When and How to Augment Your Input: Question Routing Helps Balance the Accuracy and Efficiency of Large Language Models

Shufan Chen[1,2] **, He Zheng**[2*] **and Lei Cui**[2]
[1]University of Science and Technology of China
[2]Shanghai AI Laboratory
shufanchen@mail.ustc.edu.cn
{zhenghe, cuilei}@pjlab.org.cn

Abstract

Although large language models rely on parametric knowledge to achieve exceptional performance across various question-answering tasks, they still face challenges when addressing knowledge-based long-tail questions. Augmented generation techniques, such as chain-of-thought prompting and retrieval augmentation, can effectively enhance the ability of these models to answer long-tail questions. However, improving accuracy through augmented generation often results in significant latency within question-answering systems. This paper addresses the issue of "when and how to augment the input" by proposing an adaptive question routing framework. This framework employs a query router to select the most appropriate augmentation path at the right time, thereby enhancing both the accuracy and efficiency of question-answering systems. Extensive comparative experiments on benchmarks such as AmbigNQ, HotpotQA, MMLU-STEM, and PopQA demonstrate that our method surpasses existing approaches in both accuracy and efficiency. Furthermore, this paper introduces two metrics for evaluating adaptive question augmentation methods and presents a new benchmark for adaptive question augmentation, aiming to advance the field.

1 Introduction

Since the release of ChatGPT by OpenAI, we have witnessed the remarkable achievements of large language models in the field of Natural Language Processing (NLP) (Brown et al., 2020)(OpenAI, 2023). These models have demonstrated impressive competitiveness in various NLP tasks, such as intent recognition, entity answering, and reading comprehension. However, their hallucination problem limits their applicability in certain critical areas (Zhou et al., 2021). This issue is comparable to an overly enthusiastic assistant who, despite possessing extensive knowledge, refuses to acknowledge new information. The assistant is overly familiar with the trained corpus and question patterns and confidently responds to every user query, regardless of whether it truly understands the answer or processes the query correctly. This behavior contributes to the hallucination issue in large language models.

[*]Corresponding author

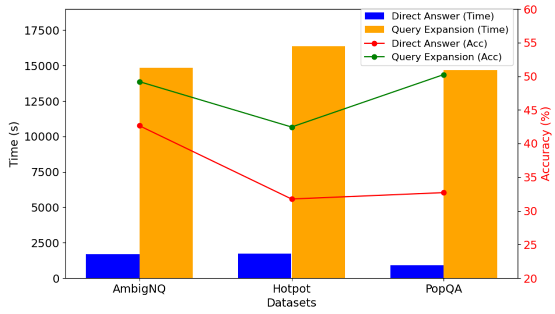

Figure 1: Performance comparison using GPT-3.5 across AmbigNQ, Hotpot, and PopQA datasets. Query expansion (orange bars) takes nearly ten times longer than direct answers (blue bars) but achieves only a moderate increase in accuracy (green vs. red lines).

Research indicates that appropriately reformulating queries plays a crucial role in reducing hallucinations in large language models (Shuster et al., 2021). This paper focuses exclusively on three types of query reformulation methods: query expansion, query rewriting, and query processing guidance. Query expansion involves incorporating additional elements into the original query, such as keywords, paragraphs, or even possible answers related to the question. Recent work shows that augmenting LMs with nonparametric memories (i.e., retrieved text chunks) enables much smaller models to match the performance of larger models. Query rewriting means breaking down complex questions, rephrasing ambiguous and vague questions, and so on. When the abstract parts of a query are reduced, or the query expression is transformed into a form that the model is familiar with and can understand, the hallucination problem can be significantly alleviated. Properly guiding the model in processing the query refers to prompt engineering techniques used to unleash the model's potential. These methods typically do not alter the original query but instead guide the model to correctly understand and analyze the problem through in-context learning.

As shown in Figure 1, query expansion in particular can improve the accuracy of large language models' responses. However, the efficiency of these improvements is quite low, indicating that query expansion is unnecessary for most queries. So understanding when and how to reformulate the query is crucial for balancing accuracy improvement and resource consumption. In particular, we aim to address the following research questions:*Do we need to reformulate every query to reduce hallucinations? If not, when should we reformulate a query?If query reformulation is needed, which methods should we use to balance reducing hallucinations and conserving resources?*

This paper investigates these three questions and makes the following main contributions:

1. We propose a query reformulation framework for efficiently reducing hallucinations for large language models. It adaptively selects the most suitable reformulation method for the input query to balance latency and accuracy.
2. We introduce new metrics for evaluating adaptive query reformulation methods.
3. We construct a query reformulation dataset, serving as a new benchmark for adaptive query reformulation methods.

2 Related work

Query expansion Query expansion methods can be categorized into two types based on the scale of expansion: keyword-enhanced query expansion and context-enriched query expansion. Keyword-enhanced query expansion primarily involves adding keywords related to the original query, focusing on enhancing the immediate relevance of the query. Yang and Lin (2019) used semantic matching techniques to search for a series of weighted expansion keywords to address the vocabulary mismatch problem in axiomatic information retrieval. Jaleel et al. (2004) selected the top 200 most probable words from relevance models as expansion keywords, effectively improving recall in the second retrieval process. Jagerman et al. (2023) proposed using large language models to generate new keywords, thereby improving retrieval recall and Mean Reciprocal Rank (MRR). Roy et al. (2016) proposed an automatic query expansion method using word embeddings and KNN.

