`Self-DC`: When to Reason and When to Act Self Divide-and-Conquer for Compositional Questions

Hongru Wang[αβ†], **Boyang Xue**[αβ†], **Baohang Zhou**[γ], **Tianhua Zhang**[α], **Cunxiang Wang**[σ], **Huimin Wang**[μ], **Guanhua Chen**[δ‡], **Kam-Fai Wong**[αβ‡]
[α]The Chinese University of Hong Kong [γ]Nankai University
[β]MoE Key Lab of High Confidence Software Technologies, CUHK
[σ]Westlake University [μ]Jarvis Research Center, Tencent YouTu Lab
[δ]Southern University of Science and Technology
{hrwang, kfwong}@se.cuhk.edu.hk chengh3@sustech.edu.cn

Abstract

Previous research has typically concentrated on leveraging the internal knowledge of Large Language Models (LLMs) to answer known questions (i.e., *internal reasoning such as generate-then-read*). In contrast, for questions that fall outside their known scope, these models rely on external knowledge retrieval to provide accurate responses (i.e., *external acting such as retrieve-then-read*). However, few previous works consider the *compositional questions*, which consist of several known and unknown sub-questions, necessitating the dynamic combination of previous two methods (i.e., *internal reasoning and external acting*) to achieve a better trade-off between effectiveness and efficiency. To this end, we introduce a **Self** **D**ivide-and-**C**onquer (`Self-DC`) framework, accompanying with the first **C**ompositional **u**nknown **Q**uestion-**A**nswering dataset (CuQA). This framework enables LLMs to adaptively choose between using internal knowledge and retrieving external knowledge as needed, resulting in a better trade-off between effectiveness and efficiency. Experimental results on two datasets demonstrate that `Self-DC` can achieve comparable or even better performance with much fewer external calls compared with several strong baselines.

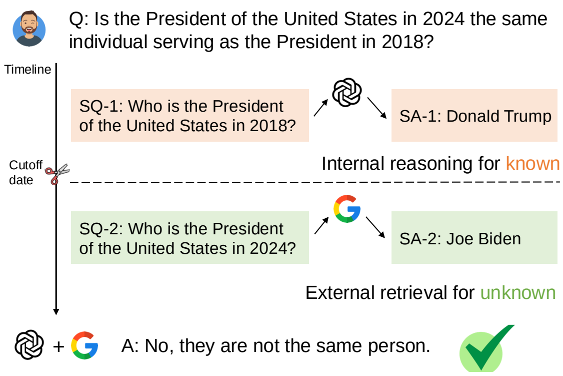

Figure 1: A example of compositional questions, in which a unknown question consists of some sub-questions can be answered using *known* knowledge while other sub-questions necessitate *unknown* knowledge according to the cutoff date of LLMs.

1 Introduction

Large Language Models (LLMs) (Ouyang et al., 2022; Touvron et al., 2023) possess extensive world knowledge thanks to the scaling of size of pre-training data and model (Kaplan et al., 2020), resulting in exceptional capabilities to answer open-domain questions using internal known knowledge encoded in their parameters (Yu et al., 2023a; Bang et al., 2023). However, due to the cutoff date of training data, it is difficult for them to answer questions out of their known knowledge (a.k.a., unknown questions), which necessitates the augmentation of external retrieval (Lewis et al., 2021; Zhuang et al., 2023; Vu et al., 2023; Gabburo et al., 2024), such as Google Search and Wikipedia.

To provide more accurate answers for the questions, most previous works tend to employ external retrieval methods indiscriminately without considering different types of questions, resulting in redundant retrieval and unnecessary cost (Trivedi et al., 2023; Shao et al., 2023). Alternatively, some methods simply classify questions into binary categories (i.e., known and unknown), and utilize either self-generated context or retrieved external context to answer them, respectively (Wang et al., 2023d), following a generate-then-read (Yu et al., 2023a) or retrieve-then-read (Lewis et al., 2021) paradigm. However, this binary classification is sub-optimal and inefficient for handling *compositional questions*, which consist of multiple sub-questions where each sub-question could be known or unknown, as illustrated in Figure 1. Consequently, these binary-classification methods degrade into simply retrieving information for ev-

† Equal Contributions
‡ Co-corresponding Authors

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), pages 6510–6525
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

ery question, as any compositional questions containing an unknown sub-question remain entirely unknown by large language models (LLMs). Moreover, using the original compositional question as a query frequently leads to the retrieval of noisy or unrelated documents, which hinders accurate answers (Ma et al., 2023). These limitations highlights the need for more nuanced and efficient retrieval strategies tailored to the complexity of *compositional questions*.

In this paper, we first formally introduce *compositional questions* from the perspective of known/unknown, which is more practical and challenging. To further specify the *compositional questions*, we categorized questions into four types according to the knowledge boundaries of LLMs [1]:

- ***Single Known.*** The question contains no sub-questions and can be solved using internal knowledge of LLMs, such as with the generate-then-read method.
- ***Single Unknown.*** The question contains no sub-questions and can only be solved using external knowledge, such as with the retrieve-then-read method.
- ***Compositional Known.*** The question contains several sub-questions, and each sub-question is *Single Known.*
- ***Compositional Unknown.*** The question contains several sub-questions, and at least one sub-question is *Single Unknown.*

Determining whether a question is known or unknown to LLMs, and whether it is a compositional question, is a complex task that may require multi-step reasoning. In this paper, we introduce a **Self D**ivide-and-**C**onquer (`Self-DC`), designed to effectively and efficiently identify and decompose compositional questions. The main idea of `Self-DC` is to use the inherent signals of LLM to control its own behavior, e.g., elicit the internal knowledge or call external retrieval. Specifically, we define each action as a function, and model the whole decomposition as dynamic function calls guided by self-aware confidence signals. Therefore, the internal reasoning capabilities of LLMs can be well elicited while making every external retrieval call count. In summary, our contributions can be outlined as follows:

- To the best of our knowledge, we are the first to study *compositional questions* from the perspective of known / unknown.
- We introduce an automatic data collection pipeline to create the first **C**ompositional **u**nknown **Q**uestion **A**nswering dataset (CuQA), serving as an important evaluation benchmark for LLMs in known/unknown.
- We present a flexible and robust *`Self-DC`* framework, which is capable of adaptively calling different functions on-demand for *compositional questions* decomposition.
- Experimental results on CuQA and FreshQA (Vu et al., 2023) datasets show the superiority of `Self-DC` in terms of both effectiveness and efficiency, revealing its promising potential to solve compositional reasoning problem.

2 Related Work

Known and Unknown of LLMs. Investigations into the known and unknown boundaries of large language models (LLMs) have gained attention in recent literature (Kadavath et al., 2022; Amayuelas et al., 2023; Yin et al., 2023). Despite the parameters of LLMs containing a wealth of knowledge to excel in various tasks, they are still limited due to the continuously increasing information. Specifically, LLMs have showcased satisfactory performance to evaluate the validity of their own claims and predict which questions they will be able to answer correctly by predicting "P(IK)", the probability that "I know" the answer to a question (Kadavath et al., 2022). Furthermore, Yin et al. (2023) evaluate LLMs' self-knowledge by assessing their ability to identify unanswerable or unknowable questions. Similarly, Amayuelas et al. (2023) further assesses the LLMs' ability to differentiate between known and unknown questions and classify them accordingly by collecting Known-Unknown Questions (KUQ). Their results show that the LLMs still have room for improvement in classifying known-vs-unknown questions, even with the incorporation of retrieval augmentation (Ren et al., 2023). More recently, Xue et al. (2024a) utilize both semantic entropy and confidence signal to guide the behaviors of LLMs for known and unknown questions. Distinguished from previous

[1]The definition begins from the data side instead of model side such as the cutoff date of training data, we discuss hallucination issue of model side at Sec 8.

works, our paper targets *compositional questions*, considering various types of questions in practice.

