Self-DC: When to retrieve and When to generate Self Divide-and-Conquer for Compositional Unknown Questions

Anonymous ACL submission

Abstract

Retrieve-then-read and *generate-then-read* are two typical solutions to handle unknown and known questions in open-domain question-answering, while the former retrieves necessary external knowledge and the later prompt the large language models to generate internal known knowledge encoded in the parameters. However, few of previous works consider the *compositional unknown questions*, which consist of several known or unknown sub-questions. Thus, simple binary classification (known or unknown) becomes sub-optimal and inefficient since it will call external retrieval excessively for each compositional unknown question. To this end, we propose the first **C**ompositional **u**nknown **Q**uestion-**A**nswering dataset (CuQA), and introduce a **Self D**ivide-and-**C**onquer (***Self-DC***) framework to empower LLMs to adaptively call different methods on-demand, resulting in better performance and efficiency. Experimental results on two datasets (CuQA and FreshQA) demonstrate that Self-DC can achieve comparable or even better performance with much more less retrieval times compared with several strong baselines [1].

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), such as

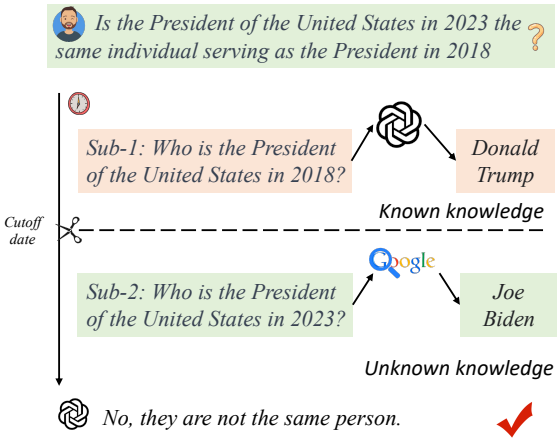

Figure 1: A example of compositional unknown questions, in which a unknown question consists of some sub-questions can be answered using *known* knowledge while other sub-questions necessitate *unknown* knowledge according to the cutoff date of LLMs.

Google Search. The questions arise: *When to utilize external retrieval and when to prompt LLMs to generate already known knowledge in order to answer current question?*

To address this question, previous studies have taken different approaches. Some indiscriminately employ external retrieval methods (Trivedi et al., 2023; Shao et al., 2023), while others first assess the certainty or uncertainty of large language models (LLMs) regarding the question (Lin et al., 2022; Chen and Mueller, 2023). Subsequently, they call either *generate-then-read* or *retrieve-then-read* according to the assessment (Wang et al., 2023d). Nevertheless, both of these methods have their own limitations. On the one hand, simply calling retrieval for every question does not guarantee better performance while making the extensive known knowledge meaningless (Wang et al., 2023d), and also brings serious efficiency issue especially when there are iterative retrieval (Trivedi et al., 2023). On the other hand, dividing the questions into binary oppositions of known and unknown oversimplifies

[1] The dataset and code will be publicly released soon.

their complexity, being infeasible or sub-optimal in certain cases. For example, as shown in Figure 1, a compositional unknown question consists of both known and unknown sub-questions. Therefore, it will degrade into simply calling retrieval for every question automatically, as the compositional unknown remains an unknown question for LLMs.

In this paper, we target the compositional questions (Yang et al., 2018; Press et al., 2023) from the perspective of known/unknown, which is more practical and challenging. Specifically, we first introduce another dimension to view questions: *simple to composition*, according to the question whether or not can be further divided. If not, it is a simple known or unknown question, then it can be answered by *generate-then-read* (Yu et al., 2023a) or *retrieve-then-read* (Lewis et al., 2021). However, when the question becomes compositional, especially compositional unknown, the problem becomes tricky as it can be divided into several cases, which contains all other types of questions [2], such as the combination of single (or, compositional) known and single unknown. To this end, we build the first **C**ompositional **u**nknown **Q**uestion-**A**nswering dataset (CuQA), accompany with a automatic data collection pipeline with minimal human efforts. Then we propose a **S**elf **D**ivide-and-**C**onquer (***Self-DC***) method based on above observations. Seeing the potential of LLMs in expressing their certainty/uncertainty (Lin et al., 2022; Xiao et al., 2022), we divide the question into different types of sub-questions and call different functions iteratively according to the confidence score outputed by LLMs themselves. Overall, our contributions can be summarized as follows:

- To 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 corresponding dataset according to recent events from Wikipedia, named CuQA dataset, serving as an important evaluation benchmark for LLMs in known/unknown.
- We present a flexible and robust ***Self-DC*** framework, which is capable to adaptively call internal known knowledge and external unknown knowledge on-demand.
- Experimental results on CuQA and FreshQA datasets show the superiority of Self-DC in terms of both effectiveness and efficiency, revealing its promising potential to solve compositional reasoning problem.

