieval based on the similarity between the input and retrieved passages.

Recent studies have shifted focus beyond merely enhancing the retriever or reader components, emphasizing the refinement of pre-retrieval and post-retrieval processes. To address the pre-retrieval gap—the disparity between the information retrievable from original queries and the knowledge required for optimal responses—GenRead (Yu et al., 2023) replaces the retrieval module with a knowledgeable LLM, thereby narrowing the gap between the user query and retrieval process. It prompts the LLM to generate contextual information for the query, using these generated documents as retrieval results to formulate the final answer. Self-ask (Press et al., 2023) proposes an iterative approach using chain-of-thought prompting to generate self-posed questions that refine the response. For the post-retrieval gap—the challenge of creating optimal responses from given information—strategies include document re-ranking or summarization. For instance, PRCA (Yang et al., 2023) trains a contextual adaptor module to summarize retrieved documents with a black-box LLM reader.

Several studies have also proposed significant modifications to the original RAG pipeline, introducing complex systems that include both pre-retrieval and post-retrieval modules (Rackauckas, 2024), and adapting the pipeline into iterative or recursive frameworks (Yao et al., 2022; Asai et al., 2023). While these advanced systems demonstrate notable performance enhancements, they incur substantial costs and typically require multiple interactions with LLM. In contrast, our work focuses on refining the single-turn RAG framework, introducing a flexible and trainable module adaptable to existing systems.

2.2 Query Optimization for Retrieval Augmentation

Recent research highlights a significant discrepancy between input queries and LLM readers for RAG systems, especially under the current trend of using off-the-shelf web search tools or black-box LLMs that are difficult to customize (Ma et al., 2023). Typically, these input queries often originate directly from users or specific datasets, which could be either poorly formulated or adhere to a static query format. To overcome these challenges, an effective approach is to optimize the query in the pre-retrieval phase, thereby improving the quality of retrieved information and response generation. The *Rewrite-Retrieve-Read* (RRR) framework, for instance, trains a query rewriting module using an LLM to better align retrieval queries with LLM readers (Ma et al., 2023) that generate the final response, as illustrated in Figure 1. Additionally, RRR introduces a trainable scheme that employs reinforcement learning with Proximal Policy Optimization to fine-tune a small open-source model based on feedback from the LLM reader, achieving improved results. HyDE addresses the demand for accurate information retrieval by creating hypothetical documents and encoding them through unsupervised contrastive learning for efficient retrieval operations (Gao et al., 2023). Furthermore, Step-Back Prompting (Zheng et al., 2024) converts original queries into high-level abstract questions, aiding LLMs in generating better responses for complex queries requiring abstract thinking.

While these efforts have markedly improved the performance of original RAG systems by focusing on query optimization, they often overlook the importance of synchronizing queries with the specific knowledge requirements of the LLM reader. Unlike the RRR framework, our approach includes an additional parametric knowledge extraction step to assess the knowledge possessed by the LLM. We then perform retrieval based on optimized queries to refine this parametric knowledge, thereby further enhancing retrieval-augmented LLMs.

3 Methodology

In this section, we elaborate on the details of Extract-Refine-Retrieve-Read (ERRR), a framework for improving the retrieval-augmented LLMs through query optimization for parametric knowledge refinement. Section 3.1 formally defines the central task addressed by ERRR and introduces its key concepts. The design of the framework is discussed in Section 3.2, where we outline a frozen scheme using a black-box LLM reader and standard web search tools. Additionally, Section 3.3 discusses a trainable scheme of the framework.

3.1 Pre-retrieval Information Gap

A task with retrieval augmentation can be formulated as follows. Given an input query q, a set of theoretical golden documents D that has the accurate information to answer query q, and a ground-truth answer a, we denote:

$$\mathrm{LLM}(D, q \mid \theta) = a \tag{1}$$

where LLM denotes an LLM reader and θ denotes the parametric knowledge of the LLM.

