Received: 26 August 2023 | Revised: 4 December 2023 | Accepted: 20 December 2023

DOI: 10.4218/etrij.2023-0355

FEATURED ARTICLE

ETRI Journal WILEY

Towards a small language model powered chain-of-reasoning for open-domain question answering

Jihyeon Roh | **Minho Kim** | **Kyoungman Bae**

Language Intelligence Research Group, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea

Correspondence
Jihyeon Roh and Minho Kim, Language Intelligence Research Group, Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea.
Email: jihyeon.roh@etri.re.kr and kimmh@etri.re.kr

Funding information
This research was supported by the Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government (MSIT) (no. 2022-0-00369, [Part 4] Development of AI Technology to support Expert Decision-making that can Explain the Reasons/Grounds for Judgment Results based on Expert Knowledge).

Abstract

We focus on open-domain question-answering tasks that involve a chain-of-reasoning, which are primarily implemented using large language models. With an emphasis on cost-effectiveness, we designed *EffiChainQA*, an architecture centered on the use of small language models. We employed a retrieval-based language model to address the limitations of large language models, such as the hallucination issue and the lack of updated knowledge. To enhance reasoning capabilities, we introduced a question decomposer that leverages a generative language model and serves as a key component in the chain-of-reasoning process. To generate training data for our question decomposer, we leveraged ChatGPT, which is known for its data augmentation ability. Comprehensive experiments were conducted using the HotpotQA dataset. Our method outperformed several established approaches, including the *Chain-of-Thoughts* approach, which is based on large language models. Moreover, our results are on par with those of state-of-the-art *Retrieve-then-Read* methods that utilize large language models.

KEYWORDS
chain-of-reasoning, data augmentation, language models, open-domain question answering, question decomposition

1 | INTRODUCTION

Following the recent emergence of large-scale language models (LMs), methodologies for complex question answering (QA) have improved. What was once scarcely attempted-the chain-of-reasoning–has now become a reality, yielding significant performance improvements as evidenced by various methods [1]. Although large language models (LLMs) have powerful capabilities, they also present inherent challenges. The first challenge relates to hallucinations. There have been instances where the model derived false inferences during the reasoning process instead of relying on factual data [1, 2]. Furthermore, LLMs face the challenge of becoming outdated quickly; once trained, they fail to incorporate the new knowledge that emerges subsequently. Retrieval-based LMs have been shown to effectively address hallucinations and outdated information issues [3]. Because of these attributes, the *Retrieve-then-Read* pipeline has gained popularity among LLMs [4-6].

LLMs are also very expensive to operate based on their incredibly high parameter count. For example, GPT-3 boasts 175 billion model parameters, whereas T5-base/large models feature 220/770 million parameters.

Jihyeon Roh and Minho Kim equally contributed to this work.

Moreover, by taking into account the size ratio alone, an approximate difference of 800x/250x exists. Finetuning definitely helps the LLMs, but the effort alone causes notable cost increases. On the other hand, small models complemented by information retrieval and finetuning methods have recently demonstrated comparable or superior performance results at much lower costs [7-9]. Synthesizing the aforementioned viewpoints, retrieval-based, small-sized LMs exhibited greater efficiency in addressing problems than their large-scale counterparts while concurrently mitigating hallucinations and outdated knowledge issues. We consider models with billions of parameters as "large" and those with millions of parameters as "small."

The objective of our study is to present a system with efficient, small LMs to adequately handle question-answering tasks that require open-domain exploration, multihop searches, and chain-of-reasoning processes while minimizing the reliance on large-scale counterparts. We employ retrieval-based LMs to attain a performance level similar to that of LLMs. Furthermore, this study analyzes the problem types appearing in complex question sets and offers chain-of-reasoning pipelines tailored to each specific type.

A challenging aspect of the proposed architectural configuration is question decomposition. Although this can be accomplished using generative LMs [10], preparing the required training data is difficult. As is commonly understood, involving humans is the most effective approach to acquiring high-quality data. However, one drawback is the substantial cost associated with this method. Recently, LLMs have shown their potential for generating high-quality texts [11]. Consequently, we employed the well-known LLM, ChatGPT, to generate training data for our question decomposer.

