Small Empowering Large: Leverage Performance and Efficiency by Exploring the Cooperation of LLMs

Anonymous ACL submission

Abstract

The combined use of Large Language Models (LLMs) and Information Retrieval (IR) has made significant progress in solving the multi-hop QA problem. However, achieving high performance requires increasingly complex and interactive integration of IR and "large" LLMs, which poses challenges to efficiency and domain specialization capabilities. A specifically fine-tuned "small" LLM, such as LlaMa-7B, presents a viable solution to this challenge. Nevertheless, addressing the challenges entails considering two aspects: 1) **Where** Problem: identifying the phases in which employing a "small" LLMs is most beneficial is essential. 2) **How** Problem: devising effective strategies for combining "small" and "large" LLMs is necessary. A lightweight approach is proposed where the "large" LLMs service and a specifically fine-tuned "small" LLMs cooperate to answer the multi-hop questions. Our research reveals that the "large" LLMs service primarily handles top-level planning, while the fine-tuned "small" LLMs is tasked with generating answers and rectifying any inconsistencies with the retrieved information. Experimental results on the HotPotQA dataset demonstrate that our proposed method achieves comparable accuracies with significantly reduced costs.

1 Introduction

Currently, Large Language Models (LLMs) matching Information Retrieval (IR) engines have demonstrated impressive performance on knowledge-intensive tasks such as complex multi-hop problems (Menick et al., 2022; Liu et al., 2023c,a). Considering the advantages of the IR engines for real-time and long-tail content, together with the decomposition and contemplation ability of LLMs for complex problems, it improves the accuracy and interpretability in multi-hop scenarios. Despite the effectiveness of this combination, there are still many problems, such as high interaction costs and difficulty in domain specialization. We first categorized these methods and then analyzed the problems with each category.

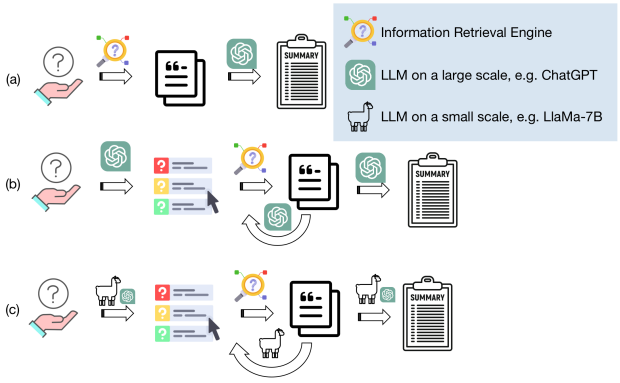


Figure 1: Three categories for combining LLMs and IR.

According to the role played by the large language model, ongoing efforts to integrate search engines and large language models can be divided into two categories. The first framework is the LLM acts predominantly as a summarizer for the outputs generated by IR systems (Feng et al., 2023; Qin et al., 2023), leverages the LLM's capacity to synthesize and conclude information from retrieved documents, as illustrated in Figure 1(a). Since the LLM's role is confined to processing the final stage of summarizing information without engaging in the initial query understanding or information filtering processes (Liu et al., 2023d; Schick et al., 2023), the approach cannot utilize the strong comprehension and inference capabilities of LLM. Conversely, as illustrated in Figure 1(b), the second framework represents a more holistic integration of LLM throughout the entire question resolution process (Ram et al., 2023; Peng et al., 2023; Xu et al., 2023). This includes initial question decomposition, answer recall, and finally summarization. Although this method makes fuller use of the capabilities of the LLMs, it requires fre-

quent interactions between the IR system and the LLMs, resulting in increased computational costs and potentially affecting the smoothness of user interactions (Hu et al., 2023a; Bang, 2023). Another common challenge for both frameworks is the lack of domain-specific adaptations for LLMs (Liu et al., 2023e; Zhang et al., 2023c). Tailoring these models to specific fields or topics could largely enhance their accuracy and reliability. However, the requisite supplementary training/fine-tuning is prohibitive in terms of computational costs and time.

Considering the above issues, we attempt to synergize the "small" and "large" LLMs for a multi-hop reasoning task, thereby mitigating the inherent contradictions between model scalability and the necessity for specialized domain adaptation. In this way, inference cost reduction and domain specialization can be achieved by "small" LLM while being able to maintain the advanced capabilities of the "large" LLM. For ease of notation, we denote "large" LLMs that consist of 100+ billion parameters (e.g., ChatGPT, Gemini), by **LaLM**; and "small" LLMs (e.g., LLaMA-7B, Alpaca-13B) by **SmLM**. To make LaLM and SmLM collaborate for multi-hop reasoning, two key questions need to be addressed:

- **WHERE** to replace the LaLM with the SmLM throughout the inference process?
- **HOW** to maintain the ability of the SaLM when it has replaced the LaLM?

For the "**WHERE**" question, we introduce an innovative strategy that synergies the capabilities of both LaLM and SmLM, conceptualized in Figure 1(c). In our method, the LaLM plays the role of processing top-level question decomposition and executing the final answer summarization. This allocation leverages the LaLM's ability to understand complex queries and task planning. Meanwhile, the SaLM is responsible for interfacing with the IR systems and solving domain-specific problems through a more focused and efficient chain of thought (CoT) process (Wei et al., 2022; Xu et al., 2023).