However, to obtain the document set D, practical implementations often employ a retrieval function R which retrieves documents $R(q)$ from an external knowledge base, and thus the output of a

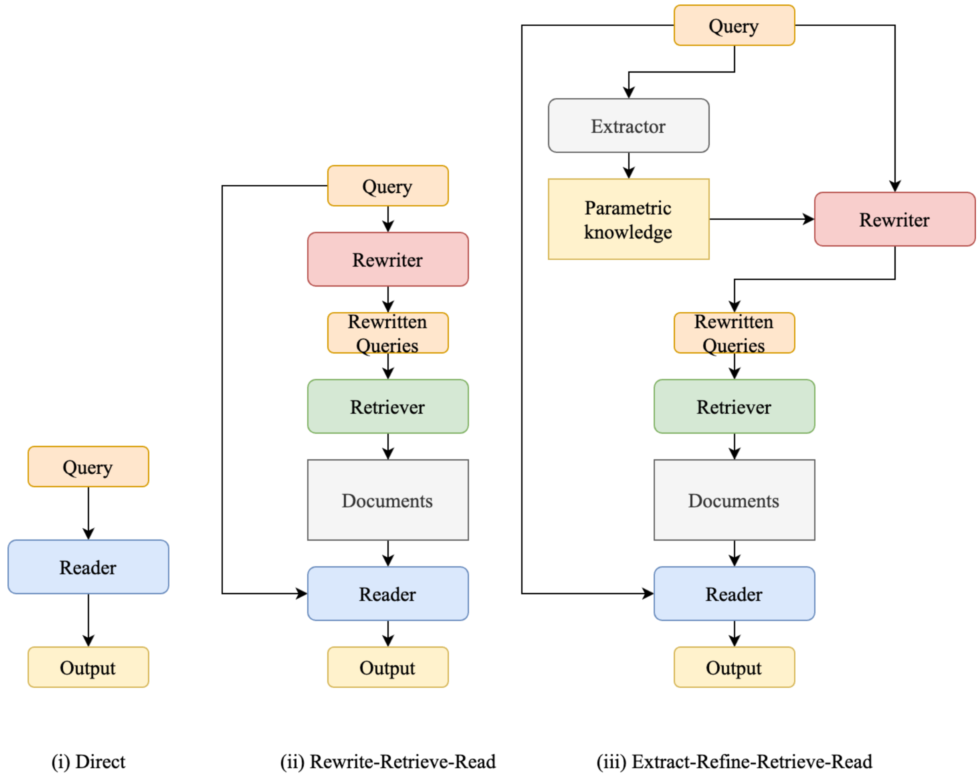

Figure 1: Overview of *Extract-Refine-Retrieve-Read* (ERRR). Extract-Refine-Retrieve-Read leverages parametric knowledge of LLMs and utilizes a specialized query optimizer to retrieve the knowledge that better aligns with LLM's needs.

retrieval-augmented system is:

$$\mathrm{LLM}(R(q), q \mid \theta) \tag{2}$$

An inherent challenge arises due to the difference in the quality and relevance of documents retrieved by R compared to the ideal documents set D:

$$\mathrm{LLM}(R(q), q \mid \theta) \neq \mathrm{LLM}(D, q \mid \theta) \tag{3}$$

The limitation discussed above describes the problem of the pre-retrieval gap in the original RAG pipeline, wherein the set $R(q)$ may not adequately represent the information necessary for generating the true answer a. Therefore, the main objective is to develop a query optimization function f that transforms the initial user query q into one or more optimized queries $f(q)$ such that $R(f(q))$ better approximates the ideal document set D.

Previous work like RRR (Ma et al., 2023) has demonstrated the effectiveness of such query optimization functions, albeit without considering the influence of θ. To this end, ERRR introduces a more tailored query optimization function f' that utilizes the parametric knowledge θ to perform the query optimization and retrieve external knowledge that refines θ and better aligns with its needs. This can be formulated as:

$$\mathrm{LLM}(R(f'(C, q)), q \mid \theta) \tag{4}$$

where

$$C = E(q \mid \theta)$$

and E denotes the parametric knowledge extraction function.

3.2 Extract-Refine-Retrieve-Read

Extract-Refine-Retrieve-Read consists of a four-step pipeline: Parametric Knowledge Extraction, Query Optimization for Parametric Knowledge Refinement, Retrieval, and Generation, as depicted in Figure 1. Detailed technical implementation for each step, covering the models, prompting techniques and training setup, is provided in Section 4.3.

Parametric Knowledge Extraction Previous studies such as GenRead (Yu et al., 2023) and HyDE (Gao et al., 2023) demonstrate that LLMs may possess substantial parametric knowledge capable of addressing user inquiries, particularly on popular topics. Inspired by the prompting methods outlined in GenRead, our approach involves a direct strategy where we prompt the LLM reader to produce a pseudo-contextual document containing all the background information. We consider these pseudo-contextual documents as a representation of the LLM's abstracted parametric knowledge. Although these documents may contain inaccuracies, they provide essential contextual information related to the original queries.