During our experiments, we observed that the proposed reasoning approach using small LMs outperformed or closely matched those of methods reliant on high-cost LLMs. We also conducted a series of experiments involving different reasoning pipelines to identify the essential components that can enhance the effectiveness of the proposed chain-of-reasoning pipelines. Our contributions can be summarized as follows:

1. We introduced *EffiChainQA*, a new chain-of-reasoning framework for open-domain QA that leverages efficient, small LMs instead of LLMs.
2. We implemented question decomposition, a crucial component of chain-of-reasoning, using a training set derived from a data augmentation based on LLMs.
3. In testing, our method outperformed some *Chain-of-Thoughts* approaches [1, 12], which rely on LLMs, and demonstrated performance comparable to the latest *Retrieve-then-Read* techniques that also use LLMs.

2 | RELATED WORKS

2.1 | Retrieval-augmented LM

Retrieval-based LMs are essential for open-domain QA. In particular, the Fusion-in-Decoder (FiD) model [13] exhibits improved performance by selecting and integrating multiple retrieved documents. FiD has been investigated in various follow-up studies due to its excellent performance [14, 15]. In particular, the method [14] (we refer to as EviQA) jointly trains two modules, one that selects evidence from retrieved results and another that generates answers. This approach improves performance while providing explanations through evidence selection. Numerous studies have been conducted on retrieval-based LLMs [3, 16, 17] to address the hallucinations and outdated issues associated with LLMs. In this study, we applied the FiD and EviQA methods as the answer generators in our chain-of-reasoning pipelines.

2.2 | Complex question decomposition

A complex question necessitates its division into simpler sub-questions that are relatively easier to answer. For question decomposition, there are established strategies such as symbolic methods based on logical approaches [18, 19], as well as methods that rely on learning models using purely natural language text [10, 20, 21]. To train a generative LM for question decomposition, a large number of sets comprising original questions and their decomposed sub-questions are needed. In previous studies, such paired sets were created either through massive human effort [21, 22] or machine-generated approaches using certain heuristics [10]. Recently, numerous studies have been leaning towards using generative LMs to augment training data [11, 23]. Moreover, LLMs have been able to produce high-quality data that are more time and cost-effective compared to human-generated content [24]. In this paper, we utilize the dataset generated by LLM to train a question decomposer model using a small LM.

2.3 | Chain-of-reasoning

Complex problems are often addressed by chaining the reasoning across multiple steps. Methods based on LLMs have recently attracted attention for this purpose. Starting with the Chain-of-Thoughts (CoT) [1], subsequent innovations have emerged. For instance, self-consistency (SC) [12] introduced a technique for self-checking the reasoning paths generated by LLMs, whereas ReAct [25]

proposed an iterative method for redesigning action plans and reasoning plans based on the outcomes of previous actions. The Rewrite-Retrieve-Read methodology [5] enhances the conventional *Retrieve-then-Read* paradigm by integrating a query-rewriting module. This inclusion facilitated dynamic and adaptive data retrieval based on the output of the previous reader. In a related advancement, Yu et al. [26] improved the results of LLM by providing the retrieved results as an in-context demonstration of the subsequent reasoning process.

Recent studies proposed several methods for the construction of reasoning chains by combining various external tools [27-29]. When provided with definitive inputs, utilizing an external tool that yields clear outcomes can minimize uncertainties. This is especially valuable compared with the case where processing solely depends on LLMs, which occasionally exhibit hallucination issues. Notably, these methods offer the advantage of generating explanations for their results. By contrast, Ma et al. [30] introduced a method using small LMs to solve a task by creating reasoning paths using core entities and hyperlinks.

This study addresses the chain-of-reasoning. Our objective is to establish a chain-of-reasoning pipeline that predominantly uses small LMs, thus ensuring minimal reliance on LLMs. Our proposed method can also be viewed as a type of tool-based LM because we utilize different reasoning pipelines depending on the question type.