For the "**HOW**" question,it becomes imperative to enhance the SmLM's ability in two critical aspects: nuanced problem decomposition and effective interaction with IR. In terms of problem decomposition, we have fine-tuned the SmLM by constructing data for positive and negative examples to teach the SmLM to ask IR questions and to better understand the IR returns. In terms of interacting the SmLM with the search engine, we note that the traditional IR model is a keyword-based system trained on lexical items, while the output of the language model is more oriented to natural language. We design a rewriting unit to change the natural language questions output by the SmLM to questions based on keyword forms.

We propose a scalable, learnable framework that adeptly combines large and small language models (LaLM and SmLM, respectively) to efficiently tackle multi-hop inference tasks, named Coop. CoopLLM leverages LaLM for decomposing complex questions into simpler sub-tasks, utilizes SaLM for detailed reasoning and information retrieval interactions, and re-engages LaLM to synthesize and summarize the final answers. The advantages of CoopLLM include: (1) significantly reduces computational costs by minimizing reliance on LaLM for intermediate steps; and (2) enhances answer accuracy through domain-specialized SmLM, striking a balance between efficiency and accuracy.

The major contributions of the paper include:

(1) We focus on the shortcomings of the current LLM approach to Multi-Hop QA, in terms of model efficiency and domain specialization. A novel paradigm of combining large and small LLMs for enhancing multi-hop inference tasks is proposed.

(2) We explore where and how to synergy the large and small LLMs, and propose a learnable and scalable framework that achieves a balance between efficiency and accuracy.

(3) We conducted groups of experiments on multi-hop QA datasets, and the experimental results verified the effectiveness of the proposed model.

2 Related Work

Multi-hop QA Problems with Large Language Model: Multi-hop QA means answering complex questions that require multiple steps to retrieve and reason about (Yang et al., 2018). While previous approaches have developed retrieval modules for selecting relevant passages, Q&A problems in multi-hop scenarios remain challenging due to the limited performance of one-step methods and the difficulty of decomposing complex problems (Ho et al., 2020). LLM has recently

demonstrated excellent performance in a range of downstream tasks, including in planning and question decomposition (Shao et al., 2023; Yoran et al., 2023). There is a line work exploring how to use LLM to solve Q&A problems in multi-hop scenarios; either by carefully designing prompts to stimulate the potential reasoning ability of the large language models (Zhang et al., 2023a; Khalifa et al., 2023), or by designing different thinking scenarios based on the chain-of-thinking approach to generate reasonable decomposition paths (Sun et al., 2023; Tang and Yang, 2024; Xu et al., 2023; Khot et al., 2022). SearChain (Xu et al., 2023) is one of the more representative approaches, and the problem decomposition approach in our work is an improvement based on it. SearChain operates by having the LLM generate a global reasoning chain, known as Chain-of-Query (CoQ), wherein each node comprises an IR-oriented query along with its corresponding answer. Subsequently, IR evaluates the accuracy of each node's answer in CoQ, correcting inconsistencies with retrieved information when confident, thereby enhancing credibility. By transforming the reasoning topology from a chain to a tree, SearChain has the ability to modify the direction of reasoning.

Synergy between Large Language Model and Information Retrieval: In recent years, the synergy between Large Language Model (LLM) and Information Retrieval (IR) systems has emerged as a pivotal methodology for addressing complex multi-hop queries, as evidenced by a growing body of literature including representative works by (Menick et al., 2022; Liu et al., 2023c,a). On one hand, the utilization of LLM in conjunction with IR systems capitalizes on the former's ability to parse and understand complex queries, break them into more solvable sub-queries, and then synthesize the retrieved information into coherent and contextually relevant answers. This process not only augments the precision of the answers provided but also enriches the explanation of the answers (Jeronymo et al., 2023; Saad-Falcon et al., 2023; Jeong, 2023). On the other hand, IR systems' strengths are searching a vast area of sources in real time, ensuring that the information used in the problem-solving process is not only broad-ranging but also up-to-date. Consequently, the integration of LLM with IR tools can fix LLM's shortcomings on the topics that are rapidly evolving or have sparse coverage in pre-existing datasets on which LLM are trained (Zhu et al., 2023; Liu et al., 2023b; Zhang et al., 2023b).

Notation	Description
(q, a)	a query-answer pair
M_{QA}	QA model
$\mathcal{F}_{\text{QA}}$	Subproblem solver
$\mathcal{F}_{\text{IR}}$	retrieval model
Q	original query
A	final answer
$S = \{q_i\}_{i=1}^{n}$	processed queries set
$R = \{(q_i, a_i)\}_{i=1}^{n}$	correct reasoning path
T	tree-of-reasoning

Table 1: Notations and explanations.

The IR + LLM paradigm marries the real-time data retrieval capabilities and extensive coverage of long-tail content inherent to IR with the nuanced problem decomposition and analytical strengths of LLM. Such a collaboration significantly enhances both the accuracy and interpret ability of responses in multi-hop problem-solving scenarios.