Query Optimization In this step, we employ an LLM as the query optimizer for parametric knowledge refinement. We prompt the query optimizer to produce one or more optimized queries seeking external knowledge that either validates or supplements the existing parametric knowledge, especially focusing on the validation of time-sensitive information.

Retrieval To illustrate the adaptability of our module across various retrieval systems and data sources, we utilize two types of retrievers: a black-boxed web search tool and a local dense retrieval system, which are then combined with the original query for processing by the LLM reader.

Generation We employ an LLM reader to generate the final answer using both the retrieved documents and the original query. Our prompting strategy involves straightforward instruction followed by 1-3 few-shot examples for question answering. These examples are consistently used within each dataset but vary across different datasets to maintain control over the task-specific output format from the LLM reader—for instance, the responses are expected to be concise in certain QA tasks, usually only one or a few words.

3.3 Trainable Scheme

Given that many powerful LLMs operate as black-box systems, significant challenges such as high computational costs, customization limitations, copyright issues, and connectivity problems have arisen. To address these issues, alongside the conventional frozen scheme, we propose a trainable scheme for our pipeline. Specifically, we fine-tuned a smaller, trainable model utilizing knowledge distillation from a high-performing teacher LLM, leveraging its broadly trained outputs as a good starting point and learning template, and intensively training student models on a distillation dataset of QA questions and generated responses to learn the intricate nuances of query optimization. This streamlined model is then integrated into our pipeline to fulfill the role of query rewriting, originally handled by a frozen LLM.

4 Experiments

4.1 Datasets and Metrics

ERRR is assessed on three open-domain question-answering (QA) datasets: AmbigQA (Min et al., 2020), PopQA (Mallen et al., 2022), and HotpotQA (Yang et al., 2018). Each dataset serves to test different capabilities of the ERRR framework. (i) The AmbigNQ dataset is the disambiguated variant of the Natural Questions (NQ) dataset, where ambiguous questions from NQ are refined into specific queries with minimal constraints. Consistent with procedures used in RRR, we evaluated ERRR using the first 1000 samples of the test set. (ii) PopQA features simpler questions that focus on less popular knowledge topics compared to other QA tasks. Due to the high similarity in sample distributions, we assessed only the first 997 samples of the test set. (iii) The HotPotQA dataset contains complex questions that require multi-hop reasoning. We conducted evaluations across the entire test set. Following the metric usage for three datasets, our method is evaluated by exact match score EM and F_1 score.

4.2 Baselines and Proposed Frameworks

We evaluated 7 baselines and proposed frameworks, as detailed below: (i) **Direct**: Directly calling GPT-3.5-Turbo to answer questions. (ii) **RAG**: The classic Retrieval-Augmented Generation framework (Lewis et al., 2020). The original user queries are used for retrieval and fed directly to the LLM reader to generate output. (iii) **ReAct**: A modified RAG framework that intertwines the reasoning and acting capabilities of LLMs to create a more cohesive and effective approach (Yao et al., 2022). This framework can iteratively perform reasoning prompts and actions, such as information retrieval, serving as our comparison baseline. (iv) **Frozen RRR**: *Rewrite-Retrieve-Read* framework (Ma et al., 2023) with a frozen configuration. It employs GPT-3.5-Turbo to rewrite the query and retrieve relevant documents based on these rewritten queries. Then the original query and retrieved documents are used for reading. This serves as our baseline for comparison. (v) **Trainable RRR**: Trainable *rewrite-retrieve-read* framework, initiating with a supervised fine-tuned T5-large model. It then applies reinforcement learning to better align the retriever and rewriter using Proximal Policy Optimization (PPO). This serves as our baseline for comparison. (vi) **Frozen ERRR**: *Extract-Refine-*

Direct Prompt
Answer the question in the following format, end the answer with '**'. {demonstration} Question: $\{x\}$ Answer:
Reader Prompt for Retrieval Augmentation Generation
Answer the question in the following format, end the answer with '**'. {demonstration} Question: $\{doc\}$ $\{x\}$ Answer:
Prompt for RRR Query Rewriter
Think step by step to answer this question, and provide search engine queries for knowledge that you need. Split the queries with ';' and end the queries with '**'. {demonstration} Question: $\{x\}$ Answer:
Prompt for Parametric Knowledge Extraction
Generate a background document from web to answer the given question. $\{x\}$
Prompt for ERRR Query Optimizer
Address the following questions based on the contexts provided. Identify any missing information or areas requiring validation, especially if time-sensitive data is involved. Then, formulate several specific search engine queries to acquire or validate the necessary knowledge. Split the queries with ';' and end the queries with '**'. {demonstration} Context: {Parametric Knowledge} Question: $\{x\}$ Queries:

Table 1: List of Prompts Used.