[2] Basically, there are four types of questions: simple known, simple unknown, compositional known and compositional unknown.

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 contain 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 assess 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 incorporating of retrieval augmentation (Ren et al., 2023). Distinguished from previous works, our work target compositional unknown questions, consisting of both unknown and known sub-questions.

Certainty and Uncertainty of LLMs. To calibrate the known and unknown of LLMs, there are lots of studies which have delved into methods for estimating and quantifying *certainty and uncertainty* in LLM 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) benchmark three categories of 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)). In this work, we build a bridge to connect known/unknown with certainty/uncertainty based on one simple assumption: LLMs are not certain about the unknown information and certain about known.

Complex Reasoning. Prompting-based large language models (LLMs) are surprisingly powerful to answer multi-hop questions (Yang et al., 2018) with CoT (Wei et al., 2023; Wang et al., 2023a). However, they still struggle in compositional reasoning, particularly when the necessary unknown knowledge is required. To address this dilemma, there has been a growing interest in exploring different reasoning techniques to enhance the reasoning and planning capabilities of LLMs, including 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). However, few of them focus on the compositional unknown questions in practice.

3 Data Collection

In this section, we detailedly 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 at the final step.

3.1 Automatic Collection

Algorithm 1 shows the pseudo-code details. Specifically, we assume there is a cutoff date for each LLM, and all the pretraining corpus is collected before the cutoff date, for example, the cutoff date of `gpt4-turbo` is April 2023[3]. In this way, we firstly collect all events happened after cutoff date from Wikipedia[4], named *unknown events*. Then we carefully design several prompt templates to utilize LLMs generating required questions according to these unknown events and internal known knowledge (a.k.a, *known events*), inspired by Wang et al. (2023b). For example, we can use one entity in the unknown events as answer, and prompt the LLMs

Algorithm 1 CuQA Generation Algorithm

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

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

to generate corresponding question according to the events (line 6). More details can be found in Appendix A.1. We store the question, answer and all intermediate results for further processing. To minimize the human efforts, we additionally introduce some automatic quality control procedures, including a python script to validate whether or not the format of outputs meets the instructions [5].

3.2 Human-in-the-Loop and Statistics

Beside automatic quality control, we also introduce human assessment to ensure the quality of CuQA dataset. In detail, we employ three well-educated annotators to: 1) filter unqualified samples, such as answer is not correct or question is not unknown; and 2) rewrite the generated question to be more natural. We finally collect 153 easy questions and 100 hard questions, while the later is further merged using compositional question-answering pairs (line 12), resulting in 253 questions totally in the dataset. The examples in the data can be found in Appendix A.2.

4 Method

In this section, we introduce the details of our proposed method ***Self-DC***. We start from the whole framework to provide a global understanding first

[3] https://openai.com/blog/new-models-and-developer-products-announced-at-devday
[4] https://en.wikipedia.org/wiki/2023
[5] It is worthy noting that our data collection can be time-evolving given the cutoff date.

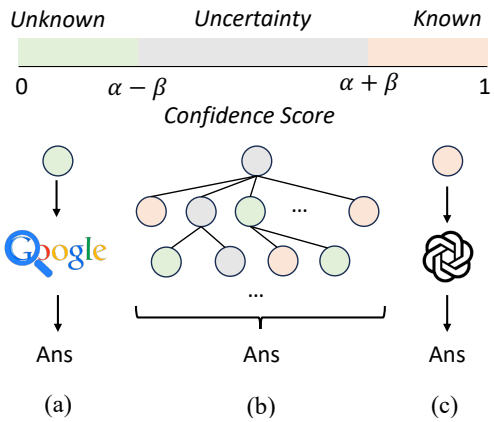

Figure 2: A high-level depiction of our proposed method: ***Self-DC***: a) *retrieve-then-read* for unknown questions, b) *decompose-and-combination* for uncertain questions; and c) *generate-then-read* for known questions.

and then delve into four specific functions within this framework.