3 Task Formulation and Analysis

3.1 Task Formulation for Multi-hop QA

Depending on the underlying complexity, multi-hop QA requires identifying and reasoning about multiple related facts. Multi-hop QA often requires logical links and comparisons, e.g. to solve "*Which genus of moth in the world's seventh-largest country contains only one species?*". We need to break it down into two steps, first we need to know "*which country is the seventh largest in the world*" and then we need to solve "*which is the only moth of a species in this country*". By introducing a global QA model M_{QA}, a sub-step solver $\mathcal{F}_{\text{QA}}$, and a sub-step retriever $\mathcal{F}_{\text{IR}}$, we define the multi-hop QA problem through a three-stage workflow:

Stage 1: Top planning. This stage begins with parsing the original query Q, where QA model M_{QA} is responsible for parsing and decomposing Q into multiple sub-queries $S = \{q_i\}_{i=1}^{n}$. This step is crucial as it sets the foundation for subsequent retrieval and information integration.

Stage 2: Solving. This stage is executed by the QA model $\mathcal{F}_{\text{QA}}$ and retrieval model $\mathcal{F}_{\text{IR}}$ together. $\mathcal{F}_{\text{QA}}$ tries to solves each sub-queries $q_i \in S$ and give the answer a_i. At this point, the $\mathcal{F}_{\text{IR}}$ performs information retrieval in the document collection D for each sub-queries, aiming to find relevant text segments that can validate the answers provided by $\mathcal{F}_{\text{QA}}$. This process might be iterative until all sub-queries are ensured to be answered correctly.

Stage 3: Summarizing. The final stage involves the QA model M_{QA}, which is responsible for integrating the information that generated in the second stage to construct a comprehensive answer A. In this stage, the model may need to establish a reasoning path $R = \{(q_i, a_i)\}_{i=1}^{n}$, which are series of sub-queries that together with the answer.

In this study, we utilized both LaLM and SaLM as the QA models. This design not only exploits the superior inference and decomposition capabilities of LaLM, but also facilitates inference cost reduction and domain specialisation. We highlighted the advantages of employing smaller Language Models (LLMs) over larger ones, such as GPT, within the context of multi-hop Question Answering (QA) tasks. These advantages are notably in terms of inference efficiency and domain specialization. Nonetheless, two pivotal questions arise: (1) **WHERE** within the multi-hop QA process should LaLM be replaced by a SmLM, and (2) **HOW** can we preserve the effectiveness of the SmLM once it has taken the place of LaLM.

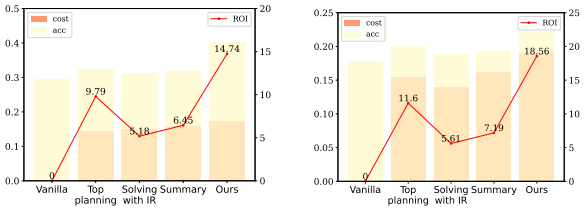
The major notations of the study are listed in Table 1.

3.2 GPT ROI Analysis of Different Stages

To address the "**WHERE**" question, we conducted a detailed analysis of the return-on-improvement (ROI) provided by LaLM across the three distinct stages of multi-hop QA, namely, initial planning, problem solving, and summarization.

In this section, we address the question of WHERE to optimally replace LaLM with SaLM within the multi-hop QA framework. To determine this, it is essential to examine the contributions and advantages of integrating LaLM at each stage of the multi-hop QA process, assessing these in terms of Return on Investment (ROI). ROI is quantified as $ROI = v_{imp}/v_{cost}$, where v_{imp} represents the performance enhancement attributed to LaLM's inclusion, as indicated by the relative improvement in multi-hop QA evaluation metrics, and v_{cost} signifies the incurred cost, measured by the number of tokens processed by LaLM.

Identifying the phase with the lowest ROI is crucial as it suggests the most advantageous stage for implementing SmLM. This substitution strategy serves a dual purpose: it not only reduces the costs associated with LaLM interactions but also minimizes the potential performance degradation in phases where SaLM is employed. We divided the problem solving steps into three major steps according to the purpose: top-level planning, solving sub-problems, and final summarization. To rigorously evaluate LaLM's ROI across different phases, we initially deploy a SaLM as the question-and-answer model throughout all three stages of multi-hop QA, establishing a baseline for performance. Subsequently, LaLM is incorporated as the QA model in each phase individually, and the ROI is calculated based on both performance improvement and cost.



(a) OpenBookQA (b) HotPotQA

Figure 2: ROI comparison of methods for incorporating LaLM at Different Stages. The yellow color indicates the average cost required to perform one interaction with the LaLM at that stage, the orange color indicates the accuracy rate, and the red dash line indicates the ROI value at that stage compared to the vanilla version.

The findings, illustrated in Figure 2, reveal that the integration of LaLM yields the lowest ROI during the problem-solving phase. This outcome is attributable to the iterative nature of this phase, which requires multiple interactions with the IR system and adjustments to the answers of sub-queries. It is more necessary to use specific knowledge, fine-tuning a SmLM to do the adaptation is a more appropriate means. Conversely, the highest ROI is observed in the initial planning phase, underscoring the foundational importance of early problem framing for the efficacy of subsequent stages.

4 CoopLLM Framework

Although we find that combining LaLM and SmLM can get good ROI, but the overall performance is lower than LaLM. In this section, we introduce a novel, learnable, and scalable framework– referred to as CoopLLM. The framework facilitates the integration of varying-sizes LLMs within the multi-hop QA workflow, which is designed to substantially lower the costs associated with LaLM interactions. In response to the second question, our target is to enhance the SmLM's ability in problem decomposition and their interaction with IR models. To achieve this, we suggest specific strategies for fine-tuning and for the development of query rewriting units. Furthermore, we explore methods

to bolster the domain-specific expertise of SmLM, thereby maximizing their utility in multi-hop QA scenarios.