Retrieve-Read framework with a frozen configuration, as described in Section 3.2. (vii) **Trainable ERRR**: Trainable *Extract-Refine-Retrieve-Read* framework, as described in Section 3.3.

These frameworks are evaluated using a web search tool or a local retriever with a static corpus, as described in Section 3.2. Due to resource limitations, some frameworks were not evaluated under the local dense retriever setting.

4.3 Implementation Details

For all baselines, we utilized GPT-3.5-Turbo as the primary LLM and adhered to their implementation from the original paper. GPT-3.5 Turbo was chosen for its balance of performance and cost, aligning with our focus on optimizing retrieval-augmented generation systems rather than benchmarking generative models themselves. While GPT-4 offers improved capabilities, our emphasis remained on augmenting the model's utility through query optimization. Notably, for the Trainable RRR, we employed the supervised fine-tuned T5 model checkpoint as the base model. This checkpoint, open-sourced by the original authors, has been warmed up and fine-tuned on multiple datasets to function as the query rewriter. Then we replicated their reinforcement learning process since we replaced the original search tool with the Brave Search Engine. These trainings were conducted on the first 1000 data points for each dataset evaluated, with The training parameters set as follows: a learning rate of 2e-5, 3 epochs, and a batch size of 8.

For our proposed methods ERRR, in addition to the settings mentioned in Section 3.2, the following sections outline technical details:

Parametric Knowledge Extraction To perform parametric knowledge extraction, we use the same prompts from the GenRead paper and choose the top prompt that is most likely to produce pseudo-contextual documents. We outline these extraction prompts in Table 1.

Query Optimization Our specific prompt structure is detailed in Table 1, where demonstration consists of 2 manually crafted examples. These examples are consistently used across all tests and primarily serve as one or few-shot examples for the query optimizer.

Retrieval For our web search engine, we opt for the Brave Search Engine, which, although it may provide slightly lower quality results compared to major competitors like Google or Bing, offers a significantly more cost-effective API. This search API retrieves website snippets, simulating a typical user experience of entering a query in a search engine, pressing Enter, and reviewing the top results at a glance. For local retrieval, we utilize WikiDPR, a specialized subset of Wikipedia collections tailored for the Dense Passage Retrieval (DPR) model (Karpukhin et al., 2020). This database consists of 21 million passages from Dec. 20, 2018, each limited to 100 words, along with their 768-dimensional embedded vectors. The retrieval process involves converting a query into a DPR embedding and finding the top k vectors with the closest L2 distances. For both systems, we retrieve the top 5 results, con-

	AmbigQA		PopQA		HotPotQA	
Methods	EM	F1	EM	F1	EM	F1
Direct	0.391	0.4996	0.392	0.4289	0.311	0.4178
RAG	0.473	0.5842	0.425	0.4704	0.329	0.4424
ReAct	0.477	0.5787	0.451	0.4917	0.344*	0.4649*
Frozen RRR	0.452	0.5577	0.445	0.4904	0.337	0.4567
Trainable RRR	0.460	0.5577	0.389	0.4238	0.337	0.4548
Frozen ERRR	0.4815	0.5823	0.480	0.5256	0.369	0.4941
Trainable ERRR	**0.4975**	**0.5988**	**0.485**	**0.5309**	**0.372**	**0.4989**

Table 2: The retrieval system in the above methods is Brave Search API. "Frozen" indicates the rewriter or the query optimizer is GPT-3.5-Turbo, while "Trainable" refers to the rewriter or the query optimizer is a supervised fine-tuned T5 model. Trainable RRR is also trained using proximal policy optimization (PPO) following the original paper. '*' indicates that it is evaluated on 500 random questions drawn from HotPotQA due to resource limitation.

catenate them with the original query, and feed them to the LLM reader.