Certainty and Uncertainty of LLMs. To calibrate the known and unknown of LLMs, there are lots of studies that have delved into methods for estimating and quantifying *certainty and uncertainty* in LLMs predictions (Xiao et al., 2022; Lin et al., 2022; Xiong et al., 2023; Kuhn et al., 2023). There are two types of methods: 1) logit-based which utilize the model logits (Xiao et al., 2022; Mielke et al., 2022); and 2) non-logit-based methods, such as expressing uncertainty about its own answer in natural language (Lin et al., 2022), particularly with the rise of closed-source LLMs. More recently, Xiong et al. (2023) benchmarks three categories of the first type: verbalize-based, consistency-based, and their hybrid methods. They find that LLMs exhibit a high degree of overconfidence when verbalizing their confidence, which can be alleviated by different prompting strategies (e.g., Chain-of-thoughts (Wei et al., 2023)) or more complicated methods (e.g., Self-consistency (Wang et al., 2023c)). Moreover, different languages also trigger different level of certainty and uncertainty of language models (Xue et al., 2024b).

Reasoning and Acting of LLMs. On the one hand, lots of previous methods investigate various methods to elicit the internal reasoning capability of LLMs (Wei et al., 2023; Wang et al., 2023b, 2025), such as program-guided reasoning (Pan et al., 2023; Khattab et al., 2023), Self-Ask (Press et al., 2023) and retrieval-augmented reasoning (Trivedi et al., 2023; Yu et al., 2023b; Shao et al., 2023), especially for multi-hop questions (Yang et al., 2018) and in-depth dialogues (Wang et al., 2023b). On the other hand, it is important to empower the stateless LLMs to interact with external world with the augmentation of different tools (Wang et al., 2024a). Therefore, LLMs can perform tasks that go beyond their intrinsic knowledge such as retrieving up-to-date information (Wang et al., 2023a, 2024c) and providing domain-specific services by calling different functions / APIs (Wang et al., 2024b). However, only a few of them consider the relationship between internal reasoning and external acting, especially for compositional problems when the necessary unknown knowledge is required. To address this dilemma, we explore the better trade-off between internal reasoning and external acting in terms of effectiveness and efficiency.

3 Data Collection

In this section, we thoroughly introduce how to collect the **C**ompositional **u**nknown **Q**uestion-**A**nswer dataset (CuQA) automatically, with the minimum human efforts to filter unqualified samples.

3.1 Automatic Collection

Algorithm 1 shows the pseudo-code details. Specifically, we assume there is a cutoff date for each LLM with the latest cutoff date for all LLMs, and all the pretraining corpus is collected before the cutoff date, for example, the cutoff date of `gpt4-turbo` is April 2023[2]. In this way, we first collect all events that happened **after** the cutoff date from Wikipedia[3], named unknown events. Then we carefully implement different functions (i.e., UnknownQuestionGen) by prompting LLMs using different templates. We provide different information, e.g., internal known events and external unknown events, in the template to guide LLMs in generating the required output. For example, we use one entity in the unknown events as an answer and prompt the LLMs to generate corresponding questions according to the events (line 6). Appendix A.1 shows the details of all functions' prompts. We finally store the questions, answers, and all intermediate results for further processing [4].

3.2 Quality Control and Statistics

To ensure the quality of the dataset, we additionally introduce some automatic quality control procedures and human evaluations. First of all, we write a Python script to validate whether or not the format of outputs meets the instructions in the functions. Moreover, we employ three well-educated annotators to: 1) filter unqualified samples (≈10%), such as answer is not correct or can not be inferred according to unknown events; and 2) rewrite the generated question to be more natural. Afterward, we successfully collect around 550 questions. It is worth noting 100 of them are hard questions which are further composed using multiple easy question-answering pairs (line 13). The examples in the data can be found in Appendix A.2.

[2]https://openai.com/blog/new-models-and-developer-products-announced-at-devday

[3]https://en.wikipedia.org/wiki/2023

[4]It is worth noting that our data collection can be time-evolving given the cutoff date.

Algorithm 1 CuQA Generation Algorithm

Input: Cutoff date t, Wikipedia W, LLM $\mathcal{M}$
Output: Generated Questions $\mathcal{Q}$

$U_e = W(t)$ // get the unknown events according to the cutoff date of LLMs
for $e_j \in U_e$ **do**
  ent_j = getEntities(e_j) // get a list of entities
  cur_{ent} = random.sample(ent_j)
  uk_q = UnknownQuestionGen(e_j, cur_{ent})
  k_e = KnownEventsGen(cur_{ent})
  if random.randint(1,9) < 5 **then**
    k_q = KnownQuestionGen(e_j, cur_{ent})
  else
    k_{ent} = random.sample(getEntities(k_e))
    k_q = KnownQuestionGen2(k_e, k_{ent}, cur_{ent})
  end if
  q = MergeQuestions(k_q, uk_q)
  $\mathcal{Q}$.append((q, cur_{ent} or uk_q))
end for
return $\mathcal{Q}$

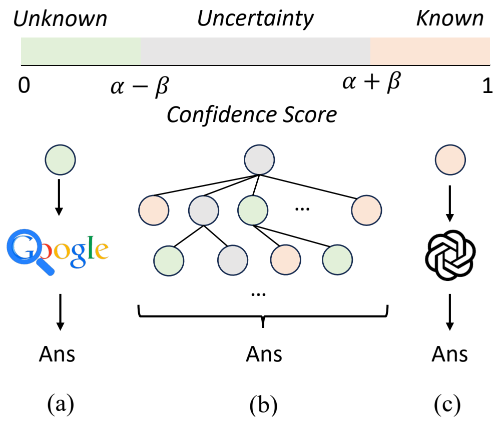

Figure 2: Overview of `Self-DC`: a) retrieve-then-read for unknown questions, b) decompose-and-combination for uncertain questions; and c) generate-then-read for known questions.

4 Method

To adaptively call different functions on-demand for *compositional questions* understanding, it is essential to determine: **a)** whether the current question is known or unknown to the LLMs, and **b)** whether the current question can be further decomposed into different sub-questions. Therefore, given a question, we first get the confidence score of LLMs for a question and then (iteratively) call different functions, aiming to collect enough information to generate the final answer. Figure 2 shows an overview of the proposed self divide-and-conquer framework, `Self-DC`.