4.1 Framework: Self Divide-and-Conquer

Figure 2 shows the high-level depiction. Generally, given a question, we firstly get the confidence score of LLMs to answer the question, then we can choose which way to handle the current question accordingly. Specifically, we divide the confidence score into three ranges $[0, \alpha - \beta], (\alpha - \beta, \alpha + \beta), [\alpha + \beta, 1]$ in which the α and β are hyper-parameters. Thus, when the confidence score falls into the left ($[0, \alpha - \beta]$) or right ($[\alpha + \beta, 1]$) side, we can call *retrieve-then-read* or *generate-then-read* to answer respectively. However, when it encounters the uncertain or confusing questions, we then decompose the question into several sub-questions, aiming to decrease the uncertainty. We set several termination conditions to guarantee the iteration do not overflow: 1) *the number of sub-questions is 1*, which means it probably is a *single* question; and 2) *the number of iteration depth is less than one predefined* t. Once these situations happen, we simply regard current sub-question as unknown question and then call *retrieve-then-read*. We then combine answers of all sub-questions to prompt the LLMs to get the final answer for the original question. In this way, we only call compositional reasoning when necessary instead of treat all questions indiscriminately. Figure 3 shows the simplified implementation details of the main function.

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 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:

$$conf = \frac{1}{N} \sum_{i=1}^{N} \hat{p}_i \quad (1)$$

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 level of confidence score, we carefully design several functions to complete the compositional reasoning task, aiming to provide 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 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 to answer the question by referring the *retrieved* passage.