3 | METHOD

Figure 1 presents an overview of our reasoning pipeline. The pipeline extends from the framework shown in Figure 1A to that shown in Figure 1B. Figure 1B illustrates *EffiChainQA*. Our pipeline consists of three primary modules: *reasoning-type classifier, question decomposer*, and *RecomposeNet*.

The first module, the reasoning-type classifier, serves as a discriminator that determines the reasoning type upon receiving a question to facilitate path planning. As shown in Figure 1B, questions are categorized into "bridge" and "comparison". Based on the HotpotQA dataset, we assume that complex questions fall

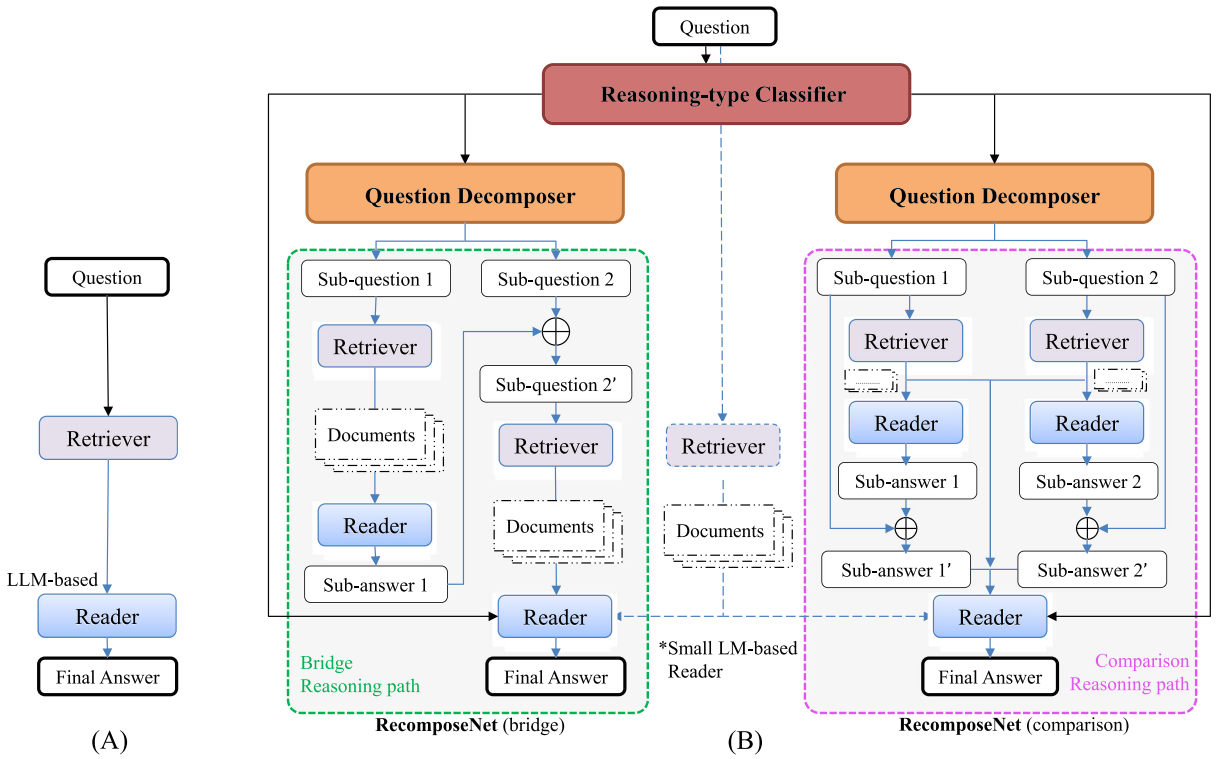

FIGURE 1 Overview of our proposed pipeline: (A) simple *Retrieve-then-Read* system using large language models (LLMs) and (B) our extended *Retrieve-then-Read* framework with modularized reasoning components. In (B), we initiate the process by classifying the reasoning types with the *reasoning-type classifier*. Then, the multihop questions are decomposed into more simpler sub-questions using the *question decomposer*. Finally, *RecomposeNet*, using small LMs in Reader, integrates the sub-questions and sub-answers to generate final answers for each distinct reasoning path. The symbol ⊕ represents the concatenation operator. Corresponding to the reasoning types, we adopt two distinct reasoning paths: bridge and comparison.