4.1 Framework: Self Divide-and-Conquer

Since LLMs express certainty in different ways and are prone to hallucination issues, therefore, we define α as a mean of confidence score distribution for specific LLM, along with β as the corresponding standard deviation. In this way, the LLMs can recognize when a question might be too complex or ambiguous for a straightforward answer, necessitating the decomposition into simpler parts or the combination of multiple pieces of information. Specifically, we divide the confidence score into three ranges $[0, \alpha-\beta]$, $(\alpha-\beta, \alpha+\beta)$, $[\alpha+\beta, 1]$. When the confidence score falls into extreme ranges, such as the left ($[0, \alpha-\beta]$) or right ($[\alpha+\beta, 1]$) side, we can directly apply retrieve-then-read or generate-then-read to answer the question respectively. However, when it encounters uncertain or confusing questions (i.e., fall into the middle part), we decompose the question into several sub-questions to decrease the uncertainty. We then iteratively solve these sub-questions in the same way and combine all sub-answers to answer the original *compositional question* as shown in Figure 3. To ensure efficiency and reduce unnecessary costs, we implement several pruning conditions to prevent iterations from overflowing: 1) the number of sub-questions is 1, which means it should be a *Single Known* or *Single Unknown* question; and 2) the number of iteration depth is less than a pre-defined τ. Once these situations happen, we simply regard the current sub-question as the unknown question and then call retrieve-then-read. In this way, we can call compositional reasoning when necessary instead of treating all questions indiscriminately for different LLMs.

4.2 Confidence Score Acquisition

Inspired by lots of previous works (Lin et al., 2022; Xiong et al., 2023), we use two types of method to prompt the LLM itself to get the confidence score to answer the question.

- ***verbalize-based*** *(verb)*. We instruct the LLMs to output the confidence level from 0 to 100 following the answer to the question (Xiong et al., 2023). We clearly note that the confidence level indicates the degree of certainty. Then we remap the confidence score to the range $[0, 1]$. The details of the prompt can be found in Appendix.
- ***probability-based*** *(prob)*. We additionally utilize

the probability information to calculate the confidence score. Specifically, we firstly prompt the LLMs to generate the answer using a few words, and then we get the probability $\hat{p}_i$ of i-th token in the generated content. We take the average of probabilities in the sequence as the confidence score (Varshney et al., 2023) following Eq. 1:

$$conf = \frac{1}{N} \sum_{i=1}^{N} \hat{p}_i \tag{1}$$

Considering the poor performance of LLMs to express uncertainty as reported by lots of existing works (Lin et al., 2022; Xiong et al., 2023) and complex situations in practice, we additionally introduce α and β to control the range of uncertainty, enhancing the flexibility and robustness of `Self-DC`.

4.3 Other Sub-Functions

According to different levels of confidence scores, we carefully design several functions to complete the compositional reasoning task, aiming to provide a more accurate answer. We present the details of other sub-functions one by one as follows:

- **Generate-then-read:** Following Yu et al. (2023a), we firstly prompt the LLM to generate a background document from Wikipedia to answer the given question, and then ask the LLM to answer the question by referring to the generated passage. The prompt details can be found in the original paper.
- **Retrieve-then-read:** We utilize the retriever to retrieve external knowledge at the first step and then ask the LLM to answer the question by referring to the retrieved passage.
- **Decompose:** We prompt the LLMs to systematically break down the overarching question into several smaller sub-questions. The answers to these sub-questions collectively contribute to deriving the answer to the original overarching question, similar to Press et al. (2023) and Xu et al. (2023).
- **Combine answers:** After the decomposition, we call the main function to enter the next iteration as shown in Figure 3, aiming to get the answer to each sub-question. Subsequently, we combine the answers to all sub-questions to get the answer to the original question.

```
def SelfDC(m, r, q, alpha, beta):
    # m: large language model
    # r: retriever for searching documents
    # q: question to be answered
    # alpha, beta: hyperparameters for defining ranges

    c = get_confidence_score(m, q)

    if c < alpha + beta and c > alpha - beta:
        sub_qs = decompose(m, q)
        sub_as = [SelfDC(m, r, sub_q, alpha, beta) for
            sub_q in sub_qs]
        answer = combine_sub_qas(m, q, sub_qs, sub_as)
    elif c >= alpha + beta:
        answer = generate_then_read(m, q)
    else:
        answer = retrieve_then_read(m, r, q)

    return answer
```

Figure 3: The simplified python implementation details of `Self-DC`, consisting of several functions: 1) *decompose*; 2) *combine-sub-qas*; 3) *generate-then-read*; and 4) *retrieve-then-read*.

5 Experiment

5.1 Baselines and Evaluation Metrics

Baselines. To provide a comprehensive evaluation, we compare our method with different prompting methods with or without the involvement of retrieval augmentation: 1) **Direct Prompting** (Brown et al., 2020); 2) **Chain-of-thought (CoT) prompting** (Wei et al., 2023), including zero-shot and few-shot setting; 3) **GenRead** (Yu et al., 2023a) which firstly prompts the LLMs to generate known knowledge and then answer the question; 4) **Retrieve-then-read (RR)** which retrieves the related passages first and then answers the questions, following Yu et al. (2023b); 5) **Self-Ask** (Press et al., 2023) involves generating follow-up questions, retrieving information based on those questions, and providing answers, until no more follow-up questions are generated and the LLMs answer the original question at the last; 6) **IRCoT** (Trivedi et al., 2023) interleaves retrieval with steps (sentences) in a CoT, guiding the retrieval with CoT and in turn using retrieved results to improve CoT; 7) **REFEED** (Yu et al., 2023b) and 8) **ITER-RETGEN** (Shao et al., 2023) utilize the generated answer or intermediate reasoning results to enrich the query, leading to better retrieval and final answer to original question, respectively.

Datasets and Evaluation Metrics. We conduct our experiments mainly on two datasets: 1) the newly proposed CuQA dataset; and 2) FreshQA (Vu et al., 2023), which contains 600 question-answer pairs that require fast-changing world knowledge, including the latest ones [5]. We note here that FreshQA is not a typical compositional QA dataset despite it containing few *compositional questions*. To select suitable values for α and β, we randomly sample 50 instances as a development set for CuQA, leaving 500 instances for testing. For FreshQA, we use the original split: 500 test instances and 100 development instances. Following previous works (Yu et al., 2023a,b; Trivedi et al., 2023), we select Exact Match (EM)[6], F1 to evaluate the performance of different methods. Furthermore, to enhance the robustness of the evaluation, we use Acc† as an additional metric and prompt LLMs to assess the predictions related to the actual ground-truth answers following Shao et al. (2023).

5.2 Implementation Details

We mainly conduct our experiments on two different backbone models: `gpt-3.5-turbo-1106` and `gpt-4o-mini`, hereinafter referred to as `1106` and `4o-mini` respectively, following lots of previous works (Yu et al., 2023a,b; Shao et al., 2023). For the Acc† evaluation, we always use `4o-mini` as evaluation backbone model. We set both the temperature and top p as 0.1 to reduce the randomness of LLMs for all methods, rendering a more fair comparison. We implement the Google search engine following `LangChain` [7] as an external retriever, and we set the number of retrieved results as 3 and the max iteration depth τ as 3. According to the preliminary results on the validation set, we fix β as 0.1 and α as 0.9 for verb (0.8 for prob) on `1106` for both datasets, and α as 0.6 for verb (0.6 for prob on CuQA; 0.8 for prob on FreshQA) on `4o-mini`. The significant test (t-test) is conducted with $p < 0.05$ to ensure statistical improvement.