Context-enriched query expansion provides broader contextual information to the original query, including related background stories, definitions, examples, and even possible answers, thereby offering a more comprehensive query context. Wang et al. (2023) proposed the Query2doc method, which uses large language models to generate pseudo-documents for query expansion. Gao et al. (2022) proposed the HyDE method, which generates hypothetical documents and uses an unsupervised contrastive encoder for zero-shot dense retrieval. Mao et al. (2020) used large language models to generate three types of context (titles, sentences, and answers) to enhance queries, significantly improving the accuracy of downstream models in answering open-domain questions.

Query rewriting Query rewriting is a method of transforming the original query into a form that is easier to retrieve or understand, involving paradigms such as problem decomposition, redundancy removal, and disambiguation. Ma et al. (2023) developed a novel Rewrite-Retrieve-Read framework that enhances the performance of retrieval-augmented LLMs. Mao et al. (2023) developed the LLM4CS framework to leverage large language models for interpreting search intents in conversational queries. Peng et al. (2023) designed the BEQUE framework to address the semantic gap in long-tail query rewriting for E-commerce search.

Query processing guidance Query processing guidance aims to establish a framework for how large language models (LLMs) should think and handle queries. Wei et al. (2022) explore how chain-of-thought prompting enhances the performance of LLMs in managing complex query processing tasks. Dhuliawala et al. (2023) propose the Chain-of-Verification (CoVe) method to reduce hallucinations in LLMs. Zheng et al. (2023) introduce STEP-BACK PROMPTING, a technique that improves LLMs' reasoning capabilities by guiding them to abstract high-level concepts and principles from specific details, thereby solving complex tasks more effectively.

3 Methodology

In this section, we detail the proposed adaptive query reformulation method for large language models. Please refer to Figure 2 for the overall framework flowchart.

3.1 Preliminaries

We first introduce the five query reformulation methods that are part of the stage preceding the use of large language models to answer questions.

3.1.1 Direct answer

Given a large language model and a query q, we do not perform any preprocessing on q. Instead, we use a tokenizer to convert q into a sequence of tokens $[t_1, \ldots, t_n]$, which are then processed by the *LLM* to generate an answer $[a_1, \ldots, a_n]$ = *LLM*(q). Directly inputting the unprocessed query into the *LLM* serves as a simple and fast baseline method in our framework. However, it only leverages the model's parametric knowledge and is highly influenced by the query's format as understood by the model. This strategy is typically best for straightforward and general questions.

3.1.2 Add CoT before QA

Compared to having the language model answer directly, prompting the language model to output a lengthy but well-structured response often results in higher accuracy. The chain-of-thought (CoT) prompting method involves adding prompts to the original query to generate a chain of thought. This new query allows the instruction-following language model to fully utilize its parametric knowledge and enhance its reasoning abilities. It can be considered a form of query processing guidance. It does not modify the query itself but guides the language model to produce an answer step-by-step.

3.1.3 Query Rewrite before QA

Queries input by ordinary users often contain ambiguities, typos, grammatical errors, or are overly complex. The presentation of the query—such as its structure, phrasing, and word choice—can significantly affect the accuracy of the language model's response, even without changing the problem's difficulty. We select an instruction-following language model or fine-tune a smaller language model to act as a query rewriter, transforming the original query q into a more understandable form q' for the subsequent base model. This rewriting process aims to improve the accuracy of the query responses, represented as $q' = \mathit{Rewriter}(q)$. The language model then processes the rewritten query to generate an answer, $[a_1, \ldots, a_n] = \mathit{LLM}(q')$.

3.1.4 Query Expansion before QA

Many questions cannot be answered solely by relying on the model's internal knowledge. In such cases, it is necessary to actively retrieve external knowledge to supplement the query. For example, a language model trained on data before 2023 cannot answer questions about events occurring in 2024. In this scenario, we initialize a retriever and use the query q as the input to the retriever. The retriever returns a set of relevant documents $d = \mathit{retriever}(q)$ from an external knowledge base (e.g., Wikipedia) based on the query. These relevant documents are then combined with the original query to form an expanded query $q' = q \oplus d$, which is input into the language model. The language model then processes the expanded query to generate an answer, $[a_1, \ldots, a_n] = \mathit{LLM}(q')$. This process is a classic Retrieval-Augmented Generation (RAG) method.

3.1.5 Self-Knowledge Guided QA

Due to the limitations of the maximum reasoning capabilities of QA LLMs and the currently available query reformulation methods, we acknowledge that some questions cannot be answered by the downstream language model, regardless of the query reformulation method used. For such cases, we propose a special reformulation method that incorporates a specific "don't know" prompt to guide the QA language model's self-knowledge[1]. This approach enhances the QA language model's awareness of its own limitations and reduces the hallucination of fabricated facts.

Formally, given a query q that the language model is unable to answer, we reformulate it by adding a "don't know" prompt, resulting in $q' = q +$ " don't know". The language model then processes the reformulated query to generate an answer, $[a_1, \ldots, a_n] = \mathit{LLM}(q')$, where the presence of the "don't know" prompt guides the model to appropriately acknowledge its limitations.

This method aims to reduce the instances of the model producing hallucinated or fabricated answers by making it explicitly aware of when it lacks

[1]Self-knowledge refers to the model's awareness of its own limitations and understanding of when it can't provide a correct answer.