- ***decompose*:** We prompt the LLMs to systematically break down the overarching question into several smaller sub-questions. The answers

```python
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*.

to these sub-questions collectively contribute to deriving the answer to the original overarching question, similar with Press et al. (2023) and Xu et al. (2023).

- ***combine-sub-qas*:** After the decomposition, we call the main function to enter the next iteration as shown in Figure 3, aiming to get the answer of each sub-question. Subsequently, we can combine the answers to all sub-questions to get the answer to the original question, which completes the iteration.

Combining these function together, ***Self-DC*** can be easily and quickly implemented and extended to different tasks or models.

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[6]. Following previous works (Yu et al., 2023a,b; Trivedi et al., 2023), we select Exact Match (EM), F1 to evaluate the performance of different methods. Furthermore, we also use Acc† as an additional metric to prompt LLMs to assess the predictions in relation to the actual ground-truth answers, following Shao et al. (2023) [7], aiming to enhance the robustness of the evaluation.

5.2 Implementation Details

We mainly conduct our experiments on two different backbone models: `gpt-3.5-turbo-1106` and `gpt-3.5-turbo-instruct`[8], hereinafter referred to as `1106` and `instruct` respectively, following lots of previous works (Yu et al., 2023a,b; Shao et al., 2023). For the Acc† evaluation, we always use `1106` 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 google search engine following `LangChain` [9] as a external retriever, and we set the number of retrieved results as 3 and the max iteration depth t as 3. We keep all demonstrations as same and the details can be found in the Appendix.

[6]We use the FreshQA_v01122024, and the detailed analysis can be found in Appendix due to space limit.

[7]We use the exactly same prompt template here following original paper.

[8]We choose this since `text-davinci-003` is shut down and replace using `gpt-3.5-turbo-instruct` by OpenAI.

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

Methods	EM	F1	Acc†
w/o retrieval			
Direct	15.4	12.4	30.8
CoT	13.8	12.2	28.9
Few-shot-CoT*	10.3	2.3	16.2
GenRead	11.1	8.8	25.7
w/ retrieval			
RR	15.0	14.0	31.2
REFEED	13.4	4.2	30.8
IRCoT	**19.0**	7.6	34.8
Self-Ask*	7.9	2.9	11.1
ITER-RETGEN*	9.9	2.5	14.2
Self-DC (*verb*)	17.8	**15.9**	35.6
Self-DC (*prob*)	17.4	15.2	**36.4**

Table 1: The performance of baselines and *Self-DC* with the `1106`. The baseline* means it uses demonstrations. We **bold** the highest performance and underline the second-highest performance.

Methods	EM	F1	Acc†
w/o retrieval			
Direct	13.0	5.8	25.3
CoT	15.0	6.0	24.9
Few-shot-CoT*	19.4	4.4	31.2
GenRead	16.6	9.6	26.9
w/ retrieval			
RR	11.9	7.3	25.7
REFEED	6.7	4.5	26.5
IRCoT	17.4	**9.7**	30.8
Self-Ask*	7.1	7.4	24.9
ITER-RETGEN*	9.9	6.7	28.5
Self-DC (*verb*)	**21.1**	9.4	**35.1**

Table 2: The performance of baselines and *Self-DC* with the `instruct`. The baseline* means it uses demonstrations.

5.3 Main Results

Table 1 and Table 2 show the performances of all baselines and our proposed ***Self-DC*** on the `1106` and `instruct` respectively. We can draw the following conclusions 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 worse than Direct. In a ideal situation, the performance of all baselines *w/o retrieval* for unknown questions should be zero with no exception. However, we found it is highly related to the types of questions and the hallucination of LLMs (Ji et al., 2023). For a example, the answer of LLMs to a typical of *Yes/No* question in the dataset, *Did the first AI Safety Summit take place in an African country?*, is No. These LLMs should not aware this information during their training but their answers are correct. Thus, we can found the performance of *w/o retrieval* of all baselines is not zero perfectly. Moreover, we can found that Few-shot-CoT can not bring consistent improvement too, as shown in Table 1, revealing the poor reasoning and in-context learning capability of LLMs when it comes to unknown questions. The similar phenomenon is also found by Wang et al. (2023d).