5.3 Main Results

Table 1 and Table 2 show the performances of all baselines and our proposed *`Self-DC`* on the `1106` and `4o-mini` respectively. Therefore, several conclusions can be drawn from the results:

CoT (or Few-shot-CoT) does not bring consistent improvements over direct prompting (Direct). We surprisingly found that the performance of CoT at both Table 1 and Table 2 is usually worse than Direct, and Few-shot-CoT can not further boost the performance particularly with `1106`, revealing the complexity of compositional reasoning.

Methods	#R	CuQA			FreshQA		
		EM	F1	Acc†	EM	F1	Acc†
w/o retrieval							
Direct	0	21.0	19.3	34.2	20.6	21.6	37.6
CoT	0	21.8	20.5	36.6	21.2	22.9	38.8
Few-shot-CoT*	0	7.2	1.7	9.6	18.0	11.1	26.8
GenRead	0	12.2	12.6	23.2	18.8	19.3	36.0
w/ retrieval							
RR	n	30.4	**24.7**	48.2	34.2	**28.9**	<u>61.6</u>
REFEED	$2n$	<u>35.2</u>	8.2	**53.2**	29.6	16.1	49.2
IRCoT	$3n$	**39.0**	8.1	50.4	32.0	15.5	61.2
Self-Ask*	0-n	8.6	4.3	11.2	16.8	13.4	27.4
ITER-RETGEN*	$2n$	19.2	5.8	25.4	32.4	15.7	46.6
Self-DC (*verb*)	0-$2n$	31.8	20.4	49.4	<u>34.3</u>	25.2	58.1
Self-DC (*prob*)	0-n	32.6	<u>21.7</u>	<u>50.6</u>	**36.2**	<u>28.4</u>	**62.2**

Table 1: The performance of baselines and `Self-DC` with the `1106`. The baseline* means it uses demonstrations and The column **#R** denotes the number of retrieval calls in terms of number of test cases n. We **bold** the best performance and <u>underline</u> the second-best performance.

Retrieval-based method generally achieves better performance than non-retrieval methods but the gap is smaller with compositional questions. It is observed that RR and IRCoT are capable of achieving better performance than non-retrieval baselines, and IRCoT sometimes achieves the highest performance due to a more complex retrieval design, accompanied by more cost. Secondly, the gap between retrieval-based and non-retrieval-based methods on FreshQA is relatively larger than on CuQA. This discrepancy is likely because CuQA contains more *compositional questions*, which, when used directly as queries, result in noisier documents. Furthermore, we surprisingly observe that Self-Ask and ITER-RETGEN achieve the lowest performance, especially on CuQA. To understand the reason, we examined the intermediate reasoning results and found that Self-Ask tends not to generate follow-up questions and directly answer the question, rarely calling for retrieval given the compositional unknown question. On the other hand, ITER-RETGEN retrieves external documents step-by-step but introduces a lot of noise since the queries are mostly related to the original *compositional question*. These observations reveal the significance and valuable insights provided by the CuQA dataset, highlighting its importance for understanding the challenges associated with *compositional questions*.

[5] We use the version on 30th Sep, 2024.

[6] We consider it is matched when the predicted answer in the ground truth answer due to various outputs by LLMs.

[7] https://python.langchain.com/docs/integrations/tools/google_search

Methods	#R	CuQA			FreshQA		
		EM	F1	Acc†	EM	F1	Acc†
w/o retrieval							
Direct	0	29.0	19.4	46.4	27.2	17.3	53.0
CoT	0	28.8	18.2	46.0	29.2	18.1	53.8
Few-shot-CoT*	0	43.0	3.2	50.8	35.0	9.1	55.4
GenRead	0	29.6	29.2	47.4	26.8	27.7	52.0
w/ retrieval							
RR	n	32.0	31.6	55.4	35.2	32.6	63.4
REFEED	$2n$	26.2	33.5	51.8	28.8	34.5	57.4
IRCoT	$3n$	**47.8**	13.5	**64.6**	34.2	17.8	61.4
Self-Ask*	0-n	19.8	3.8	48.4	5.6	9.8	59.0
ITER-RETGEN*	$2n$	23.4	12.6	50.9	31.2	21.1	55.8
Self-DC (*verb*)	0-n	34.0	32.2	53.8	30.2	30.2	59.8
Self-DC (*prob*)	0-n	36.4	**36.5**	56.4	**37.4**	**36.6**	**66.4**

Table 2: The performance of baselines and `Self-DC` with the `4o-mini`.

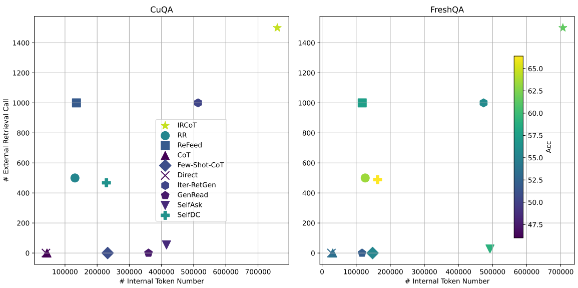

Figure 4: The efficiency analysis of different methods using `4o-mini`.

***`Self-DC`* achieves better trade-off between efficiency and effectiveness than retrieval-based methods.** When comparing `Self-DC` to other baselines considering the consumption of retrieval calls (#R), it is evident that `Self-DC` achieves better performance compares with the method utilizing same or more calls, for example, `Self-DC` (prob) v.s. RR. Even compared with some methods that require 2 to 3 times more retrieval, `Self-DC` still achieves comparable results and even outperforms them in specific dataset. This is important to highlight, as it not only establishes an effective and efficient framework to call external retrieval, but also demonstrates a promising path for controlling the behavior of LLMs by leveraging the internal signals they generate (i.e., the internal confidence scores).

6 Analysis

In this section, we present a comprehensive analysis of `Self-DC` mainly using the CuQA dataset, covering three key aspects: efficiency analysis, the choices of α and β and different iteration depth on latest model `gpt-4o-mini`.

6.1 Efficiency Analysis

To directly validate the efficiency of `Self-DC`, we consider three dimensions: # internal token consumption, # external retrieval calls and the final performance. Table 4 illustrate the report. Ideally, we aim for a method which achieves the best performance appears at the left bottom of figure. Only in such a case, the method would demonstrate its superiority by not only delivering better performance but, more importantly, by eliciting the great potential of the internal capabilities of LLMs and minimizing reliance on external resources or tools. According to the figure, it is obvious that `Self-DC` achieves great balance between these three factors. It is worthy noting we observe similar trends on `1106` for both datasets.

6.2 The Impacts of Different α and β

It is vital to balance alpha and beta for optimizing the performance of LLMs to different tasks. In this section, we provide detailed analysis of different choices of α and β. Firstly, we fix $\beta = 0.1$ and set α to $[0.1, 0.2, 0.3, ..., 0.9]$. The results can be found in Figure 5. The entire processing can be seen as a 0.2-length uncertainty block starts from 0 to 1 with stride = 0.1. First of all, We found that none of the lines shows monotonically increasing or decreasing, and most of the best performances are achieved in the middle choice of α, revealing the complexity of the target problem. In detail, there is an upward and then downward trend globally (e.g., in the right figure). It is reasonable since LLMs utilize more generate-then-read functions at the beginning (e.g., α=0.1, β=0.1), resulting in poor performance. With the uncertainty, blocks move to the right side (a.k.a, 1), LLMs will utilize retrieve-then-read more frequently. Once exceeds a specific threshold, the performance will drop since the decomposition will introduce more noise compared with gains.