Inspired by this pilot experiment, we subsequently designed a novel, learnable, and scalable framework, referred as CoopLLM.

4.1 Synergizing LaLM and SmLM

Building on our previous deliberations, we opted to implement a SaLM for the problem-solving step, while continuing to employ LaLM for both the problem planning and answer summarization phases. This strategic allocation is depicted in Figure 3. Through this architectural approach, we aim to substantially decrease the costs associated with LaLM interactions and mitigate any potential decline in performance. Additionally, this structure allows for the facile specialization of the SmLM to function as an expert model across various tasks within the solving phase. Our architecture integrates three pivotal components: the problem planning prompt, the problem solver and checker, and the answer summarization prompt.

Problem Planning Prompt: This component is crafted to harness LaLM's planning capability, steering it towards a facilitative role in problem decomposition rather than direct problem-solving. Initially, we present a structured prompt, for example, *Construct a global reasoning chain*, to LaLM along with the original query. LaLM's task is to dissect the original query into manageable sub-queries, ensuring that the decomposition maintains logical coherence and simplifies the complexity of the overarching problem. This step is crucial for setting a solid foundation for the problem-solving process, as it prepares the sub-queries in a manner that is conducive to efficient and focused solving by the subsequent components.

Problem Solver and Checker: This segment of our architecture features a specialized SaLM as the solver and an Information Retrieval (IR) model serving as the checker. We refer to Searchain (Xu et al., 2023) for the processing flow of this phase, with the difference that the full process of interacting with IR is done using SmLM. The situations to be handled by the SaLM are categorized into three types: (1) Confirmation of the answer's correctness, leading to progression to the next sub-query; (2) Identification of missing answers, indicating cases where the solver fails to provide a response, prompting the IR checker to supply the necessary information for the solver to re-engage with the problem; (3) Detection of incorrect answers, necessitating the provision of the correct answer by the IR checker to realign the solver's efforts. This iterative mechanism ensures the generation of a complete and accurate chain of reasoning across sub-queries, culminating in a comprehensive solution ready for summarization by LaLM.

Answer Summarization Prompt: The final component is tailored to leverage LaLM's summarization prowess, enabling it to constructively contribute to resolving the original complex query through a step-by-step engagement with the reasoned chain of sub-query answers. Similar to the planning phase, a specific prompt, e.g., *You can try to generate the final answer for the [Question] by referring to the following [Query]-[Answer] pairs.*, is submitted to LaLM alongside the reasoned chain. The expectation is for LaLM to utilize this structured input to synthesize the final answer, employing its summarization capabilities to integrate the details from the problem-solving process into a coherent and comprehensive response.

Algorithm 1 Description of the whole pipeline

Initialize: R, S = null; T = Q;
Output: Final Generated Answer
Function Main(Q):
  if CallLaLM(T) == true **then** ▷ Top planning
    $T \leftarrow$ ChainGenerate(T,LaLM)
  else
    $T \leftarrow$ ChainGenerate(T,SmLM)
  end if
  for (q, a)in T **do**
    if q not in S **then**
      $\tilde{T}$ = SearChain(T,(q, a)) ▷ *Solving sub-problems with IR in SearChain (Xu et al., 2023)*
      R.add(T,$\tilde{T}$)▷ *Compare the forward and backward trees and add the correct node to the path R*
      $T \leftarrow \tilde{T}$
    end if
  end for
  if CallLaLM(T) == true **then** ▷ Summary
    return AnswerSummary(Q,R,LaLM)
  else
    return AnswerSummary(Q,R,SmLM)
  end if

4.2 Specializing Smaller Language Models

In addressing the second question regarding **HOW** to augment the capabilities of SmLM for problem decomposition and their interaction with IR mod-