predominantly into these two categories. Although the questions could be categorized more intricately, we adhered to the types provided in dataset. The third module has two distinct reasoning paths. Even with the addition of new reasoning types for a question, this architecture remains applicable. For example, when introducing the "intersection" type, the classifier can be updated accordingly followed by the addition of a new reasoning path. A detailed classification methodology is presented in Section 3.1.

The second module is the question decomposer, as shown in Figure 2, which decomposes a question into simpler sub-questions. Although this decomposition is compatible with a range of LMs, we specifically leveraged two models in our study: ONUS, which operates on a small LM, and ChatGPT, which is a representative LLM. The question decomposition method is described in detail in Section 3.2.

The third module is RecomposeNet, in which different reasoning paths are pursued, depending on the reasoning type of the question. The decomposed questions (i.e., sub-questions) and original question pair were used as inputs for each reasoning path based on reasoning type, ultimately leading to the derivation of the final answer. Details of the RecomposeNet are provided in Section 3.3.

3.1 | Reasoning-type classifier

To generate final answers tailored to the reasoning types of questions, our system adopts specific reasoning processes. In the initial step, we implemented a module dedicated to classifying question types. This module accepts a question as input and yields reasoning type t as the output, where $t \in$ {bridge, comparison}. The outcomes of type classifications play a crucial role in the third module and determine the direction of the reasoning paths.

For reasoning-type classification, we explored several LMs, including nonfine-tuned LLMs (e.g., GPT-3.5 and GPT-4) and a trainable BERT model [31]. For the LLMs, we utilized an in-context learning approach via ChatGPT to determine the types of questions. Four question-type pairs were used as context examples, with two examples each for the bridge and comparison types.

For the BERT, we fine-tuned the model using the question-type pairs from the HotpotQA dataset.

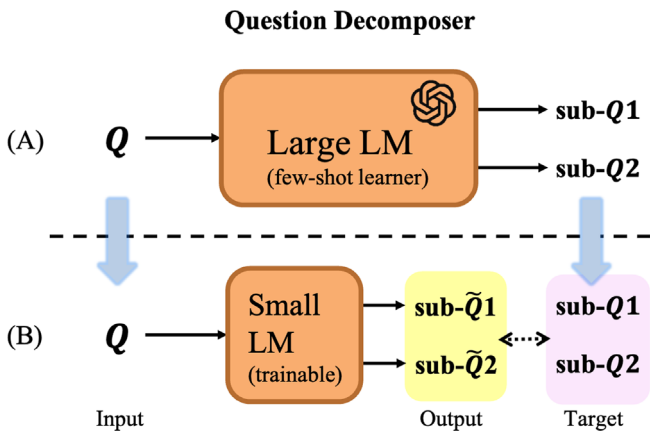

FIGURE 2 A complex question, denoted as $\boldsymbol{Q}$, is broken down into simpler questions (either **sub-*Qn*** or **sub-$\tilde{Q}$*n***) via the question decomposer, which can be based on either (A) (few-shot learner) large language model (LLM) or (B) a (trainable) small language model (LM). In (B), we use **sub-*Qn*** derived from the LLM as the target for model training. Either **sub-*Qn*** or **sub-$\tilde{Q}$*n*** is used as the input for the reasoning pipeline.

3.2 | Question decomposer

3.2.1 | Question decomposition using LLMs

In Figure 2A, we utilize ChatGPT API (a variant of GPT-3.5) to construct a question decomposition corpus using an in-context learning approach (also known as few-shot learning) [32]. This process entailed the presentation of instructions along with several examples of prompts, followed by a target question. Some examples were manually crafted. We provided different in-context examples and instructions, depending on the reasoning type (bridge or comparison). Based on this process, we obtained a substantial number of question and decomposition pairs. The decompositions derived in this study are referred to as pseudo-decompositions. These serve either as a training corpus for a smaller question decomposition LM or a direct input to RecomposeNet.