Self-DC* VS. *generate-then-read / retrieve-then-read*.** We emphasise that the GenRead and RR methods can be regarded as two special cases of ***Self-DC with different choices of α and β [10]. It is obvious that ***Self-DC*** outperforms both them in two different backbone models, revealing its effectiveness and robustness to divide compositional questions and solve sub-question using different methods dynamically. In addition, RR even performs worse than GenRead at `instruct`, we suspect that the model's difficulty in distinguishing useful information from noisy retrieved results may cause this issue.

Self-DC* V.S Iterative Retrieval.** When comparing ***Self-DC to other iterative retrieval methods like REFEED, IR-CoT, Self-Ask, and ITER-RETGEN, it is evident that ***Self-DC*** achieves the highest performance. Besides that, we note that there are several advantages which may be more important in certain cases: 1) ***Self-DC*** *does not require any demonstrations*. Some of these methods necessitate demonstrations to make the output of LLMs meets the requirements. For example, Self-Ask requries the LLMs to answer Yes or No to decide whether or not to continue generating follow up question. We found it is hard to achieve when the demonstrations is less or the questions is out of the knowledge of LLMs (e.g., unknown questions). 2) ***Self-DC*** *require much less times of external retrieval.* Suppose there are n questions in the test set, the times of retrieval of these baseline methods mostly is greater than $2n$ due to the intrinsic design, while ***Self-DC*** only requires less than n times [11]. More analysis can be found in next section.

[10] GenRead: $\alpha = 0, \beta = 0$; RR: $\alpha = 1, \beta = 0$

[11] Specifically, Self-DC requires around $0.6n$ to $0.8n$ with

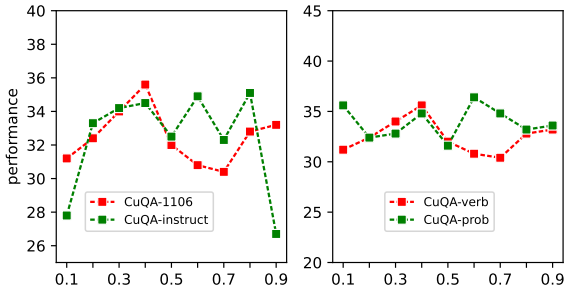

Figure 4: The performance of different choices of α with $\beta = 0.1$. **Left:** The performance of different models with confidence type is *verb*; and **Right:** The performance of different confidence types (*verb* or *prob*) with the same model `1106`.

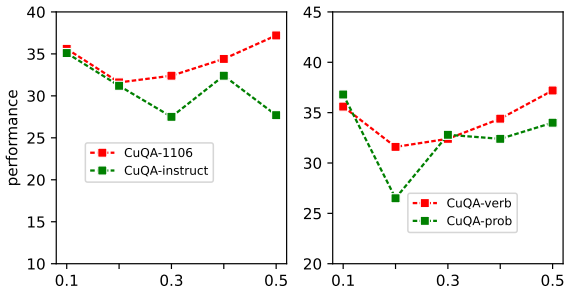

Figure 5: The performance of different choices of β with a fixed α. **Left:** The performance of different models with confidence type is *verb*; and **Right:** The performance of different confidence types (*verb* or *prob*) with the same model `1106`.

6 Analysis

6.1 The Impacts of Different α and β

It is impossible to consider all combinations of α and β due to computation cost, thus we design a two-stage selection to determine the final choice of α and β according to the performance of Acc†. Firstly, we fix $\beta = 0.1$ and set α to $[0.1, 0.2, 0.3, ..., 0.9]$ to select the best α. The results can be found in Figure 4. 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 best performances are achieved in the middle choice of α, revealing the complexity of the target problem. In detail, when using *verb* to acquire confidence score, as shown in the left of Figure 4, it demonstrates an upward and then downward trend globally. It is reasonable since LLMs utilize more *generate-then-read function* at the begging (e.g., $\alpha = 0.1, \beta = 0.1$), 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 noises compared with gains. To conclude, we found that α=0.4 (or, 0.6) works best with *verb* (*or, prob*) methods when using `1106`, α=0.8 works best with *verb* method when using `instruct`.

Then 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 5 shows the final results. Similarly, there is no monotonically increasing or decreasing trends. Despite we can find better performance with other choices of β, simply initializing β as 0.1 is a smart strategy to achieves near-optimal performance. In general, the choices of α and β are extremely tricky in practice with lots of external factors, including datasets, model, different methods to collect confidence scores, and also time cost.

6.2 The Impacts of Different Iteration Depth t

We choose the settings of best performance to conduct the rest analysis to save the cost [12]. First of all, we can find that different choices of t have a serious effects on the final performance. Specifically, when the iteration depth increases, the number of retrieval times will increase accordingly as observed in `instruct`. However, if we choose another model, i.e., `1106`, the number of retrieval times will not be affected too much [13]. We suspect this is due to the better instruction-following capability of `instruct`. Even when encountering unknown questions, the model could continuously break down the question as long as it stays within the predefined iteration depth t, resulting in more retrieval times. Most importantly, we want to emphasise here that the number of retrieval times usually will not exceed the number of original test set n, and sometimes it only need to call less than $0.5n$ times, revealing the great advantages of ***Self-DC*** over other iterative retrieval-augmented baselines.