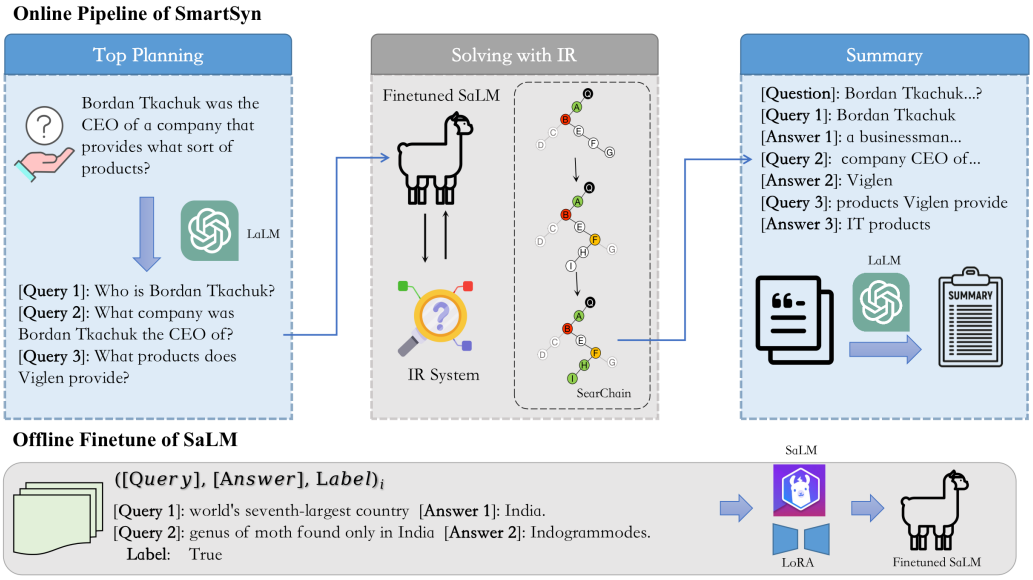


Figure 3: The overall pipeline of our method. We first let the LaLM do the top-level planning for the solution steps of the complex problem, then use the fine-tuned SaLM to interact with IR to complete the intermediate steps, and finally call the LaLM again to summarise and generate the final answer. Gray undertones indicate offline work, blue undertones indicate the need for real-time interaction with the LaLM.

els, we delineate a two-fold strategy focusing on model fine-tuning and query rewriting to enhance the interaction with IR systems.

Parameter-Efficient Fine-Tuning: We employ a widely recognized and efficient Parameter-Efficient Fine-Tuning (PEFT) technique known as Low-Rank Adaptation (LoRA), which offers an optimal compromise between resource utilization and model efficacy. To specifically enhance the SmLM's performance in the context of multi-hop problems that necessitate interaction with IR models, we have developed a customized "error book" for the SmLM. This involves utilizing a "searchain" process to identify and label both successful reasoning interaction pathways and erroneous paths that fail to lead to the correct final answer. Inspired by the training set format of LoRA, we utilize these erroneous paths as inputs and the correct reasoning pathways as outputs. The model is then fine-tuned with the prompt of "*Generate a more logical question-answer reasoning link,*" thereby improving its ability to navigate through complex reasoning tasks more effectively.

Query Rewriting Unit: After fine-tuning, we introduce a query rewriting unit aimed at refining the interaction between the SmLM and the IR system. This involves two adjustments: Firstly, for queries passed from the SmLM to the IR, we transform the natural language questions into keyword-based search terms to streamline the search process. Secondly, for content retrieved by the IR to be processed by the SmLM, recognizing the SmLM's proficiency in parsing extensive text, we increase the volume of returned documents. This expansion aims to enhance the consistency and accuracy of the SmLM's final answer by providing it with a broader context and more contents for analysis. Through these strategic enhancements, we seek to optimize the synergy between SmLMs and IR models, thereby improving the overall effectiveness and precision of the multi-hop QA process.

5 Experiments and Results

5.1 Experimental Setups

Datasets and Evaluation For the dataset we chose the classic multi-hop quiz dataset HotPotQA (Yang et al., 2018). The metric used is cover-EM (Rosset et al., 2020), which determines whether the generated answer contains the ground truth answer. This ensures objectivity in the evaluation process. Regarding the way costs are measured, we evaluate them in three dimensions. One is the average number of interactions, which can reflect the number of engagements of the LaLM in solving a multi-hop problem. For a more careful measurement, we record the average number of input and output tokens of the LaLM through binary groups

$(input\_tokens, output\_tokens)$. For a more intuitive comparison, we use the current GPT3.5-turbo charging method, i.e., 0.001/1k tokens for inputs and 0.002/1k tokens for outputs, to calculate the average cost of solving a problem.

Baselines Our baseline is primarily based on existing Searchain methods (Xu et al., 2023) for LLM+IR interaction frameworks, which are compared in terms of accuracy and interaction cost.

Implementation Details The LaLM model we used is provided by the API of an widely used chatbot. The retrieval model we used is ColBERTv2 (Santhanam et al., 2021). Regarding the SmLMs, we use a merger of LLaMA-7B-HF, LLaMA-13B-HF, LLaMA-33B-HF, and vicuña-related parameter scales publicly available on huggingface. For the experiments on small model selection, we used the open source models alpaca-7b-lora and Llama-2-7b-chat-hf on huggingface. For the training details, we use AdamW optimizer and set a batch size of 16, rank r = 4. The adapter matrices B are initialized to be zero, while the entries of A are randomly initialized using Kaiming Uniform (He et al., 2015). The maximum input sequence length was set to 1280, and efficient training was facilitated by utilizing bf16 precision. The experiments are available at https://anonymous.4open.science/r/CoopLLM-5EE0.

5.2 Main Results

Performance of our method and baselines on complex multi-hop question answering tasks are shown in Table 2. The top half of the table represents the comparison of our method with the baseline, and the bottom half represents the ablation experiments of the different components of our method. $LaLM + SmLM$ denotes the use of open source vicuña, without any fine-tuning, directly with the LaLM to solve the multi-hop problem; $CoopLLM$ denotes our approach, i.e., by fine-tuning the combination of SmLM and LaLM to solve the multi-hop problem.**(1) Reduction of interaction costs.** The three numbers following the vertical lines in the table indicate the average number of times answering a question interacts with the LaLM, the monetary cost incurred, and the number of tokens input and output, respectively. A comparison with $LaLM + SmLM^{LoRA}_{w/oDecision}$ in Table 2 shows that using a combination of small and large models, and letting the SmLM decide whether to call the LaLM model or not, this approach allows for a significant reduction in the number of interactions and the cost while ensuring that there is no significant drop in accuracy, and that this decision-making ability increases as the size of the SmLM increases. **(2) Effect of fine-tuning.** As can be seen from the comparison between the second and third rows of the table, fine-tuning of SmLMs is necessary regardless of their size. This helps the SmLM to have more domain-specific knowledge. In the multi-hop QA scenario, fine-tuning can help the SmLM to better decompose the complex question as well as better interact with the IR. Additionally through diagonal comparisons, we can see that the fine-tuned SmLM can essentially achieve what it would be possible to achieve after scaling up the scale parameters by one level over it.