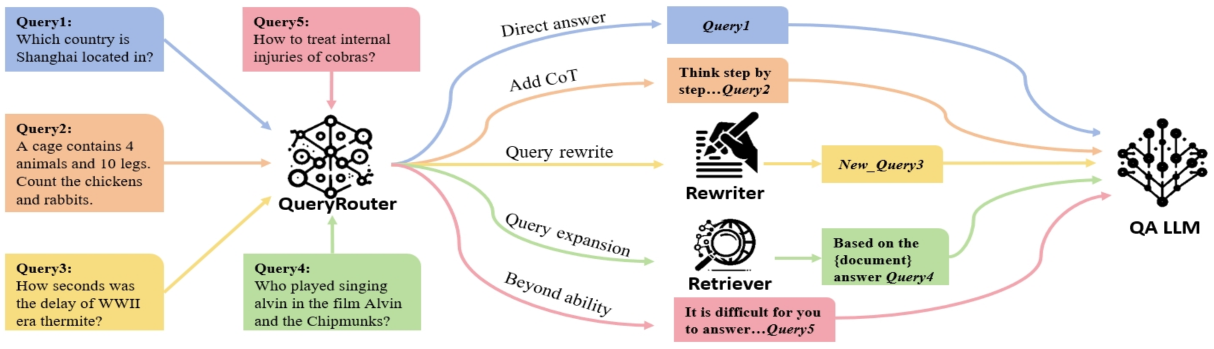
Query1:
Which country is Shanghai located in?
Query5:
How to treat internal injuries of cobras?
Query2:
A cage contains 4 animals and 10 legs. Count the chickens and rabbits.
Query3:
How seconds was the delay of WWII era thermite?
Query4:
Who played singing alvin in the film Alvin and the Chipmunks?
QueryRouter
Direct answer
Add CoT
Query rewrite
Query expansion
Beyond ability
Query1
Think step by step...Query2
Rewriter
New_Query3
Retriever
Based on the {document} answer Query4
It is difficult for you to answer...Query5
QA LLM

Figure 2: Flowchart of our adaptive query reformulation method.

sufficient information to provide a correct response.

3.2 Adaptive Query Reformulation Framework

In this section, we formally introduce the adaptive query reformulation framework. Our intuition is that (1) different queries require distinct optimal reformulation strategies, and (2) selecting the most appropriate method in advance can significantly enhance the accuracy of subsequent model responses while reducing latency. The overall framework process is as follows: input the query, select the best reformulation method, reformulate the query, and then the large language model provides the answer.

3.2.1 Adaptive Query Routing

The core of this framework is the adaptive selection of the most suitable reformulation method for the input query. We achieve this by pre-training a lightweight language model specifically for classifying raw queries, allowing us to route different queries to their most appropriate reformulation paths. Specifically, given a query q, we determine the suitable reformulation method m using the *QueryRouter* as follows: $m = \mathit{QueryRouter}(q)$. The construction of this router involves two stages: pre-training and fine-tuning.

During the pre-training stage, like most traditional language models, the lightweight language model undergoes self-supervised training on a large amount of unsupervised data to acquire the basic ability to understand normal query sentences (Radford and Narasimhan, 2018). Given an unsupervised corpus of tokens $\mathcal{U} = \{u_1, \ldots, u_n\}$, we use a standard language modeling objective to maximize the following likelihood:

$$\mathcal{L}_1(\mathcal{U}) = \sum_i \log P(u_i \mid u_{i-k}, \ldots, u_{i-1}; \Theta) \quad (1)$$

where k is the size of the context window, and the conditional probability P is modeled using a transformer variant with parameters Θ.

During the fine-tuning stage, we first construct the query routing dataset through the following steps: (1) Each question in the training set is processed through all available routing paths to generate corresponding responses. (2) For each question, the path that produces the correct answer with the shortest response time[2] is designated as the ground-truth label. If a question remains unanswered correctly after applying the first four reformulation methods from the preliminary phase, it is automatically assigned label 0, corresponding to the fifth reformulation method. (3) To enhance the dataset, each question is paraphrased into a semantically similar variant while preserving the original label. (4) Low-quality samples are systematically filtered out using rule-based scripts and manual verification to ensure dataset quality and consistency. After constructing the dataset, we fine-tune the model as a query router using the following loss function:

$$\mathcal{L}_2 = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \quad (2)$$

where C represents the total number of classes (in this case, 5, corresponding to labels 0 through 4), y_i denotes the one-hot encoded ground-truth label, and $\hat{y}_i$ corresponds to the predicted probability for each class.

3.2.2 Query Reformulation

After routing the input query to the appropriate reformulation method, the query enters the reformulation stage.

[2]This measurement begins when a question is input into the system and ends when the last token of the answer is generated by the large language model.

As shown in Figure 2, if the question is straightforward, such as a common knowledge question like *query1*, it is directly fed into the QA LLM for an answer. However, if the query involves mathematical or logical reasoning, such as *query2*, the reformulation stage adds the classic prompt "Think step by step..." before the query to guide the downstream QA LLM to generate a chain of thought. If the query is deemed ambiguous, syntactically ill-formed or too complex such as *query3*, a query rewriter is invoked during the reformulation stage. This rewriter refines vague or underspecified queries to enhance clarity, restructures grammatically incorrect inputs for better readability, or decomposes inherently complex queries into simpler sub-queries *New_Query3*.

If the question cannot be answered using the QA LLM's parametric knowledge, such as *query4*, it will be routed to the Query Expansion path.

If the query router determines that the above methods cannot enable the QA LLM to answer correctly, the query is routed to the self-knowledge guidance path. In this path, the query is augmented with prompts such as "don't know" to suppress the hallucination of the downstream large language model.

4 Experimental Setups

In this section, we provide a detailed introduction to the benchmarks, metrics, baselines, framework components, and implementation.