6.3 Different Number of Retrieved Results

We then set the number of retrieved results ranging from 1 to 5 to investigate the effects. Figure 6

the reported performance.

[12] `1106`: α=0.4, β=0.1, *verb*
`instruct`: α=0.8, β=0.1, *verb*.

[13] We also find similar situation when using *prob* method for `1106`.

Model	2	3	4
`1106` (*verb*)	32.8 (163)	35.6 (163)	32.4 (163)
`instruct` (*verb*)	28.5 (74)	35.1 (82)	27.7 (91)

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

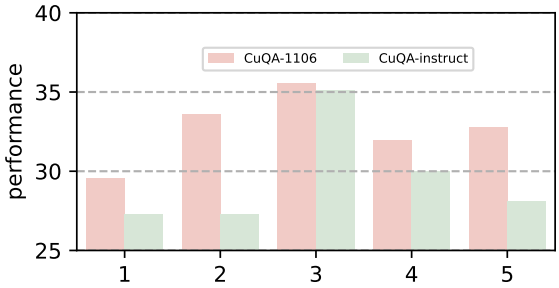

Figure 6: The performance of different number of retrieved results using same *verb* method under differen models.

shows the results. Despite setting the number of retrieved results as 3 leads to the best performance for both of these two models, there is a nuanced difference in the overall trend. For example, when the number is increased from 1 to 3, we can found it always leads to better performance with `1106`. However, the performance of `instruct` does not change when the number is less than 3. In addition, when the number exceeds 3, its performance decreases dramatically while the `1106` still achieves comparable performance compared with the number is 2. We attribute this to the more powerful capability of `1106` to capture the related information and filter noises.

6.4 Error Analysis

Despite the effectiveness and robustness of the proposed ***Self-DC*** framework for compositional unknown questions, we would like to better understand the bottleneck and failures of ***Self-DC*** for further improvement.

Performance of different types of questions. Table 4 shows the results of different types of questions in CuQA. We can found there is a large gap between easy question and hard question, revealing the higher requirements for the models towards complicated compositional unknown questions.

Accuracy of confidence scores. Furthermore, we conduct a deep analysis of confidence scores the behaviors of ***Self-DC***. On the one hand, when using *verb* method, we find that the confidence scores are

Types	EM	F1	Acc†
easy	24.2	19.4	38.6
hard	1.0	8.0	23.0

Table 4: The performance of ***Self-DC*** on two types of question: *easy* and *hard* using `1106`.

0 for more than 65% cases, and over 0.9 for around 20% cases with `1106`. However, the trend reverses when it comes to `instruct` which gives 0.9 more frequently ($\approx$ 45%). 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. 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%). We conclude that `1106` performs better at the uncertainty estimation compared with `instruct`, and *prob* method outperforms *verb*, achieving more accurate confidence scores.

Analysis of decomposition The times of decomposition is highly affected by the confidence scores. Sometimes, the best performance can be achieved without any decomposition with well-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 [14].

7 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 less retrieval times, but also shows a promising potential in different applications with carefully selected hyper-parameters.

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

Limitations

To best of our knowledge, there are two major limitations in this paper.

Dataset and Model. Due to space limit and cost, we choose to conduct our experiments on two datasets and two models. We would like to evaluate the performance of more model, i.e., several open-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 clear and straightforward. We left more complex implementations in our future works.

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.

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.

Jiuhai Chen and Jonas Mueller. 2023. Quantifying uncertainty in answers from any language model and enhancing their trustworthiness.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

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.

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.

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.

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, Rui Wang, Fei Mi, Yang Deng, Zezhong Wang, Bin Liang, Ruifeng Xu, and Kam-Fai Wong. 2023a. 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.