3.2.2 | Question decomposition using small LM

Unlike the ONUS approach [10], which extracts simple sub-question examples from a Common Crawl, our method emphasizes the autonomous generation of a dataset without human intervention facilitated by the use of LLMs. Consequently, we employed pseudo-decompositions generated by LLMs to train the ONUS, as depicted in Figure 2B. The resulting model is referred to as "ONUS +L". The ONUS system adopted XLM [33] due to its notable performance in both translation and decomposition, achieved using the back-translation technique. During inference, ONUS+L generates sub-questions given an input question. These outputs then act as inputs for the subsequent modules.

3.3 | RecomposeNet for each reasoning path

Building on the inherent characteristics of the different question types, we designed RecomposeNet with distinct reasoning paths for each type. Both bridge and comparison networks consistently incorporate the *retriever* and *reader* models. The input and output components of these networks encompass the original question, sub-questions, sub-answers, retrieval results for the sub-questions, and retrieval results for the original question. These are denoted as Q, sub-Qs, sub-As, $R_{sub\text{-}Qs}$, and R_Q, respectively. We represent the *n*th sub-question and its answer as sub-Q*n* and sub-A*n*, respectively.

In our experiments, we utilized the Contriever search engine from ATLAS [7] for the *retriever*. This engine is specifically fine-tuned for Natural Questions (NQ) [34] and is denoted as Contriever–ATLAS–NQ. For the *reader*, we employed either EviQA or FiD to generate answers from the retrieved multiple documents.

This subsection details the method used to derive answers for both bridge and comparison types.

3.3.1 | Reasoning path of bridge type

The bridge type has a predetermined order between the decomposed sub-questions. To determine sub-A2, we typically need to address sub-A1. This process proceeds as follows:

1) **Retrieval for sub-Q1:** Using Contriever–ATLAS–NQ, we extracted 20 relevant paragraphs from the external datastore that aligned closely with sub-Q1.
2) **Generate answer for sub-Q1:** We utilized EviQA which was fine-tuned to the NQ dataset (referred to as EviQA–NQ), as a reader. EviQA–NQ processes the retrieved documents alongside sub-Q1 to produce the corresponding answer (sub-A1).
3) **Retrieval for sub-Q2:** The sub-A1 plays a pivotal role in addressing sub-Q2 and serves as a bridge between them. Hence, we substitute this answer with a bridge word in sub-Q2, denoted as sub-Q2′ in Table 1. To simplify this process, we can create the sub-Q2′ by concatenating sub-A1 and sub-Q2.
4) **Generate final answer:** The final *reader* acts as the final answer generator. The input can include either sub-Q2′ or Q. Notably, by leveraging the original question, our system adeptly addresses those questions labeled as "bridge type" even if they do not strictly fit the bridge-type definition, instead of aligning more closely with single-hop or intersection types.

TABLE 1 Example of bridge-type question and its sub-questions.

Q: Who was the biological father of the emperor who built the Nemi ships?
sub-Q1: Who was the emperor that built the Nemi ships?
sub-Q2: Who was the biological father of that emperor?
sub-A1: Caligula
sub-Q2′: Who was the biological father of Caligula?

Note: sub-Q2′ is formed by replacing the bridge word in sub-Q2 with the answer to sub-Q1 (i.e., sub-A1).

For instance, the query *"Kathleen Matthews holds what political position in the Maryland Democratic Party?"* is more of a single-hop type than a bridge-type question. To ensure system robustness, we structured the input for the final reader by incorporating the retrieval results from both Q and sub-Q2′. Specifically, the input comprises three primary components: [(Q or sub-Q2′), R_Q, and $R_{sub\text{-}Q2'}$], where R_Q and $R_{sub\text{-}Q2'}$ are the retrieval results for Q and sub-Q2′, respectively. Section 4.7.1 presents a detailed ablation study on this combination. We employed either FiD or EviQA as the final readers, both fine-tuned for HotpotQA.