4.1 Benchmarks

AmbigNQ focuses on the inherent ambiguity present in open-domain question answering. This benchmark involves an open-domain question answering task that requires the prediction of a set of question-answer pairs (Min et al., 2020). Each pair features a plausible answer coupled with a disambiguated rewrite of the original question, emphasizing the clarification of ambiguous inquiries.
HotpotQA is a question answering benchmark that requires multi-hop reasoning across multiple documents to answer questions (Yang et al., 2018). It provides a diverse set of questions and sentence-level supporting facts to guide reasoning, and includes novel comparison questions that challenge systems to analyze and compare detailed facts.
MMLU-STEM is a subset of the MMLU benchmark (Hendrycks et al., 2020), focusing on STEM subjects ranging from basic topics like high school biology to advanced areas like machine learning. It evaluates models in zero-shot and few-shot settings, testing their knowledge and problem-solving skills across various educational levels.
PopQA is an open-domain QA dataset with entity-centric QA pairs sourced from Wikidata (Mallen et al., 2022). Questions are generated using templates and cover a variety of entities, highlighting their popularity. The dataset aims to assess language models' ability to handle entities of varying popularity.

4.2 Metrics

For the benchmarks AmbigNQ, HotpotQA, and PopQA, we use the accuracy metric following the standard evaluation protocol in similar works. Here, accuracy is defined as the proportion of outputs that contain the correct answer (ground truth answer). For the MMLU-STEM benchmark, since it consists of multiple-choice questions, we use the Exact Match (EM) metric, where EM is the proportion of outputs that exactly match the answer.

Furthermore, for the adaptive query reformulation framework, task performance and resource consumption must be balanced. However, the academic community currently lacks corresponding efficiency evaluation metrics. Based on this, we propose two metrics for evaluating adaptive query reformulation: the time consumed per correct answer and the time consumed per accuracy improvement.

The first metric, *Time per Correct Answer (TCA)*, is defined as the total time consumed in answering all questions in the dataset divided by the number of correct answers:

$$\text{TCA} = \frac{T_{\text{total}}}{N_{\text{correct}}} \tag{3}$$

where T_{total} is the total time consumed and N_{correct} is the number of correct answers. The second metric, *Time per Accuracy Improvement (TAI)*, is defined as the difference in time between using the query reformulation method and not using it, divided by the difference in accuracy achieved with and without the query reformulation method:

$$\text{TAI} = \frac{T_{\text{method}} - T_{\text{baseline}}}{A_{\text{method}} - A_{\text{baseline}}} \tag{4}$$

where T_{method} and T_{baseline} are the total times consumed with and without the query reformulation method, respectively, and A_{method} and A_{baseline} are the accuracies achieved with and without the query reformulation method, respectively.

Samples	Content
1	{"question": "Who plays the doctor in Dexter season 1?", "answer": ["Tony Goldwyn", "Goldwyn"], "method_choice": 2}
...	...
16000	{"question": "The Visitors falls under which genre?", "answer": ["Pop music"], "method_choice": 3}

Table 1: The format of the query routing dataset. Method choice: 0 represents Beyond ability, 1 represents Direct answer, 2 represents Add CoT, 3 represents Query rewrite, and 4 represents Query expansion.

4.3 Framework Component

QueryRouter We use BERT-base (110M) and BERT-large (335M) pretrained on the SQuAD and GLUE datasets as the base models for QueryRouter.

Rewriter We follow the implementation from (Ma et al., 2023), using T5-large (738M) as the query rewriter, responsible for decomposing complex questions and reformulating ambiguous ones.

Retriever We use Bing v7[3] as the retriever in our framework.It retrieves up to 5 relevant documents for each query, with a maximum of 800 tokens per document.

QA LLM We choose GPT-3.5-turbo-16k[4] as the QA LLM for the final stage of the framework. It performs reading comprehension and prediction with few-shot or zero-shot in-context learning.

4.4 Implementation Details

Here we detail the implementation specifics of our experiments.

Fine-tuning Following the procedure outlined in Section 3.2.1, we constructed a fine-tuning dataset and augmented it using GPT-3.5, combined with manual filtering, resulting in a dataset containing 16,000 query routing samples. The specific data format is shown in Table 1. We performed supervised fine-tuning (SFT) on BERT-base and BERT-large models using this new dataset, with hyperparameters detailed in Table 2.

[3]https://api.bing.microsoft.com/v7.0/search

[4]https://api.openai.com/v1/chat/completions

Prompts We employed zero-shot or few-shot in-context learning for tasks related to data augmentation, query reformulation, and QA stages. Detailed prompts are provided in A.

Hardware Setup All experiments were conducted on a single A100 GPU and two 4090 GPUs.

Experiments We conduct both baseline and comparative experiments to evaluate the effectiveness of our framework. The experiments include four parts: a comparison with individual query reformulation methods, a comparison with other adaptive query reformulation approaches, an analysis of the impact of replacing the query router, and an evaluation of the effect of replacing the QA system. Detailed results are presented in Section 5.

Hyperparameter	Value
Number of Epochs	5
Train Batch Size per Device	8
Warmup Steps	600
Weight Decay	0.005
Max Length	512
Learning Rate	5e-5
Optimizer	AdamW

Table 2: Key hyperparameters used in fine-tuning BERT.