Generation Although different prompting strategies may influence the performance of the question-answering task, this aspect is not the primary focus of our study, so we adhere to the same answer prompts used in the RRR (Ma et al., 2023) framework. The prompts we used are detailed in Table 1.

Trainable Scheme For Trainable ERRR, we employ T5-Large (Raffel et al., 2020), an open-source model with 770 million parameters, as the query optimizer. We fine-tune this student model using knowledge distillation from GPT-3.5-Turbo. The distillation dataset was assembled by selecting questions from training sets of each QA dataset, with GPT-3.5-Turbo generating the responses under identical settings utilized in the frozen scheme. We also devised a short eliciting prompt, "Rewrite better search queries to acquire or validate the knowledge needed for the question:", serving as an instruction prefix to guide T5 to adapt to the task. To ensure optimal task-specific outcomes, separate T5 models were trained with 3 epochs for each QA dataset, with a learning rate of 1e-4 and a batch size of 4.

4.4 Result

The experimental results across three datasets and two retrieval tools are presented in Table 2 and Table 3. The Frozen ERRR framework consistently outperforms all baseline methods—Direct, Frozen RRR, and Trainable RRR—regardless of the retrieval system used. These results highlight the effectiveness of addressing the pre-retrieval information gap, demonstrating ERRR's adaptability across diverse retrieval systems and datasets. Furthermore, the Trainable ERRR framework achieves even better performance, surpassing all baselines and its teacher model (GPT-3.5 Turbo) across all three datasets. We attribute this improvement to the distillation process, which enables the student model (fine-tuned T5) to generalize better by focusing on critical features while filtering out irrelevant information. This distilled representation allows the model to adapt more effectively to specific query optimization tasks, potentially compressing and refining the teacher's insights into a more efficient form.

The impact of the ERRR framework is more pronounced in web search retrieval systems, as evidenced by the greater performance enhancement observed in Table 2 compared to dense retrievers in Table 3. This is likely due to the higher quality and broader knowledge span of web-based retrieval systems compared to the static 2018 Wikipedia corpus used for dense retrieval. Notably, the results show that both Frozen RRR and Trainable RRR underperform the Direct method in the PopQA and HotPotQA datasets when using dense retrieval. This underperformance can be attributed to the low-quality results retrieved from the outdated and limited corpus, which includes only Wikipedia passages of constrained length and scope. These limitations lead to an increased retrieval of irrelevant documents, distracting the Large Language Model (LLM) from answering questions correctly.

In contrast, ERRR demonstrates resilience under such conditions. By optimizing queries to align with the LLM's informational needs, ERRR reduces the retrieval of irrelevant passages, mitigating distractions caused by lower-quality retrieval. This robustness is particularly valuable when operating on suboptimal document collections, as it

	AmbigQA		PopQA		HotPotQA	
Methods	EM	F1	EM	F1	EM	F1
Direct	0.391	0.4996	0.392	0.4289	0.311	0.4178
Frozen RRR	0.438	0.5373	0.378	0.4517	0.289	0.3926
Trainable RRR	0.414	0.5203	0.365	0.4242	0.282	0.3764
Frozen ERRR	0.448	0.5473	0.419	0.4685	0.337	0.4482
Trainable ERRR	**0.4595**	**0.5577**	**0.426**	**0.4694**	**0.338**	**0.4499**

Table 3: Evaluations with WikiDPR as local retrievers. The other setting is the same as Table 2. Due to resource limitations, some baselines were not fully evaluated under this setting.

	Frozen ERRR	Trainable ERRR	ReAct	Self-RAG
Cost	\$0.62	\$0.53	\$1.05	\$1.65
Latency	148s	140s	202s	270s

Table 4: The total cost and total latency of each method that is evaluated on 200 randomly drawn data points from HotPotQA.

ensures performance gains even in challenging retrieval scenarios. A detailed case study, provided in Appendix A, further illustrates how ERRR generates precise queries that enhance retrieval effectiveness and improve final answers, even when the retrieved content includes inaccuracies.