5.3 Empirical Analysis

This subsection discusses the effects of the fine-tuning and rewriting units added to the use of SmLM to ensure performance.

Selection of Fine-tuning Data and Methods During the fine-tuning process, we used a labeling approach for the construction of the dataset, and used the decomposition paths that were incorrect during previous interactions with IR as negative examples, and the decomposition paths that were ultimately answered correctly as positive examples, and were fine-tuned using the PEFT approach. Comparing our method with $LaLM+SmLM^{LoRA}_{w/oLabel}$ in Table 2, it can be found that such a dataset construction method is more helpful for the SmLMs to improve their understanding of the task. In order to verify the generalization of the framework, we chose different capability-focused SmLMs and different PEFT methods, and the Table 3 shows that the fine-tuned framework improves the results independently of the model. It can be noticed that the SmLMs that focus on enhancing the ability of multi-round dialogues perform better, this is because the solution of complex problems often requires the ability to memorize and analyze the previous sub-problems. And there is no best PEFT method, it depends on the chosen SmLMs and the resource requirements.

Functional design of the rewrite unit The large language model is based on a huge amount of natural language for pre-training, but the traditional information retrieval model is still based on item to do keyword matching, so there is a certain semantic distance between the query generated by the two of them. In order to better adapt the two, we

Accuracy | $avg\_interaction$ / avg_cost / $(input\_tokens, output\_tokens)$ — Method \ SmLM		Vicuna-7B	Vicuna-13B	Vicuna-33B	GPT
Baselines	SearChain	15.7| $^{0}_{0}$ $_{(0,0)}$	18.9| $^{0}_{0}$ $_{(0,0)}$	28.9| $^{0}_{0}$ $_{(0,0)}$	52.2| $^{4.92}_{3.71e-3}$ $_{(1567,1071)}$
	LaLM+SmLM	17.6| $^{1.89}_{1.68e-3}$ $_{(675,502)}$	22.7| $^{1.93}_{1.84e-3}$ $_{(692,574)}$	29.6| $^{1.72}_{1.49e-3}$ $_{(625,432)}$	-
CoopLLM Variations	CoopLLM	21.6| $^{1.83}_{1.69e-3}$ $_{(677,506)}$	32.7| $^{1.73}_{1.57e-3}$ $_{(685,442)}$	42.9| $^{1.87}_{1.66e-3}$ $_{(681,490)}$	-
	LaLM+SmLM$^{LoRA}_{w/oLabel}$	18.1| $^{1.92}_{1.69e-3}$ $_{(671,509)}$	29.7| $^{1.76}_{1.54e-3}$ $_{(629,455)}$	32.9| $^{1.67}_{1.27e-3}$ $_{(612,329)}$	-
	LaLM+SmLM$^{LoRA}_{w/oRewriter}$	20.9| $^{1.77}_{1.60e-3}$ $_{(663,468)}$	30.0| $^{1.84}_{1.64e-3}$ $_{(678,482)}$	40.8| $^{1.79}_{1.58e-3}$ $_{(667,457)}$	-
	LaLM+SmLM$^{LoRA}_{w/oDecision}$	21.3| $^{2}_{1.92e-3}$ $_{(873,523)}$	34.5| $^{2}_{2.01e-3}$ $_{(897,556)}$	42.8| $^{2}_{1.97e-3}$ $_{(848,561)}$	-

Table 2: The overall performance of our method and other baselines in HotpotQA. Specifically, avg_cost (in USD) is calculated based on GPT3.5-turbo's current rates, billed based on the number of input/output tokens. CoopLLM Variables reports the ablation experiments of the different components of our approach. '-' means the values are not avialable.

	SearChain	LaLM+SmLM	LaLM+SmLM^{LoRA}	LaLM+SmLM^{QLoRA}
Alpaca-7B	12.1/0	14.2$|^{1.82}_{1.69e-3}$	18.5$|^{1.81}_{1.64e-3}$	19.0$|^{1.87}_{1.77e-3}$
Vicuña-7B	15.7/0	17.6$|^{1.89}_{1.68e-3}$	21.6$|^{1.83}_{1.69e-3}$	22.1$|^{1.90}_{1.83e-3}$
Llama2-7B-chat	16.2/0	17.9$|^{1.92}_{1.91e-3}$	22.3$|^{1.88}_{1.75e-3}$	21.9$|^{1.74}_{1.66e-3}$

Table 3: Effect of Various LLMs with Different Fine-tuning Methods. The content behind the vertical lines indicates, from top to bottom, the average number of interactions and the cost of interactions.Charges are based on current GPT3.5-turbo rates in USD.