5 Results and Analyses

5.1 Comparison with Baselines

Table 3 presents the main results of our method compared to the baselines. As shown in the table, the CoT, QE, and QR query reformulation methods all improve Acc and EM compared to direct answering. In terms of improving Acc and EM efficiency, the CoT method outperforms the QR method, and the QR method outperforms the QE method. The primary reason is that invoking a retriever is more time-consuming than language model inference. Our method surpasses the aforementioned three methods in both accuracy (or EM) and efficiency because it integrates the advantages of these methods and adaptively selects the least time-consuming query reformulation method that can improve Acc or EM. This result demonstrates that our approach is a more fine-grained query reformulation method rather than applying a uniform approach for all queries.

Dataset	Metrics	Direct answer	Add CoT	Query expansion	Query rewrite	Query routing (Ours)
AmbigNQ	Acc(%)	42.65	44.60	49.20	44.80	**56.85**
	TCA(s)	**1.98**	5.26	15.08	10.39	3.76
	TAI(s/%)	NaN	1538.50	2007.40	3546.30	**182.50**
Hotpot	Acc(%)	31.75	38.00	42.45	44.60	**50.90**
	TCA(s)	**2.74**	5.72	19.29	9.92	4.60
	TAI(s/%)	NaN	416.14	1367.35	552.93	**153.53**
MMLU-STEM	EM	44.20	65.34	63.26	62.23	**78.63**
	TCA(s)	**2.13**	3.90	11.26	8.59	3.23
	TAI(s/%)	NaN	273.69	1169.67	880.66	**167.81**
PopQA	Acc(%)	32.70	32.80	50.25	35.45	**52.25**
	TCA(s)	**1.39**	6.78	14.61	10.90	4.35
	TAI(s/%)	NaN	3534.20	784.53	2477.71	**185.74**

Table 3: Performance comparison of Accuracy and Efficiency across Baseline Query Reformulation Methods

5.2 Comparison with Other Adaptive Methods

Table 4 presents the comparison results with other adaptive query reformulation methods. It is noteworthy that the query reformulation in these four methods primarily refers to query expansion. The FLARE method emphasizes adaptively triggering retrieval during the generation process (Jiang et al., 2023). The SELF-RAG method incorporates retrieval and self-reflection during generation (Asai et al., 2023). The ANTLM method determines the necessity of retrieval based on the popularity of the entity and the relationship type (Mallen et al., 2022). Adaptive-RAG trains a classifier to assess the difficulty of the query and decide the number of retrievals (Jeong et al., 2024).

Main results Our method outperforms the other four adaptive query reformulation methods in terms of improving the efficiency of question answering accuracy (Acc) or exact match (EM). Notably, it achieves significant improvements in answering mathematical and logical questions from the MMLU-STEM dataset. However, in terms of Acc, our method is optimal only on the AmbigNQ dataset, whereas FLARE and SELF-RAG perform better on the Hotpot and PopQA datasets.

Analyses on Outperformance The primary advantage of our method over other approaches lies in the efficiency of improving answer accuracy. This is evidenced by our TCA and TAI metrics, which are significantly lower than those of other methods, indicating that our method answers more questions correctly in less time. The reason for this efficiency is that retrieval is time-consuming, and our query router helps avoid external retrieval when the model's internal knowledge can answer the question. In contrast, other methods such as FLARE, SELF-RAG, and Adaptive-RAG have low retrieval thresholds and allow multiple retrievals for the same query, consuming unnecessary time.

Our method achieves substantial improvement in the EM metric on the MMLU-STEM dataset because it includes the option of adding Chain-of-Thought (CoT). For mathematical and logical questions, the logical reasoning provided by CoT is more beneficial than additional supplementary information.

Analyses on Underperformance Our method underperforms compared to FLARE and SELF-RAG in terms of accuracy on fact-based datasets such as Hotpot and PopQA, and only narrowly surpasses SELF-RAG on the AmbigNQ dataset. The main reason for this is that the bottleneck limiting accuracy on entity-focused datasets is the relevance of the external knowledge base rather than the variation in query formulation. SELF-RAG and FLARE's multiple retrievals provide much more relevant knowledge than our method, which limits the number of retrievals to at most one. Consequently, our method does not achieve the same level of absolute accuracy as these two methods.

5.3 Impact of Replacing the Query Router

To test the sensitivity of the entire framework to the query router, we conducted experiments replacing the query router. We mainly adjusted the size and the number of classifications of the query router, conducting experiments on the AmbigNQ and PopQA datasets. The experimental results are shown in Table 5 and Table 6. The 4-class classifier combines the original QR path and CoT path,

Dataset	Metrics	ANTLM	FLARE	Self-RAG	Adaptive-RAG	Query routing (Ours)
AmbigNQ	Acc(%)	55.40	54.20	56.70	54.40	**56.85**
	TCA(s)	10.88	8.91	14.22	7.41	**3.76**
	TAI(s/%)	813.49	690.42	1027.84	542.47	**182.50**
Hotpot	Acc(%)	49.00	52.67	**53.60**	48.36	50.90
	TCA(s)	13.83	10.36	15.25	9.18	**4.60**
	TAI(s/%)	685.04	438.36	668.87	429.86	**153.53**
MMLU-STEM	EM	64.37	64.26	70.64	64.58	**78.63**
	TCA(s)	10.09	8.04	10.50	7.78	**3.23**
	TAI(s/%)	994.28	760.91	884.16	722.23	**167.81**
PopQA	Acc(%)	51.28	55.34	**55.80**	51.79	52.25
	TCA(s)	12.67	8.53	13.35	11.20	**4.35**
	TAI(s/%)	650.61	377.05	605.51	559.93	**185.74**

Table 4: Performance comparison of Different Adaptive Query Reformulation Methods.

considering that problem decomposition can sometimes be replaced by CoT. The experiments show that a larger query router does not bring performance improvements. We believe this is due to the current fine-tuning dataset being insufficient to support the convergence of the larger BERT-large model. Additionally, the 4-class router performs better than the 5-class router on the PopQA dataset but worse on the AmbigNQ dataset. This is because fewer classification categories lead to higher routing accuracy, which improves Acc_QA performance on the PopQA dataset where there is no significant difference between the performance of the QR and CoT methods. In the AmbigNQ dataset, many questions are ambiguous, and in these cases, the CoT method cannot replace the QR method (decomposing the question to make it clear and simple).