4.5 Cost and Latency

Given our method's emphasis on a conventional single-turn pipeline, it demonstrates superior performance in terms of cost and latency when compared to certain advanced and iterative RAG frameworks. To underscore the cost-efficiency and flexibility of our approach, we conducted a comparative analysis with ReAct (Yao et al., 2022) and Self-RAG (Asai et al., 2023). This experiment was carried out on 200 randomly selected questions from HotPotQA. The results presented in Table 4 highlight that while still maintaining commendable performance, Frozen ERRR exhibits significantly lower costs, faster processing times, and greater efficiency than other iterative frameworks. Moreover, Trainable ERRR has the potential to further reduce costs, particularly for large datasets, by leveraging an already fine-tuned query optimizer, thereby saving on an additional LLM call to GPT-3.5-Turbo.

5 Conclusion

In this paper, we present *Extract-Refine-Retrieve-Read* (ERRR) framework for Retrieval-Augmented Generation (RAG) systems. The ERRR framework is designed to optimize queries, aligning them closely with the specific informational needs of large language models (LLMs) to enhance retrieval augmentation effectiveness. Our experimental results demonstrate that our method surpasses both the naive LLM and native query rewriting framework *Rewrite-Retrieve-Read* on benchmark datasets such as AmbigQA (Min et al., 2020), PopQA (Mallen et al., 2022) and HotPotQA (Yang et al., 2018), utilizing both web search tools and a dense retriever with local static corpus. It demonstrated ERRR's remarkable adaptability across a variety of settings, data sources, and retrieval systems. This flexibility ensures that ERRR can be effectively implemented in diverse operational environments, making it a potential and adaptable component for inclusion in more advanced RAG systems. Additionally, we have developed and implemented a trainable scheme for the ERRR framework. This approach is both cost-effective and efficient as it relies on only a fine-tuned T5 model trained on a moderately sized dataset and surpasses the performance of the frozen GPT-3.5-Turbo.

6 Limitation

We acknowledge that we recognize the existence of more sophisticated Retrieval-Augmented Generation (RAG) approaches such as Self-RAG(Asai et al., 2023) and CRAG(Yan et al., 2024). These advanced systems typically require iterative invocations of the entire pipeline to refine their answers, resulting in exceptionally high computational demands. Due to computational constraints within our study, we focused solely on scenarios that operate in a single-turn manner, wherein each module is invoked only once.

Additionally, our model does not employ any reinforcement learning techniques to enhance the performance of the supervised fine-tuned model. This decision was driven by resource limitations and observed sub-optimal performance when training with a small portion of the dataset using Proximal Policy Optimization (PPO) (Schulman et al., 2017), which constrained the potential upper limit of our model's performance.

For the future development of this work, while the ERRR framework addresses the pre-retrieval gap problem, future work could extend to methods that bridge the post-retrieval gap or incorporate ERRR into more advanced and modular RAG systems to further enhance performance in question-answering tasks. Furthermore, exploring new Reinforcement Learning (RL) algorithms to improve the query optimizer's performance for specialized tasks is also a possible direction for further exploration.

References

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2023. Self-rag: Learning to retrieve, generate, and critique through self-reflection. *Preprint*, arXiv:2310.11511.

Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. 2022. Unified scaling laws for routed language models. In *International conference on machine learning*, pages 4057–4086. PMLR.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. 2023. 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)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024. Retrieval-augmented generation for large language models: A survey. *Preprint*, arXiv:2312.10997.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online. Association for Computational Linguistics.

Katherine Lee, Orhan Firat, Ashish Agarwal, Clara Fannjiang, and David Sussillo. 2018. Hallucinations in neural machine translation.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Junlong Li, Jinyuan Wang, Zhuosheng Zhang, and Hai Zhao. 2022. Self-prompting large language models for zero-shot open-domain qa. *arXiv preprint arXiv:2212.08635*.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting in retrieval-augmented large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5303–5315, Singapore. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. *arXiv preprint arXiv:2212.10511*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. *arXiv preprint arXiv:2004.10645*.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A. Smith, and Mike Lewis. 2023. Measuring and narrowing the compositionality gap in language models. *Preprint*, arXiv:2210.03350.

Zackary Rackauckas. 2024. Rag-fusion: A new take on retrieval augmented generation. *International Journal on Natural Language Computing*, 13(1):37–47.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

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

Jinyuan Wang, Junlong Li, and Hai Zhao. 2023. Self-prompted chain-of-thought on large language models for open-domain multi-hop reasoning. *arXiv preprint arXiv:2310.13552*.

Junchao Wu, Shu Yang, Runzhe Zhan, Yulin Yuan, Derek F Wong, and Lidia S Chao. 2023. A survey on llm-gernerated text detection: Necessity, methods, and future directions. *arXiv preprint arXiv:2310.14724*.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*.