3.3.2 | Reasoning path of comparison type

Unlike the bridge type, the comparison type does not have a sequential relationship between its sub-Qs, and no bridge substitute connects them. This path was designed to independently search for and generate sub-As. The details are as follows.

1) **Retrieve and generate answers for sub-Q1 and sub-Q2:** Similar to the approach for the sub-Q1 of bridge type, Contriever–ATLAS–NQ was applied as a retriever, and EviQA–NQ was applied as an answer generator. Each sub-question was used as an independent input for the retriever and reader, ultimately obtaining answers for each sub-question.
2) **Generate final answer:** To ensure a precise final answer, Q was used as the input for the final reader. While sub-Qs aid in discerning the final answer, their answers contribute evidence rather than provide the definitive answers themselves. This is illustrated in the example in Table 2. Moreover, it is essential to include context in the input. This context comprises a combination of augmented sub-As and the retrieval results from the retriever. The augmented sub-As are constructed by concatenating the sub-As along with the corresponding sub-Qs. Specifically, sub-A1′ = [sub-

TABLE 2 Example of comparison-type question and its sub-questions.

Q: Who was born first, Arthur Conan Doyle or Penelope Lively?
sub-Q1: When was Arthur Conan Doyle born?
sub-Q2: When was Penelope Lively born?
sub-A1: 22 May 1859
sub-A2: 17 March 1933
final answer: Arthur Conan Doyle

Note: The answers to the sub-questions (i.e., sub-A1 and sub-A2) do not align with the final answer.

Q1; sub-A1] and sub-A2′ = [sub-Q2; sub-A2]. These augmented sub-As are represented as sub-A′s = [sub-A1′; sub-A2′]. The retrieval results are a combination of R_Q and $R_{sub\text{-}Qs}$. Thus, the full context is represented as context = [sub-A′s; $R_{sub\text{-}Qs}$; R_Q]. This configuration is flexible and subject to adjustment. A detailed ablation study exploring various combinations of these components is presented in Section 4.7.2.

4 | EXPERIMENTS

4.1 | HotpotQA dataset

To evaluate our multistep reasoning method, we used HotpotQA, a renowned corpus that encompasses both multihop QA and open-domain QA formats [35]. Questions within the corpus were categorized into two reasoning types: bridge and comparison. Specifically, the bridge types accounts for approximately 80 %, whereas the comparison type accounts for approximately 20 %, as shown in Table 3.

In our study, which targeted open-domain QA, we assessed our system using the full Wiki settings of the HotpotQA dataset. For this purpose, we used the December 2021 Wikipedia dump as our primary knowledge source. To structure this external knowledge, we adopted the approach used in a previous study [7]. Each document (article) was divided into sections, and sections longer than 200 words were further divided. This process yielded 37 M paragraphs, which were then converted into dense passage vectors using Contriever.

4.2 | Data augmentation by ChatGPT

To train the small-question decomposer, ONUS+L, we employed the ChatGPT-3.5-turbo application programming interface (API) to generate the training corpus. Each API call included few-shot, in-context learning examples, each with instructions tailored to its corresponding type, as listed in Table 4. We experimented with one to three in-context examples, and the performance was good when two or more examples were used. Although providing three examples is likely to yield better results, we opted for two, considering the trade-off between API call costs and the quality of outcomes.

4.3 | Pipeline configuration

The composition of our reasoning pipelines is represented as {<1st module method>−<2nd module method>−<3rd module method>}, with examples provided in rows 10–14 of Table 5. The first module employs

TABLE 3 Data statistics for HotpotQA.

	Bridge	Comparison	All
train	72 991	17 456	90 447
dev	5918	1487	7405
total	78 909	18 934	97 852

Note: Number of train/dev data for each question type are shown.

TABLE 4 Example instructions per reasoning type ([a] bridge and [b] comparison) applied to ChatGPT for generating sub-questions from a given multihop question.