	Metrics	5-class	4-class
BERT-base	Acc_{router} (%)	74.03	78.68
	Acc_{QA} (%)	56.85	55.73
	TCA (s)	3.76	3.56
	TAI (s/%)	182.50	176.35
BERT-large	Acc_{router} (%)	65.15	67.54
	Acc_{QA} (%)	50.75	53.65
	TCA (s)	4.83	4.28
	TAI (s/%)	397.36	263.64

Table 5: Performance comparison of different query routers on the AmbigNQ dataset.

	Metrics	5-class	4-class
BERT-base	Acc_{router} (%)	74.03	78.68
	Acc_{QA} (%)	52.25	53.35
	TCA (s)	4.35	4.37
	TAI (s/%)	185.74	181.74
BERT-large	Acc_{router} (%)	65.15	67.54
	Acc_{QA} (%)	48.50	49.25
	TCA (s)	4.98	4.72
	TAI (s/%)	248.65	225.88

Table 6: Performance comparison of different query routers on the PopQA dataset.

5.4 Performance with Different QA Systems

To further validate the effectiveness of our query routing method across different QA systems, we conduct experiments on the PopQA dataset using two additional LLMs: GPT-4o and Qwen2-72B.

Table 7 presents the results, showing that models directly responding to queries without query routing or query expansion (QE) exhibit relatively low accuracy due to the lack of additional context, though inference latency remains low. Applying QE improves accuracy by retrieving relevant external chunks, but the gains come at a steep efficiency cost, as reflected in the TCA and TAI metrics.

In contrast, our adaptive query routing method significantly enhances accuracy across all three QA systems while keeping inference latency within an acceptable range. These findings confirm that our approach effectively balances accuracy and efficiency, making it adaptable to diverse QA models.

Model	TCA	TAI	Acc
GPT-3.5	1.39	NaN	32.70
GPT-4o	2.04	NaN	48.50
Qwen2-72B	2.36	NaN	47.30
GPT-3.5 with Router	4.35	185.74	52.25
GPT-4o with Router	5.23	444.13	57.62
Qwen2-72B with Router	5.79	595.48	54.05
GPT-3.5 with QE	14.61	784.53	50.25
GPT-4o with QE	13.69	1968.36	55.17
Qwen2-72B with QE	15.04	2793.97	52.11

Table 7: Performance comparison of different QA systems with and without query routing and QE on the PopQA dataset.

6 Conclusion

In this work, we propose an adaptive query reformulation framework to efficiently reduce hallucinations in large language models. This framework adaptively selects the most suitable query reformulation method for the input queries of QA LLMs, thereby improving the accuracy of question answering while minimizing unnecessary time consumption. Comparative experiments show that our framework outperforms baselines in terms of both accuracy and efficiency, and it surpasses four mainstream adaptive query expansion methods in efficiency. Additionally, we introduce two new metrics, TCA and TAI, which effectively evaluate the performance of adaptive query reformulation methods. Finally, we present a benchmark for query adaptive reformulation to further advance research in this area.

Limitations

Our framework has several limitations. Firstly, the query expansion path in our framework is designed to perform a single retrieval, which is less effective for handling long-tail queries compared to methods like FLARE and Self-RAG that involve multiple retrievals. Secondly, our framework currently incorporates only four major query reformulation methods, which may not cover the optimal reformulation method for every input query. Expanding the range of reformulation methods within the framework is a crucial direction for improvement. However, it is important to note that increasing the number of reformulation paths will also increase the number of classification labels for the classifier, necessitating more training data and potentially leading to convergence difficulties. Finally, our method involves calling APIs, and network issues can impact the experimental results, posing challenges for the reproducibility of the research. Nonetheless, we are confident that the qualitative conclusions of this paper remain unaffected by these factors.

Ethics Statements

Our experiments rely exclusively on publicly available datasets, which are widely used in the research community and comply with their respective licenses and terms of use. No proprietary or private data was used in our research.

However, as these datasets are predominantly in English, our method may exhibit biases when applied to multilingual contexts. Additionally, we acknowledge that training the query router may introduce biases—for instance, if someone were to train a router that disproportionately favors a specific company's retrieval system.

To promote transparency and facilitate further research, we provide detailed descriptions of our experimental setups, benchmarks, evaluation metrics, and implementation details.

Finally, this research is intended to enhance user experience with large language models and assist enterprises in improving their question-answering systems. We explicitly oppose any use of this research for unethical purposes, including but not limited to military applications.

References

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. 2023. Chain-of-verification reduces hallucination in large language models. *ArXiv*, abs/2309.11495.

Luyu Gao, Xueguang Ma, Jimmy J. Lin, and Jamie Callan. 2022. Precise zero-shot dense retrieval without relevance labels. *ArXiv*, abs/2212.10496.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300.

Rolf Jagerman, Honglei Zhuang, Zhen Qin, Xuanhui Wang, and Michael Bendersky. 2023. Query expansion by prompting large language models. *arXiv preprint arXiv:2305.03653*.