Secondly, we fix α with different values according to the best performance above and set β to $[0.1, 0.2, ..., 0.5]$ to investigate the impacts of different β. Figure 6 shows the final results. It is obvious that there is a monotonically decreasing trend. After carefully checking the specific confidence scores distributions, we attribute this to be smaller range changes in the score. In general, despite that the choices of α and β are extremely tricky with lots of factors in practice, we humbly point out that most of simply combination (e.g., α=0.5, β=0.1)

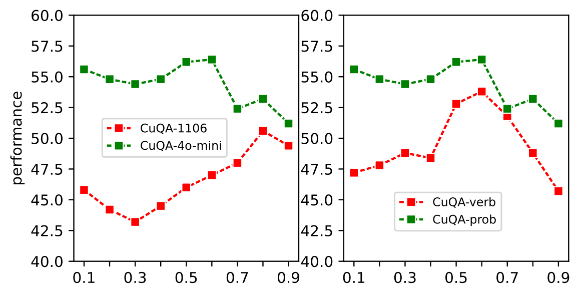

Figure 5: The performance of different choices of α with $\beta = 0.1$. **Left:** The performance of different models with confidence type is *prob*; and **Right:** The performance of different confidence types (*verb* or *prob*) with the same model `4o-mini`.

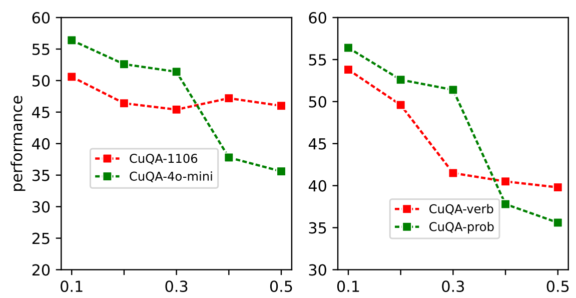

Figure 6: The performance of different choices of β with a fixed α as 0.8 for `1106` and 0.6 for `4o-mini`. **Left:** The performance of different models with confidence type is *prob*; and **Right:** The performance of different confidence types (*verb* or *prob*) with the same model `4o-mini`.

achieves comparable performance with baselines require more retrieval or token consumption even it may not be optimal combination.

6.3 The Impacts of Different Iteration Depth

Table 3 shows the results. First of all, we can find that different choices of τ have a slight effects on the final performance. As the iteration depth increases, the number of retrieval calls rises correspondingly, as noted in prob, while verb remains largely unchanged. We suspect this is due to *verb* is not as accurate as *prob*. In this way, it calls almost all external retrieval for unknown questions only within the shallow iteration. Most importantly, we want to emphasise here that the number of retrieval calls usually will not exceed the number of original test set n, and sometimes it only need to call less than $0.5n$ calls, revealing the great advantages of *`Self-DC`* over other iterative retrieval-augmented baselines.

Model	**2**	**3**	**4**
`4o-mini` (*verb*)	50.2 (76)	53.8 (78)	53.4 (78)
`4o-mini` (*prob*)	52.4 (455)	56.4 (468)	55.3 (470)

Table 3: The performance of `Self-DC` with different max iteration times. We also report the number of retrieval times in the (bracket).

Types	**EM**	**F1**	**Acc**†
Easy	38.1	37.6	58.8
Hard	26.7	22.7	33.3

Table 4: The performance of `Self-DC` on two types of question: *easy* and *hard* in CuQA using `4o-mini`.

6.4 Error Analysis

Performance of different types of questions. Table 4 shows the results of different types of questions in CuQA. There exists a significant disparity in performance between easy and difficult questions, indicating a substantial challenge for models when addressing complex compositional unknown questions. Upon analyzing the error cases, we identified several prevalent issues: 40% of errors arise from repetitive sub-questions, 13% are due to irrelevant or incorrect sub-questions, such as "*What month is it now?*", another 13% involve correct decomposition but incorrect answers.

Accuracy of confidence scores. First of all, when using *verb* method, we find that the confidence scores are 0 for more than 65% cases, and over 0.9 for around 20% cases with `1106`. However, the trend is slightly different when it comes to `4o-mini` which gives 0.9 more frequently ($\approx$ 35%). These two scores represent the top two most frequently occurring scores in both models. It seems LLMs either overestimate the correctness, or directly acknowledge the uncertainty and refuse to answer. Moreover, there is pretty rare of fine-grained confidence score (i.e, 0.82, 0.61), making the fine-grained choices of β meaningless in *verb*. On the other hand, when using *prob* method, there are much more fine-grained confidence signals, and most of them falls in the < 0.5 part ($\approx$ 90%). It is clear that *prob* leads to better performance compared with *verb* and generally `4o-mini` outperforms `1106`.

Analysis of decomposition The times of decomposition are highly affected by the confidence scores. Sometimes, the best performance can be achieved without any decomposition with well-

Q: Is the country known for its diverse wildlife and landscapes, including the Great Barrier Reef, Uluru (Ayers Rock), and the Sydney Opera House, the same as the country hosted the 2023 FIFA Women's World Cup ?

A: Yes

Q: What country is known for its diverse wildlife and landscapes, including the Great Barrier Reef, Uluru (Ayers Rock), and the Sydney Opera House? Generate-then-read

A: Australia

Q: Which country hosted the 2023 FIFA Women's World Cup? Generate-then-read

A: Australia and New Zealand

Figure 7: An example from CuQA dataset where one *compositional question* can be further divided into two known sub-questions.

Q: How many years did it take for the population of the world to reach 8 billion from 7 billion?

A: 11 Years

Q: When did the world population reach 7 billion? Generate-then-read

A: The world population reached 7 billion in 2011, specifically on October 31st.

Q: What year did the world population reach 7 billion?

A: The world population reached 7 billion in 2011.

Q: What was the specific date when the world population reached 7 billion?

A: October 31, 2011.

Q: When did the world population reach 8 billion?

A: November 15, 2022. Retrieve-then-read

Figure 8: An example from FreshQA dataset where one *compositional question* can be further divided into known and unknown questions.

selected α and β. We study the percentage of decomposition and how many original questions are answered correctly after decomposition. We found that 0% (i.e., α=0.9) to 80% (i.e., α=0.1) questions will be decomposed and it is about 40% to 50% questions are correctly answered after decomposition [8].

7 Case Study

Figure 7 and Figure 8 show an example from CuQA and FreshQA dataset respectively. We can found that `Self-DC` is capable to call different functions to address various *compositional questions* including known and unknown sub-questions.

8 Conclusion

In this paper, we firstly introduce compositional unknown questions, which contain several known and unknown sub-questions. We build a benchmark, named CuQA, to evaluate the performance and efficiency of existing compositional reasoning methods. Furthermore, we present a Self Divide-and-Conquer (`Self-DC`) method to adaptively call external or internal knowledge, which not only demonstrates comparable or even better performance compared with existing complex iterative retrieval methods with fewer retrieval calls but also shows a promising potential to elicit internal capabilities of LLMs while minimizing external reliance.

Limitations

We discuss two major limitations in this paper regarding the dataset and method issues.

Dataset and Model. Due to space limitations and cost, we choose to conduct our experiments on two datasets and two models. We would like to evaluate the performance of more models, i.e., several open-

[8] The case study and more analysis can be found in Appendix.

source models, on the proposed datasets and more *compositional questions*.