Jinyuan Wang, Junlong Li, and Hai Zhao. 2023b. Self-prompted chain-of-thought on large language models for open-domain multi-hop reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2717–2731, Singapore. Association for Computational Linguistics.

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.

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 FreshQA dataset in Table 11. Different from 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% question 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 Main Results

We report the performance of *Self-DC* in Table 12 and 13 for 1106 and instruct respectively. They key findings are similar to that on CuQA dataset.

B.3 Analysis

We present the performance details of 1106 on FreshQA by the time-frames of questions in Figure 7.

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 instruct

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.

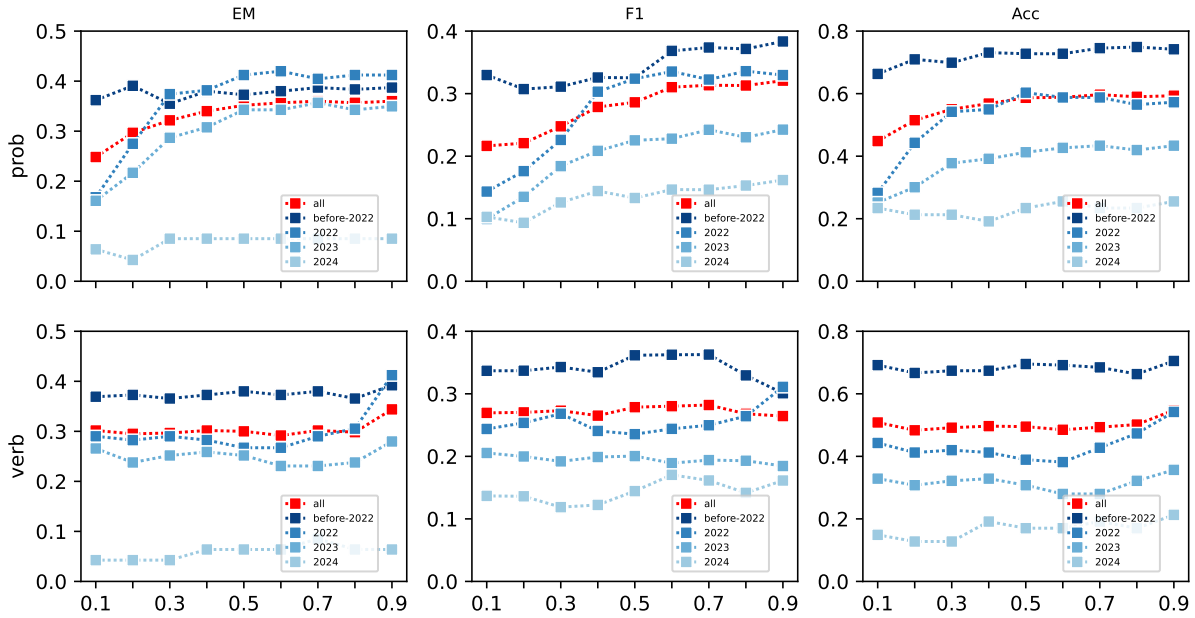

Figure 7: The performance of `1106` on FreshQA questions in different time-frames with varying α values. We fix β as 0.1 for the analysis. The first and second rows correspond to the performance with *probability*- and *verbalized*-based confidence scoring respectively.

with *verb* method. This is not surprising as its training data ends up to September 2021[15].

Retrieval augmentation introduces more benefits for more recent questions that beyond the knowledge boundary of large language models. As α ranging from 0.1 to 0.9, the decision range for *retrieve-then-read* in Figure 2(a), i.e., from 0 to $(\alpha - \beta)$, increases. The performance increment is more significant for questions in 2022, 2023, 2024 than questions before 2022. The performance of `1106` on questions before 2022 are relatively stable with the varying of α. Large language model may store efficient knowledge for some questions inclusive in its training data, where retrieval is not a must for these instances. This underscores the superiority of ***Self-DC*** to involve retrieval augmen-

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

Methods	FreshQA		
	EM	F1	Acc†
w/o retrieval			
Direct	21.3	21.5	36.5
CoT	20.8	21.6	36.0
Few-shot-CoT*	18.7	11.9	31.6
GenRead	18.2	17.1	36.3
w/ retrieval			
RR	36.0	**32.4**	61.8
REFEED	23.2	14.3	39.0
IRCoT	**37.5**	16.5	60.3
Self-Ask*	11.5	14.8	29.8
ITER-RETGEN*	25.7	13.3	37.5
Self-DC (*verb*)	35.2	27.1	56.0
Self-DC (*prob*)	36.8	31.4	**62.0**

Table 12: The performance of baselines and SelfDC with the `1106`. The baseline* means it uses demonstrations. We **bold** the highest performance and underline the second-highest performance.

Methods	FreshQA		
	EM	F1	Acc†
w/o retrieval			
Direct	19.0	8.2	38.3
CoT	17.8	7.5	38.3
Few-shot-CoT*	23.3	13.4	42.0
GenRead	21.5	7.8	36.3
w/ retrieval			
RR	**33.0**	12.1	53.8
REFEED	19.2	10.1	35.7
IRCoT	32.8	**21.2**	**56.5**
Self-Ask*	13.8	16.2	36.7
ITER-RETGEN*	14.5	16.0	43.8
Self-DC (*verb*)	25.2	19.5	52.1

Table 13: The performance of baselines and SelfDC with the `instruct`. The baseline* means it uses demonstrations. We **bold** the highest performance and underline the second-highest performance.

tation selectively as needed, optimizing a balance between the internal parametric memory of LLMs and the external retrieval knowledge.

C Case Study

Figure 8 and Figure 9 shows an example from CuQA and FreshQA dataset respectively. We can found that ***Self-DC*** is capable to address different compositional questions including known and unknown, which is highly dependent on the chosen models.

D Demonstrations

We mainly follow Shao et al. (2023) for prompt design. We list the used prompts and demonstrations for baselines in Tables 14-18 and the prompts for ***Self-DC*** in Table 21.

Please answer the following question with just a few words.

Question: `{question}`
The answer is

Table 14: 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 15: 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 16: 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 17: Prompts for Generate-then-read baseline.

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 8: 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 9: An example from FreshQA dataset where one compositional question can be further divided into known and unknown questions.

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 18: 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 19: 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 20: 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 21: Prompts for ***Self-DC***: verbalize-based confidence acquisition, *decompose*, *retrieve-then-read*, *generate-then-read*, and *combine-sub-qas*.