Nasreen Jaleel, James Allan, W. Croft, Fernando Diaz, Leah Larkey, Xiaoyan Li, Mark Smucker, and Courtney Wade. 2004. Umass at trec 2004: Novelty and hard.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C. Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. *ArXiv*, abs/2403.14403.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Active retrieval augmented generation. *ArXiv*, abs/2305.06983.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models. *ArXiv*, abs/2305.14283.

Alex Troy Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Hannaneh Hajishirzi, and Daniel Khashabi. 2022. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Annual Meeting of the Association for Computational Linguistics*.

Kelong Mao, Zhicheng Dou, Haonan Chen, Fengran Mo, and Hongjin Qian. 2023. Large language models know your contextual search intent: A prompting framework for conversational search. In *Conference on Empirical Methods in Natural Language Processing*.

Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. 2020. Generation-augmented retrieval for open-domain question answering. In *Annual Meeting of the Association for Computational Linguistics*.

Sewon Min, Julian Michael, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2020. Ambigqa: Answering ambiguous open-domain questions. In *Conference on Empirical Methods in Natural Language Processing*.

OpenAI. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Wenjun Peng, Guiyang Li, Yue Jiang, Zilong Wang, Dan Ou, Xiaoyi Zeng, Tongxu, and Enhong Chen. 2023. Large language model based long-tail query rewriting in taobao search. *Companion Proceedings of the ACM on Web Conference 2024*.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pretraining.

Dwaipayan Roy, Debjyoti Paul, Mandar Mitra, and Utpal Garain. 2016. Using word embeddings for automatic query expansion. *ArXiv*, abs/1606.07608.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Liang Wang, Nan Yang, and Furu Wei. 2023. Query2doc: Query expansion with large language models. In *Conference on Empirical Methods in Natural Language Processing*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Peilin Yang and Jimmy Lin. 2019. Reproducing and generalizing semantic term matching in axiomatic information retrieval. In *Advances in Information Retrieval*, pages 369–381, Cham. Springer International Publishing.

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. In *Conference on Empirical Methods in Natural Language Processing*.

Huaixiu Steven Zheng, Swaroop Mishra, Xinyun Chen, Heng-Tze Cheng, Ed Huai hsin Chi, Quoc V. Le, and Denny Zhou. 2023. Take a step back: Evoking reasoning via abstraction in large language models. *ArXiv*, abs/2310.06117.

Chunting Zhou, Graham Neubig, Jiatao Gu, Mona Diab, Francisco Guzmán, Luke Zettlemoyer, and Marjan Ghazvininejad. 2021. Detecting hallucinated content in conditional neural sequence generation. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1393–1404, Online. Association for Computational Linguistics.

A Prompts and Exemplars

Here, we present the prompts used in the query reformulation process and the corresponding responses from GPT-3.5, which are shown in Tables 8, 9, 10, 11, and 12.

Task	**Prompt**	**Exemplar**
Direct answer (for Hotpot, AmbigNQ, and PopQA)	Please answer the following questions directly. Your answers should not exceed ten words. **Question:** {query}	**Query:** Who plays the doctor in Dexter season 1? **Output:** Julie Benz plays the doctor in Dexter season 1.
Direct answer (for MMLU-STEM)	Please answer the following questions directly with a letter in 'A', 'B', 'C', or 'D'. **Question:** {query}	**Query:** Find the degree for the given field extension over $\mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{18})$. **Options:** A. 0 B. 4 C. 2 D. 6 **Output:** The answer is D

Table 8: Prompts used in the Direct Answer Path and Corresponding Exemplars

Task	**Prompt**	**Exemplar**
Guide self-knowledge	The following questions are likely to exceed your knowledge boundaries. Please double check if you can answer them, and if not, report truthfully. **Question:** {query}	**Query:** How to treat internal injuries of cobras? **Output:** Sorry, I am not familiar with the knowledge of cobra treatment. Please replace the QA LLM or retriever.

Table 9: Prompts Used in Guiding Self-Knowledge and Corresponding Exemplars

Task	**Prompt**	**Exemplar**
Add CoT (for Hotpot, AmbigNQ, and PopQA)	Please think step by step and answer the following question. Your answer must reflect the thought process and conclude with "so the answer is...". **Question:** {query}	**Query:** Who plays the doctor in Dexter season 1? **Output:** First, I need to recall the cast of Dexter Season 1. The main character, Dexter Morgan, is played by Michael C. Hall. To identify who plays the doctor in Dexter Season 1, I need to think about the recurring characters in the show. Dr. James Doakes is a significant character in the first season, but he is not a doctor. Dr. Evelyn Vogel is introduced in later seasons, so she is not the doctor in Season 1. The doctor in Dexter Season 1 is Dr. Emmett Meridian, who is portrayed by actor Tony Goldwyn. So the answer is Tony Goldwyn plays the doctor in Dexter Season 1.
Add CoT (for MMLU-STEM)	Please think step by step and answer the following question with a letter in 'A', 'B', 'C', or 'D'. Your answer must reflect the thought process and conclude with "so the answer is A/B/C/D". **Question:** {query}	**Query:** Find the degree for the given field extension over $\mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{18})$. **Options:** A. 0 B. 4 C. 2 D. 6 **Output:** To find the degree of the field extension over $\mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{18})$, we consider each extension individually. The degree of $\mathbb{Q}(\sqrt{2})$ over $\mathbb{Q}$ is 2 because the minimal polynomial of $\sqrt{2}$ over $\mathbb{Q}$ is $x^2 - 2$. Similarly, the degree of $\mathbb{Q}(\sqrt{3})$ over $\mathbb{Q}$ is 2 because the minimal polynomial of $\sqrt{3}$ over $\mathbb{Q}$ is $x^2 - 3$. However, $\sqrt{18}$ can be expressed as $3\sqrt{2}$, which does not introduce a new degree since it's dependent on $\sqrt{2}$. Therefore, the combined field extension $\mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{18})$ is essentially $\mathbb{Q}(\sqrt{2}, \sqrt{3})$ because $\sqrt{18}$ is not an independent element. The total degree of the field extension, calculated as the product of independent extensions, is $2 \times 2 = 4$ because the extensions for $\sqrt{2}$ and $\sqrt{3}$ are independent. Therefore, the correct answer is B.