(a) Bridge type
Divide a question into two sub-questions.
question: *What government position was held by the woman who portrayed Corliss Archer in the film Kiss and Tell?*
sub-questions:
1) *Which woman portrayed Corliss Archer in the film Kiss and Tell?*
2) *What government position was held by the woman?*
question: *Who was the biological father of the emperor who built the Nemi Ships?*
sub-questions: {}
(b) Comparison type
Generate two complementary questions from a question.
question: *Who is older, Annie Morton or Terry Richardson?*
complementary questions:
1) *How old is Annie Morton?*
2) *How old is Terry Richardson?*
question: *Who was born first, Arthur Conan Doyle or Penelope Lively?*
complementary questions:{}

Note: Owing to article space limitations, only a single example is provided here.

BERT, ChatGPT, or *Given* as reasoning-type classifiers. Here, *Given* refers to the labels provided in the HotpotQA corpus that were directly annotated by humans. The second module employs ChatGPT (specifically, GPT-3.5-turbo) or ONUS+L as the question decomposer. In the third module, we used two distinct models: UniG and SepG. UniG represents an answer generator trained on an integrated corpus that includes both bridge and comparison reasoning types and serves as a unified answer-generator for both types. In contrast, SepG employs separate answer generators trained individually for each type. UniG utilizes the EviQA framework, which is renowned for its superior performance compared with the FiD model. However, owing to its considerably higher training costs, the SepG model employs a more cost-efficient FiD model.

4.4 | Overall performance

We compared the proposed method to several baseline methods using the HotpotQA dataset. Table 5 lists the performance of all baselines and our system based on various configurations. The proposed techniques, especially Ours$^{\text{BERT–ChatGPT–SepG}}$ and Ours$^{\text{Given–ChatGPT–SepG}}$, exhibited enhancements of more than 15- and 10-points in EM performance compared with CoT-prompting and SC, respectively. Additionally, the methods presented in this study showcase state-of-the-art performance in the context of *Retrieve-then-Read* algorithms, such as Rewrite-Retrieve-Read and ReFeed. Notably, the remarkable aspect lies in the superior or comparable performances compared with the methods employing LLMs throughout their entire process.

TABLE 5 Performance of small and large LMs on the HotpotQA (full wiki) dev dataset.

Base model (size)	Method	EM
-	HopotQA-baseline [35]	24.68
T5 (770 M)	EviQA [14]	22.82
ChatGPT (175 B)	Rewrite–Retrieve–Read [5]	33.70
ChatGPT (175 B)	ReFeed [26]	43.50
ChatGPT (175 B)	ReFeed+CoT [26]	44.20
PaLM (540 B)	Chain-of-Thoughts (CoT) prompting [1]	28.90
PaLM (540 B)	Self-Consistency (SC) [12]	33.80
PaLM (540 B)	TooLearningFM [28]	35.50*
PaLM (540 B)	ReAct + CoT-SC [25]	35.10
T5 (770 M)	Ours$^{\text{BERT–ONUS+L–UniG}}$	34.28
	Ours$^{\text{BERT–ChatGPT–UniG}}$	41.74
	Ours$^{\text{BERT–ChatGPT–SepG}}$	**44.34**
	Ours$^{\text{Given–ChatGPT–UniG}}$	41.60
	Ours$^{\text{Given–ChatGPT–SepG}}$	**44.34**

Note: The best results among all experiments are highlighted in bold. The star(*) symbol denotes results derived from a subset of 200 randomly selected samples.
Abbreviation: LM, language model.

Our system encompasses up to three retrievers, three generators (readers), a classifier, and a question decomposition process. In comparison, the ReFeed [26] method involves two generators and one retriever. While Refeed involves fewer generators and retrievers than our system, it requires more computational resources overall, due to its multiple uses of ChatGPT. The key factors affecting computation and execution times include parameter size of the model and the length of the decoder sequence. Our model excelled in efficiency, with 250 times fewer parameters (Ours: 770M vs. ReFeed: 175B) and a 256 times shorter decoder sequence length (Ours: 16 vs. ReFeed: 4096). Given these substantial differences, our method is expected to significantly improve computational resource efficiency and reduce execution time.