Dataset	Vanilla	Keyword	Top3	Soft-prompt	History	All
openbookQA	0.296	0.315	0.304	0.306	0.309	0.401
HotpotQA	0.178	0.197	0.189	0.192	0.204	0.223

Table 4: Effects of different modules in the rewrite unit. Vicuña-7B is used as a base to modularly decouple the design of the rewrite unit. 'Keyword' means rewriting the natural language query into the form of keywords; 'Top3' means the top-3 ranked documents by IR; 'Soft-prompt' means turning the previous path into a vector embedded in the forefront of the prompt; and 'History' means the previous path in form of text to be added to prompt.

design the rewriting unit and the effect of different modules in this unit is shown in Table 4. Specifically, for IR input, rewriting the SmLM-generated query based on keywords and incorporating the interaction history of previous rounds can improve alignment. For output results, we utilized SmLM's ability to quickly understand large amounts of text and integrated the top three results returned by IR. The approach resulted in improved effectiveness.

6 Conclusion

In this paper, we explore the necessity of using a combination of small and large models in complex multi-hop QA scenarios, in terms of cost reduction and domain specialization. We analyze where small models can be substituted for large ones by decoupling the phases from an ROI perspective. A learnable traceability framework with a combination of large and small models is proposed. Specifically, we utilize large LLMs with rich world knowledge to do the top-level planning and summarization, and let the fine-tuned small LLMs complete the detailed parts. Experimental results verified the effectiveness of the proposed method. Future work includes applying the present framework to other vertical domains and investigating ways to adaptively select models of different sizes based on the problem difficulty.

7 Limitations

In this work, multi-hop QA scenarios was chosen as a research area. However, when considering other application scenarios, it is important to address the following two issues. If high accuracy is required, it may be necessary to make a trade-off between cost and effectiveness by increasing the number of times the LaLM is engaged to enhance its effectiveness. This work has limited granularity for combining LaLM and SmLM models and does not discuss the impact on cost and accuracy if LaLM is involved in solving intermediate subproblems. Secondly, SmLM performs reasonably well in multi-hop scenarios based on general knowledge, which are relatively common (Wang et al., 2023). However, if more specialized or vertical domains are chosen, the results may suffer, especially if the SmLM lacks knowledge in that domain (Zhao et al., 2023).

Second, solving problems in different application domains requires different capabilities (Valero-Lara et al., 2023), and thus may require experimentation in the choice of SmLM and PEFT methods when applied to specific scenarios (Hu et al., 2023b). To fine-tune the SmLM, it is necessary to construct the dataset based on the specific application scenario. In this work, the Searchain (Xu et al., 2023) process was used to generate correct and incorrect data. However, different scenarios may have different requirements and may require different approaches to problem decomposition.

Finally, regarding the choice of LaLM, we only experimented with one. However, LaLMs of different sizes and training styles may vary in their problem planning and summarization capabilities (Kalyan, 2023). This work proposes a fusion idea and framework for combining large and small models. When targeting specific problems, appropriate substitutions of components in the framework may be necessary.

8 Ethics Statement

In this work, we are committed to upholding the highest standards of integrity, respecting individuals' rights, and adhering to legal and ethical principles. The dataset and any developed models used in this research were sourced from openly available repositories. We acknowledge the contributions of the data providers and model creators, and we commit to adhering to any applicable licenses or usage agreements. In terms of privacy protection, we ensure that our research adheres to strict privacy standards. Any data used in this study has been anonymized and aggregated to ensure the confidentiality of individuals involved. Additionally, we have taken precautions to prevent any potential privacy breaches throughout the research process.

References

Fu Bang. 2023. Gptcache: An open-source semantic cache for llm applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 212–218.

Jiazhan Feng, Chongyang Tao, Xiubo Geng, Tao Shen, Can Xu, Guodong Long, Dongyan Zhao, and Daxin Jiang. 2023. Knowledge refinement via interaction between search engines and large language models. *arXiv preprint arXiv:2305.07402*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. 2020. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps.

Bin Hu, Chenyang Zhao, Pu Zhang, Zihao Zhou, Yuanhang Yang, Zenglin Xu, and Bin Liu. 2023a. Enabling efficient interaction between an algorithm agent and an llm: A reinforcement learning approach. *arXiv preprint arXiv:2306.03604*.

Zhiqiang Hu, Yihuai Lan, Lei Wang, Wanyu Xu, Ee-Peng Lim, Roy Ka-Wei Lee, Lidong Bing, and Soujanya Poria. 2023b. Llm-adapters: An adapter family for parameter-efficient fine-tuning of large language models. *arXiv preprint arXiv:2304.01933*.

Cheonsu Jeong. 2023. A study on the implementation of generative ai services using an enterprise data-based llm application architecture. *arXiv preprint arXiv:2309.01105*.

Vitor Jeronymo, Luiz Bonifacio, Hugo Abonizio, Marzieh Fadaee, Roberto Lotufo, Jakub Zavrel, and Rodrigo Nogueira. 2023. Inpars-v2: Large language models as efficient dataset generators for information retrieval. *arXiv preprint arXiv:2301.01820*.

Katikapalli Subramanyam Kalyan. 2023. A survey of gpt-3 family large language models including chatgpt and gpt-4. *Natural Language Processing Journal*, page 100048.