Table 10: Prompts Used in the Add CoT Path and Corresponding Exemplars

Task	Prompt	Exemplar
Query rewrite (for Hotpot, AmbigNQ, and PopQA)	Please break down the following questions into a series of simpler sub questions (no more 5) and ensure that the sub questions are helpful in answering the original question. **Question:** {query}	**Query:** Who plays the doctor in Dexter season 1? **Output:** 1. Who are the main characters in Dexter season 1? 2. What is the role of the doctor in Dexter season 1? 3. What is the actor's name who plays the doctor in Dexter season 1? 4. Are there any notable scenes involving the doctor in Dexter season 1?
Query rewrite (for MMLU-STEM)	Please think step by step and provide the necessary background to address the following question. **Question:** {query}	**Query:** Find the degree for the given field extension $\mathbb{Q}(\sqrt{2}, \sqrt{3}, \sqrt{18})$ over $\mathbb{Q}$. **Options:** A. 0 B. 4 C. 2 D. 6 **Output:** 1. What is a field extension in mathematics? 2. How do you calculate the degree of a field extension? 3. What are the properties of square roots in field extensions? 4. How do you determine the degree of a composite field extension?

Table 11: Prompts Used in the Query Rewrite Path and Corresponding Exemplars

Task	Prompt	Exemplar
Get key words for retriever	Identify and list the key search terms related to answering the following question, separated by ';'. **Question:** {query}	**Query:** Who plays the doctor in Dexter season 1? **Output:** doctor; Dexter; season 1.
Answer with related document	If the following passage helps with the question, please refer to it; if it does not help, please answer on your own. Keep your answer within 30 words. **Passage :** {background} **Question:** {query}	**Background:** The first season of Dexter is an adaptation of Jeff Lindsay's first novel in a series of the same name, Darkly Dreaming Dexter. Subsequent seasons have featured original storylines–Dexter finds the victim mutilated but alive and strapped to a table for Dexter to kill. Instead, Dexter reports the crime scene anonymously to his sister. The pressure on Sergeant Doakes escalates when Guerrero's associates begin following him. **Query:** Who plays the doctor in Dexter season 1? **Output:** The doctor in Dexter season 1 is portrayed by actor Christian Camargo.

Table 12: Prompts Used in the Query Expansion Path and Corresponding Exemplars

B Time Consumption Analysis

This section analyzes the time consumption of our experiments.

For fine-tuning, we follow the hyperparameter settings in Table 2. The model is trained on a single NVIDIA RTX 4090 GPU with 12,800 training and 3,200 validation samples. Fine-tuning BERT-base takes 260 seconds per epoch, while BERT-large takes 320 seconds. During validation and inference, the model processes 263 samples per second.

For comparative experiments, we evaluate our method against baselines and other adaptive query reformulation strategies on four datasets. MMLU-STEM contains 3,609 samples, while the other datasets have 2,000 samples each. Total time costs are summarized in Tables 13 and 14. Network fluctuations introduce variability in time measurements, particularly when calling large language model APIs or retrieval APIs, causing occasional inconsistencies in processing times for the same input. However, these fluctuations are infrequent and affect only a limited number of queries.While network latency impacts absolute efficiency, it does not change the study's qualitative conclusions. Methods with query routing consistently achieve lower TCA and TAI values than those without, confirming the efficiency advantages of our approach.

Dataset	Path	Time (s)
AmbigNQ	Direct Answer	1688
	Add CoT	4688.07
	Query Expansion	14836.32
	Query Rewrite	9312.52
	Ours	**4279.54**
Hotpot	Direct Answer	1743
	Add CoT	4343.88
	Query Expansion	16373.64
	Query Rewrite	8848.21
	Ours	**4683.08**
MMLU-STEM	Direct Answer	3403.69
	Add CoT	9189.57
	Query Expansion	25697.66
	Query Rewrite	19282.03
	Ours	**9182.81**
PopQA	Direct Answer	911.5
	Add CoT	4446.12
	Query Expansion	14680
	Query Rewrite	7725.21
	Ours	**4542.66**

Table 13: Total time consumption in comparative experiment 5.1.

Dataset	Method	Time (s)
AmbigNQ	ANTLM	12060
	FLARE	9662.4
	Self-RAG	16129.2
	Adaptive-RAG	8062
	Ours	**4279.54**
Hotpot	ANTLM	13560
	FLARE	10913.64
	Self-RAG	16358
	Adaptive-RAG	8883
	Ours	**4683.08**
MMLU-STEM	ANTLM	23458.5
	FLARE	18667.66
	Self-RAG	26781
	Adaptive-RAG	18122.66
	Ours	**9182.81**
PopQA	ANTLM	13000
	FLARE	9448
	Self-RAG	14899
	Adaptive-RAG	11600
	Ours	**4542.66**

Table 14: Total time consumption in comparative experiment 5.2.