Method. We mainly implement our method in zero-shot setting, and do not consider more complex implementation for each function within the framework, in order to demonstrate the great potential and effectiveness of our proposed method more clearly and straightforwardly. We left more complex implementations in our future works.

Furthermore, we would like to discuss the hallucination issues or other issues from the model side. Since different LLMs express certainty in various levels and may hallucinate the confidence score, we have meticulously designed the parameters α and β to ensure that our framework remains flexible and easily adaptable to a broader range of LLMs. While we acknowledge it may be relatively difficult to choose them, we are encouraged to see more and more recent studies align certainty expression across LLMs (Tao et al., 2024; Xu et al., 2024; Lee et al., 2024) and our method still outperforms other baselines even with the existing of these issues. From a dynamic and development standpoint, we believe our method and dataset could play a key role in the field of *compositional question* answering.

Ethical Statements

In this paper, there are only one issue about dataset collection.

Human Annotation The human inspection and annotation process are conducted by a respected data annotation company. All annotators receive fair compensation based on market rates and their personal information is not disclosed.

Acknowledgement

This work was partially supported by Hong Kong RGC GRF No. 14206324, CUHK direct grant No. 4055209, and CUHK Knowledge Transfer Project Fund No. KPF23GWP20.

References

Alfonso Amayuelas, Liangming Pan, Wenhu Chen, and William Wang. 2023. Knowledge of knowledge: Exploring known-unknowns uncertainty with large language models.

Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. 2023. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.

Matteo Gabburo, Nicolaas Paul Jedema, Siddhant Garg, Leonardo F. R. Ribeiro, and Alessandro Moschitti. 2024. Measuring retrieval complexity in question answering systems.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know.

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

Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2023. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive nlp.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. 2023. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation.

JoonHo Lee, Jae Oh Woo, Juree Seok, Parisa Hassanzadeh, Wooseok Jang, JuYoun Son, Sima Didari, Baruch Gutow, Heng Hao, Hankyu Moon, Wenjun Hu, Yeong-Dae Kwon, Taehee Lee, and Seungjai Min. 2024. Improving instruction following in language models through proxy-based uncertainty estimation.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-augmented generation for knowledge-intensive nlp tasks.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. Teaching models to express their uncertainty in words.

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

Sabrina J. Mielke, Arthur Szlam, Emily Dinan, and Y-Lan Boureau. 2022. Reducing conversational agents' overconfidence through linguistic calibration. *Transactions of the Association for Computational Linguistics*, 10:857–872.

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

Liangming Pan, Xiaobao Wu, Xinyuan Lu, Anh Tuan Luu, William Yang Wang, Min-Yen Kan, and Preslav Nakov. 2023. Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6981–7004, Toronto, Canada. Association for Computational Linguistics.

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

Ruiyang Ren, Yuhao Wang, Yingqi Qu, Wayne Xin Zhao, Jing Liu, Hao Tian, Hua Wu, Ji-Rong Wen, and Haifeng Wang. 2023. Investigating the factual knowledge boundary of large language models with retrieval augmentation.

Zhihong Shao, Yeyun Gong, Yelong Shen, Minlie Huang, Nan Duan, and Weizhu Chen. 2023. Enhancing retrieval-augmented large language models with iterative retrieval-generation synergy. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9248–9274, Singapore. Association for Computational Linguistics.

Shuchang Tao, Liuyi Yao, Hanxing Ding, Yuexiang Xie, Qi Cao, Fei Sun, Jinyang Gao, Huawei Shen, and Bolin Ding. 2024. When to trust llms: Aligning confidence with response quality.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. 2023. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10014–10037, Toronto, Canada. Association for Computational Linguistics.

Neeraj Varshney, Wenlin Yao, Hongming Zhang, Jianshu Chen, and Dong Yu. 2023. A stitch in time saves nine: Detecting and mitigating hallucinations of llms by validating low-confidence generation.

Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2023. Freshllms: Refreshing large language models with search engine augmentation.

Hongru Wang, Minda Hu, Yang Deng, Rui Wang, Fei Mi, Weichao Wang, Yasheng Wang, Wai-Chung Kwan, Irwin King, and Kam-Fai Wong. 2023a. Large language models as source planner for personalized knowledge-grounded dialogues. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9556–9569, Singapore. Association for Computational Linguistics.

Hongru Wang, Yujia Qin, Yankai Lin, Jeff Z. Pan, and Kam-Fai Wong. 2024a. Empowering large language models: Tool learning for real-world interaction. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 2983–2986, New York, NY, USA. Association for Computing Machinery.

Hongru Wang, Huimin Wang, Lingzhi Wang, Minda Hu, Rui Wang, Boyang Xue, Yongfeng Huang, and Kam-Fai Wong. 2025. Tpe: Towards better compositional reasoning over cognitive tools via multi-persona collaboration. In *Natural Language Processing and Chinese Computing*, pages 281–294, Singapore. Springer Nature Singapore.

Hongru Wang, Rui Wang, Fei Mi, Yang Deng, Zezhong Wang, Bin Liang, Ruifeng Xu, and Kam-Fai Wong. 2023b. Cue-CoT: Chain-of-thought prompting for responding to in-depth dialogue questions with LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12047–12064, Singapore. Association for Computational Linguistics.

Hongru Wang, Rui Wang, Boyang Xue, Heming Xia, Jingtao Cao, Zeming Liu, Jeff Z. Pan, and Kam-Fai Wong. 2024b. AppBench: Planning of multiple APIs from various APPs for complex user instruction. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 15322–15336, Miami, Florida, USA. Association for Computational Linguistics.

Hongru Wang, Boyang Xue, Baohang Zhou, Rui Wang, Fei Mi, Weichao Wang, Yasheng Wang, and Kam-Fai Wong. 2024c. UniRetriever: Multi-task candidates selection for various context-adaptive conversational retrieval. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 17074–17086, Torino, Italia. ELRA and ICCL.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023c. Self-consistency improves chain of thought reasoning in language models.

Yile Wang, Peng Li, Maosong Sun, and Yang Liu. 2023d. Self-knowledge guided retrieval augmentation for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10303–10315, Singapore. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models.

Yuxin Xiao, Paul Pu Liang, Umang Bhatt, Willie Neiswanger, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2022. Uncertainty quantification with pre-trained language models: A large-scale empirical analysis. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 7273–7284, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, Yifei Li, Jie Fu, Junxian He, and Bryan Hooi. 2023. Can llms express their uncertainty? an empirical evaluation of confidence elicitation in llms.

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.

Tianyang Xu, Shujin Wu, Shizhe Diao, Xiaoze Liu, Xingyao Wang, Yangyi Chen, and Jing Gao. 2024. Sayself: Teaching llms to express confidence with self-reflective rationales.

Boyang Xue, Fei Mi, Qi Zhu, Hongru Wang, Rui Wang, Sheng Wang, Erxin Yu, Xuming Hu, and Kam-Fai Wong. 2024a. Ualign: Leveraging uncertainty estimations for factuality alignment on large language models.

Boyang Xue, Hongru Wang, Rui Wang, Sheng Wang, Zezhong Wang, Yiming Du, Bin Liang, and Kam-Fai Wong. 2024b. A comprehensive study of multilingual confidence estimation on large language models.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Jiawen Wu, Xipeng Qiu, and Xuanjing Huang. 2023. Do large language models know what they don't know? In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 8653–8665, Toronto, Canada. Association for Computational Linguistics.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023a. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Wenhao Yu, Zhihan Zhang, Zhenwen Liang, Meng Jiang, and Ashish Sabharwal. 2023b. Improving language models via plug-and-play retrieval feedback.