Muhammad Khalifa, Lajanugen Logeswaran, Moontae Lee, Honglak Lee, and Lu Wang. 2023. Few-shot reranking for multi-hop qa via language model

prompting. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15882–15897.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Jiongnan Liu, Jiajie Jin, Zihan Wang, Jiehan Cheng, Zhicheng Dou, and Ji-Rong Wen. 2023a. Reta-llm: A retrieval-augmented large language model toolkit.

Jiongnan Liu, Jiajie Jin, Zihan Wang, Jiehan Cheng, Zhicheng Dou, and Ji-Rong Wen. 2023b. Reta-llm: A retrieval-augmented large language model toolkit. *arXiv preprint arXiv:2306.05212*.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023c. Evaluating verifiability in generative search engines.

Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. 2023d. Summary of chatgpt-related research and perspective towards the future of large language models. *Meta-Radiology*, page 100017.

Zhengliang Liu, Aoxiao Zhong, Yiwei Li, Longtao Yang, Chao Ju, Zihao Wu, Chong Ma, Peng Shu, Cheng Chen, Sekeun Kim, et al. 2023e. Tailoring large language models to radiology: A preliminary approach to llm adaptation for a highly specialized domain. In *International Workshop on Machine Learning in Medical Imaging*, pages 464–473. Springer.

Jacob Menick, Maja Trebacz, Vladimir Mikulik, John Aslanides, Francis Song, Martin Chadwick, Mia Glaese, Susannah Young, Lucy Campbell-Gillingham, Geoffrey Irving, and Nat McAleese. 2022. Teaching language models to support answers with verified quotes.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Yujia Qin, Zihan Cai, Dian Jin, Lan Yan, Shihao Liang, Kunlun Zhu, Yankai Lin, Xu Han, Ning Ding, Huadong Wang, et al. 2023. Webcpm: Interactive web search for chinese long-form question answering. *arXiv preprint arXiv:2305.06849*.

Ori Ram, Yoav Levine, Itay Dalmedigos, Dor Muhlgay, Amnon Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. In-context retrieval-augmented language models. *arXiv preprint arXiv:2302.00083*.

Corby Rosset, Chenyan Xiong, Minh Phan, Xia Song, Paul Bennett, and Saurabh Tiwary. 2020. Knowledge-aware language model pretraining. *arXiv preprint arXiv:2007.00655*.

Jon Saad-Falcon, Omar Khattab, Keshav Santhanam, Radu Florian, Martin Franz, Salim Roukos, Avirup Sil, Md Arafat Sultan, and Christopher Potts. 2023. Udapdr: Unsupervised domain adaptation via llm prompting and distillation of rerankers. *arXiv preprint arXiv:2303.00807*.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2021. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *arXiv preprint arXiv:2302.04761*.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. *arXiv preprint arXiv:2302.00618*.

Hao Sun, Hengyi Cai, Bo Wang, Yingyan Hou, Xiaochi Wei, Shuaiqiang Wang, Yan Zhang, and Dawei Yin. 2023. Towards verifiable text generation with evolving memory and self-reflection. *arXiv preprint arXiv:2312.09075*.

Yixuan Tang and Yi Yang. 2024. Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries. *arXiv preprint arXiv:2401.15391*.

Pedro Valero-Lara, Alexis Huante, Mustafa Al Lail, William F Godoy, Keita Teranishi, Prasanna Balaprakash, and Jeffrey S Vetter. 2023. Comparing llama-2 and gpt-3 llms for hpc kernels generation. *arXiv preprint arXiv:2309.07103*.

Cunxiang Wang, Xiaoze Liu, Yuanhao Yue, Xiangru Tang, Tianhang Zhang, Cheng Jiayang, Yunzhi Yao, Wenyang Gao, Xuming Hu, Zehan Qi, et al. 2023. Survey on factuality in large language models: Knowledge, retrieval and domain-specificity. *arXiv preprint arXiv:2310.07521*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Shicheng Xu, Liang Pang, Huawei Shen, Xueqi Cheng, and Tat-Seng Chua. 2023. Search-in-the-chain: Towards accurate, credible and traceable large language models for knowledge-intensive tasks.

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*.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. *arXiv preprint arXiv:2304.13007*.

Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Yong Liu, and Shen Huang. 2023a. Beam retrieval: General end-to-end retrieval for multi-hop question answering. *arXiv preprint arXiv:2308.08973*.

Jianzhang Zhang, Yiyang Chen, Nan Niu, and Chuang Liu. 2023b. A preliminary evaluation of chatgpt in requirements information retrieval. *arXiv preprint arXiv:2304.12562*.

Yating Zhang, Yexiang Wang, Fei Cheng, Sadao Kurohashi, et al. 2023c. Reformulating domain adaptation of large language models as adapt-retrieve-revise. *arXiv preprint arXiv:2310.03328*.

Wenting Zhao, Justin T Chiu, Jena D Hwang, Faeze Brahman, Jack Hessel, Sanjiban Choudhury, Yejin Choi, Xiang Lorraine Li, and Alane Suhr. 2023. Uncommonsense reasoning: Abductive reasoning about uncommon situations. *arXiv preprint arXiv:2311.08469*.

Yutao Zhu, Huaying Yuan, Shuting Wang, Jiongnan Liu, Wenhan Liu, Chenlong Deng, Zhicheng Dou, and Ji-Rong Wen. 2023. Large language models for information retrieval: A survey. *arXiv preprint arXiv:2308.07107*.