Haoyan Yang, Zhitao Li, Yong Zhang, Jianzong Wang, Ning Cheng, Ming Li, and Jing Xiao. 2023. 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*, pages 5364–5375, Singapore. Association for Computational Linguistics.

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

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. *Preprint*, arXiv:2209.10063.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed H. Chi, Quoc V Le, and Denny Zhou. 2024. Take a step back: Evoking reasoning via abstraction in large language models. *Preprint*, arXiv:2310.06117.

A Case Study

To explicitly and intuitively demonstrate the effectiveness of the ERRR compared to the RRR framework, we present two examples in Table 5 comparing their rewritten queries and final outputs. In the first example, the original question is *Stories USA starred which actor and comedian from "The Office"?*. The query rewriter in RRR framework produces a simplified query, *actor comedian "The Office" Stories USA cast*, which merely reformulates the original question for clearer web searching. In contrast, the ERRR not only answers correctly in the Parametric Knowledge Extraction phase but also generates refined queries such as *"actor and comedian from "The Office" in Stories USA"* and *"Steve Carell role in Stories USA"*, These queries not only attempt to validate the actor name of the T̈he Officeb̈ut also attempt to validate the name *Steve Carell* from the parametric knowledge, enabling the retriever to source better results.

In the second example, the rewritten query from RRR, *Pakistani actor writer Islamabad Coke Kahani 2012*, rewrites into only a few random keywords from the original question, which fails to facilitate a high-quality search. On the other hand, the first rewritten query from ERRR, *Pakistani actor and writer from Islamabad who helped write for Coke Kahani*, provides a clearer and more comprehensible query for search possibly inspired by the contextual information from the extracted parametric knowledge. The second rewritten query, *Faisal Rehman contributions to Coke Kahani* aims to verify the name derived from parametric knowledge, specifically Faisal Rehman. Interestingly, even though the name is incorrect, the information retrieved subsequently clarifies that Faisal Rehman is not the correct actor and writer, which effectively rectifies the LLM's output. Together with the information gathered from the first query, this leads to a correct final answer. This example illustrates that even if the pseudo-contextual document contains inaccuracies, the ERRR framework, by concentrating on the specific needs of the LLM reader, can still retrieve the most useful information for the LLM reader which results in a correct outcome.

Example 1
Question: Stories USA starred which actor and comedian from "The Office"? Answer: Steven John Carel RRR rewritten query: "actor comedian "The Office" Stories USA cast" Output: "Ricky Gervais" (*incorrect*) Parametric Knowledge Extraction: Stories USA starred actor and comedian Steve Carell from "The Office." **Steve Carell** is best known for his role as Michael Scott on the hit TV show "The Office," but he has also appeared in a number of films and other television shows. In Stories USA, Carell plays a man who is struggling to come to terms with his past and find his place in the world. The film is a collection of short stories that explore different aspects of the American experience, and Carell's performance is praised for its depth and emotional resonance. Overall, Stories USA is a thought-provoking and engaging film that showcases Carell's talent as both a comedic and dramatic actor. ERRR rewritten queries: "actor and comedian from "The Office" in Stories USA" "Steve Carell role in Stories USA" Output: "Steven John Carell" (*correct*)
Example 2
Question: What Pakistani actor and writer from Islamabad helped write for the 2012 Pakistani comedy drama sitcom, "Coke Kahani"? Answer: Yasir Hussain RRR rewritten query: "Pakistani actor writer Islamabad Coke Kahani 2012" Output: "Ali Abbas"(*incorrect*) Parametric Knowledge Extraction: One of the Pakistani actors and writers from Islamabad who helped write for the 2012 Pakistani comedy drama sitcom, "Coke Kahani," is **Faisal Rehman**. Faisal Rehman is a talented actor, writer, and director in the Pakistani entertainment industry. He has appeared in numerous television dramas and films, showcasing his versatility and acting skills. In addition to his acting career, Faisal Rehman has also written scripts for various television shows, including "Coke Kahani," which was well-received by audiences for its humor and engaging storyline. ERRR rewritten queries: "Pakistani actor and writer from Islamabad who helped write for Coke Kahani" "Faisal Rehman contributions to Coke Kahani" Output: "Yasir Hussain"(*correct*)

Table 5: Case Study from RRR and ERRR framework.