Yuchen Zhuang, Yue Yu, Kuan Wang, Haotian Sun, and Chao Zhang. 2023. Toolqa: A dataset for llm question answering with external tools.

A Data Collection

A.1 CuQA Construction

We detail the prompts used for dataset construction in Tables 5-9.

Unknown: {unknown_event} According to the unknown event, please generate a question to which the answer is the entity {unknown_entity}.

Table 5: UnknownQuestionGen() function in Algorithm 1: generate an unknown_question about the unknown_event with the unknown_entity serving as the answer.

Generate a detailed passage about {entity}

Table 6: KnownEventsGen() function in Algorithm 1: generate a supporting background information known_events about the unknown_entity based on the internal known knowledge of LLMs.

Known: {known_event} According to known events, please generate a question to which the answer is be the entity {entity}.

Table 7: KnownQuestionGen() function in Algorithm 1: generate a known_question based on the known_event with the entity serving as the answer.

Seen: {known_passage} Generate a question that meets the following conditions: 1. contains the terms {unknown_entity} in question, 2. the answer is {known_entity}.

Table 8: KnownQuestionGen2() function in Algorithm 1: given the known_event, generate a known_question which contains the unknown_entity in the question and can be answered with known_entity sampled from the known_event.

Question One: {unknown_question} Question Two: {known_question} Generate a more natural combined question of question one and question two.

Table 9: MergeQuestions() function in Algorithm 1: merge the generated unknown_question and known_question into a single multi-hop question.

A.2 Data Examples

We list two easy examples from CuQA dataset in Table 10. There are two reasoning types in CuQA: (1) AAB represents the two questions Q1 and Q2 are independently created before being merged; (2) ABC means the generation of Q2 depends on Q1, where in the listed example, A1 is embedded within Q2. It means the three QA pairs are synthesized in a concatenated form. We also regard two merged QA pairs as the sub-problems, combining them to form a more complex question that demands enhanced reasoning and more decomposition.

B Experiment on FreshQA Dataset

B.1 Data Statistics

# hops		effective year				total
multi-hop	one-hop	before-2022	2022	2023	2024	-
137	463	279	131	143	47	600

Table 11: Data statistics of FreshQA.

We report the data statistics of the FreshQA dataset in Table 11. Different fromthe CuQA dataset that involves multi-hop reasoning for all instances, FreshQA is constructed to benchmark large language models' ability in addressing questions with time-changing knowledge. More than 77% of questions are single-hop that requires no additional problem decomposition. The questions are split into four categories according to the effective year of the answers: before-2022 (46.50%), 2022 (21.83%), 2023 (23.83%), 2024 (7.83%).

B.2 Analysis

We present the performance details of 1106 on FreshQA by the time-frames of questions in Figure 9.

The performance of *Self-DC* increases as the effective year of questions become earlier. In general, the best performance is achieved on questions before 2022 and a decreasing trend is observed for more recent questions with both *verb* and *prob* confidence acquisition methods using 1106. We also identified the same finding when using 4o-mini with *verb* method. This is not surprising as its training data ends up to September 2021[9].

[9] https://platform.openai.com/docs/models/gpt-3-5-turbo.

Reasoning Type	Examples
AAB	**Q1:** Which countries signed a trilateral pact on 18 August, 2023? **A1:** The United States, Japan, and South Korea **Q2:** What's the G7 member countries? **A2:** Canada, France, Germany, Italy, Japan, the United Kingdom and the United States. **Merged-Q:** Which two G7 member countries signed a trilateral pact on 18 August, 2023? **Merged-A:** The United States, Japan.
ABC	**Q1:** Where did the first AI Safety Summit take place? **A1:** United Kingdom **Q2:** Is United Kingdom an African country? **A2:** No. **Merged-Q:** Did the first AI Safety Summit take place in an African country? **Merged-A:** No

Table 10: Data examples from CuQA dataset. For each example, one question is generated based on an unseen event and the other is generated based on model generated passage described in Section 3. The two questions and corresponding answers are then merged and post-processed to get the final question and answer.

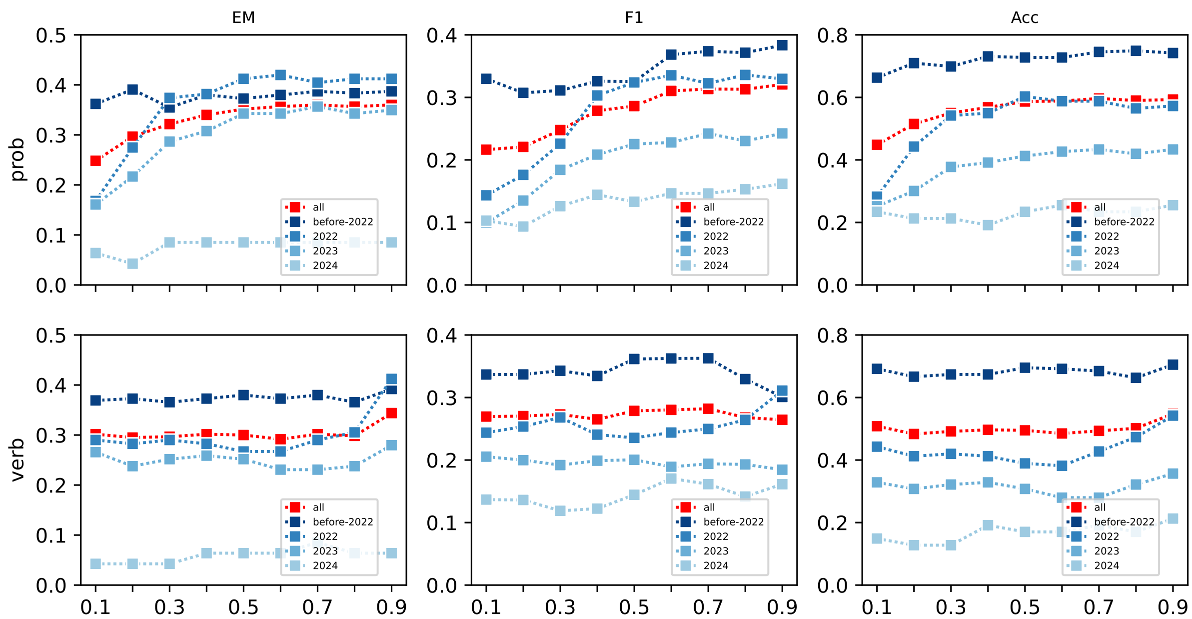

Figure 9: The performance of `1106` on FreshQA questions in different time-frames with varying α values. We fix β as 0.1 for the analysis. The first and second rows correspond to the performance with *probability*- and *verbalized*-based confidence scoring respectively.

C More Analysis

C.1 Different Number of Retrieved Results

We then set the number of retrieved results ranging from 1 to 4 to investigate the effects. Figure 10 shows the results. It is found that setting the number of retrieved results as 3 leads to the best performance for both of these two datasets, and the performance on FreshQA is more sensitive to the number of retrieved documents compared with CuQA.

C.2 More Models.

We additionally run experiments on the Qwen2.5-7b-Instruct model by following the setting at main experiments. Table 12 shows the final results. It is observed that our method still achieves better trade-off between effectiveness and efficiency.

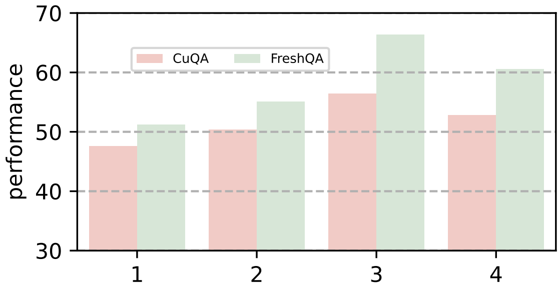

Figure 10: The performance of different number of retrieved results using *prob* methods on `4o-mini`.

Methods	#R	EM	F1	Acc
RR	n	22.2	27.8	38.2
ReFeed	$2n$	24.8	17.2	33.8
IRCoT	$3n$	37.6	4.5	43.6
`Self-DC` (Verb)	0-n	23.6	26.9	35.8
`Self-DC` (Prob)	0-n	23.8	28.3	40.0

Table 12: Performance results on Qwen2.5-7b-Instruct model.

D Demonstrations

We mainly follow Shao et al. (2023) for prompt design. We list the used prompts and demonstrations for baselines in Tables 13-17 and the prompts for *`Self-DC`* in Table 20.

Please answer the following question with just a few words.

Question: {`question`}
The answer is

Table 13: Prompt for Direct Prompting baseline.

Please answer the following question with just a few words. Let's think step by step.

Question: {`question`}
The answer is

Table 14: Prompt for zero-shot Chain-of-thought baseline.

Refer to the passage below and answer the following question with just a few words.

Passage: {`passage`}
Question: {`question`}
The answer is

Table 15: Prompt for Retrieve-then-read baseline. The `passage` comes from retrieval results.

Generate a background document from Wikipedia to answer the given question.

{`question`}

Refer to the passage below and answer the following question with just a few words.

Passage: {`passage`}
Question: {`question`}
The answer is

Table 16: Prompts for Generate-then-read baseline.

Quesion: Which country that has joined in 2023 Rugby World Cup in the final also held the 2023 FIFA Women's World Cup?
Are follow up questions needed here: Yes
Follow up: Which countries held the 2023 FIFA Women's World Cup?
Intermediate answer: Australia and New Zealand.
Follow up: Which countries have joined in 2023 Rugby World Cup in the final?
Intermediate answer: New Zealand, South Africa
So the final answer is: New Zealand

Question: Is the country that held the 49th G7 summit same as where the 42nd G7 summit took place?
Are follow up questions needed here: Yes
Follow up: Where did the 42nd G7 summit take place?
Intermediate answer: Kruen, Bavaria, Germany
Follow up: Where did the 49th G7 summit take place?
Intermediate answer: Hiroshima, Japan
So the final answer is: No

Question: What's the rate share of the 2023 elected president in the most developed South-East Asian countries?
Are follow up questions needed here: Yes
Follow up: Which country is the most developed in South-East Asia?
Intermediate answer: Singapore
Follow up: What's the rate share of the 2023 elected president in Singapore?
Intermediate answer: over 70 percent
So the final answer is: over 70 percent

Table 17: Demonstrations for Self-Ask baseline on CuQA.

Quesion: Which country that has joined in 2023 Rugby World Cup in the final also held the 2023 FIFA Women's World Cup?
Let's think step by step
New Zealand and South Africa have joined in 2023 Rugby World Cup in the final, and Australia and New Zealand held the 2023 FIFA Women's World Cup.
So the answer is New Zealand

Question: Is the country that held the 49th G7 summit same as where the 42nd G7 summit took place?
Let's think step by step
The 42nd G7 summit took place in Kruen, Bavaria, Germany and The 42nd G7 summit took place in Japan So the answer is No

Question: What's the rate share of the 2023 elected president in the most developed South-East Asian countries?
Let's think step by step
Singapore is the most developed and wealthy South-East Asia country, and the the rate share of the 2023 elected president in Singapore is over 70 percent.
So the answer is over 70 percent

Table 18: Demonstrations for few-shot Chain-of-thought baseline on CuQA.

Passage: September 8 – October 28 – The 2023 Rugby World Cup is held in France, and New Zealand (the All Blacks) lost 11–12 to South Africa in the final at the Stade de France. 20 July – August 20 – The 2023 FIFA Women's World Cup is held in Australia and New Zealand. In the final, Spain wins 1–0 against England.
Quesion: Which country that has joined in 2023 Rugby World Cup in the final also held the 2023 FIFA Women's World Cup?
Let's think step by step
New Zealand and South Africa have joined in 2023 Rugby World Cup in the final, and Australia and New Zealand held the 2023 FIFA Women's World Cup. So the answer is New Zealand

Passage: The 42nd G7 summit took place in Kruen, Bavaria, Germany. The 49th G7 summit takes place in Hiroshima, Japan. Ukrainian president Volodymyr Zelenskyy arrives in Japan on the second day of the summit.
Question: Is the country that held the 49th G7 summit same as where the 42nd G7 summit took place?
Let's think step by step
The 42nd G7 summit took place in Kruen, Bavaria, Germany and The 42nd G7 summit took place in Japan So the answer is No

Passage: 1 September – 2023 Singaporean presidential election: Economist and former deputy prime minister Tharman Shanmugaratnam is elected president with a vote share of over 70 percent.
Question: What's the rate share of the 2023 elected president in the most developed South-East Asian countries?
Let's think step by step
Singapore is the most developed and wealthy South-East Asia country, and the the rate share of the 2023 elected president in Singapore is over 70 percent.
So the answer is over 70 percent

Table 19: Demonstrations for ITER-RETGEN baseline on CuQA.

Please read the question, give the answer and indicate your level of confidence. Use the following format to provide your answer and confidence level:

Answer: [Your answer]
Confidence (0-100): [Your confidence level, please only include the numerical number, e.g. 80]%

Note: The confidence level indicates the degree of certainty you have about your answer and is represented as a percentage. For instance, if your confidence level is 80%, it means you are 80% certain that your answer is correct and there is a 20% chance that it may be incorrect. If you do not know the answer, simply output confidence as 0%.

Question: `{question}` Please answer this question and provide your confidence level. Note that the confidence level indicates the degree of certainty you have about your answer and is represented as a percentage.
Answer:

Please read the question, divide the question into smaller, independent parts. By solving these individual sub-questions and combining their answers, you can derive the solution to the main question. Use the following format to provide your answer: #1: [sub-question 1], #2: [sub-question 2], ...

Question: `{question}`
Answer:

Refer to the passage below and answer the following question with just a few words. Passage: `{passage}`

Question: `{question}`
The answer is

Generate a background document from Wikipedia to answer the given question. `{question}`

Refer to the passage below and answer the following question with just a few words. Passage: `{passage}`

Question: `{question}`
The answer is

Question: `{question}`

Here are all related sub-questions and corresponding answers: `{sub_qas}`

According to answers of all related sub-quesions of the original question, please generate the final answer of the original question using a few words.

Table 20: Prompts for ***Self-DC***: verbalize-based confidence acquisition, *decompose*, *retrieve-then-read*, *generate-then-read*, and *combine-sub-qas*.