Furthermore, the pipeline that used BERT for type prediction exhibited the same performance as the pipeline that utilized the *Given* state provided in the problem set. Ours$^{\text{BERT–ONUS+L–UniG}}$ method, which does not use LLM, outperformed some of the algorithms that relied solely on LLM.

Within the entire pipeline, the model utilizing the LLM (ChatGPT) solely for question decomposition yields a 10.06-point performance improvement compared with the model that did not use the LLM at all (34.28 vs. 44.34 EM). This demonstrates the potential of improving the question decomposer using smaller LMs.

4.5 | Results on varying pipeline components

Table 6 presents the experimental comparisons based on variations in the module compositions of the reasoning pipeline. The overall performance ("all" columns in Table 6) was calculated as the micro average of the bridge- and comparison-type EMs.

Regarding the different types of classifiers, our fine-tuned $\text{BERT}_{\text{base}}$ outperformed both GPT-3.5 and GPT-4 and performed on par with the *Given* setting. Note that the number of types predicted by GPT-3.5 varied significantly from those of *Given*. This discrepancy can be attributed to the subpar performance of type classifications, which affect the overall performance. For the GPT-3.5–ChatGPT–UniG system, approximately 800 noisy and complex questions initially annotated as bridge type may have been

TABLE 6 Results from our pipelines combining various models.

Pipeline			EM			Number of questions	
Type classifier	**Decomposer**	**Reader**	**bridge**	**comparison**	**all**	**bridge**	**comparison**
Given	ChatGPT	UniG	38.89	52.39	41.60	5918	1487
Given	ChatGPT	SepG	39.02	**65.50**	**44.34**	5918	1487
GPT-4	ChatGPT	UniG	38.82	53.24	41.74	5908	1497
GPT-3.5	ChatGPT	UniG	**39.11**	42.57	40.17	5143	2262
BERT	ONUS+L	UniG	32.90	39.58	34.28	5879	1526
BERT	ChatGPT	UniG	38.86	52.82	41.74	5879	1526
BERT	ChatGPT	SepG	<u>39.04</u>	<u>64.74</u>	**44.34**	5879	1526

Note: We report exact match (EM) scores for the Hotpot dev dataset. The terms "bridge" and "comparison" denote the reasoning types associated with questions. The term "all" signifies that all the samples in the dev set, including bridge and comparison types, were used for evaluation. The values in the "Number of questions (bridge/comparison)" columns showcase the changes in dataset composition when different type classifiers were applied. The highest value is marked in bold, and the second highest value is underlined.

misclassified by the GPT-3.5 classifier and transferred to the comparison group. This misclassification could potentially inflate the performance metrics for the bridge type while adversely affecting those for the comparison type. As a result, the overall performance of the system was reflected in the second-worst EM score of 40.17. In contrast, the BERT–ChatGPT–SepG system, which nearly accurately classified both bridge and comparison types, attained the second-best performance in each category. This level of accuracy contributed to it achieving the highest overall performance.

Regarding the different question decomposer, ChatGPT outperformed ONUS+L. Regarding the final *reader*, we observed that SepG performed better than UniG. Especially, when training the SepG model solely on the corpus of comparison-type questions, a performance improvement of 13.11 points (from 52.39 to 65.50) was observed. In contrast to the bridge type (from 38.89 to 39.02), a significant performance improvement was achieved. Regarding the size of the training corpus, the comparison type was approximately 1/4 that of the bridge type. The same ratio was employed when training UniG, which led to the speculation that the UniG model may have been at a disadvantage in learning from the comparison type because of this ratio. Overall, the "BERT–ChatGPT–SepG" and "Given–ChatGPT–SepG" pipelines exhibited the highest performance. EM performance across different question reasoning types matched the overall performance pattern.

4.6 | Reasoning-type classifier performance

Table 7 presents the performance outcomes of the different models in reasoning-type classifications.

TABLE 7 Classification accuracy for reasoning type of questions in the HotpotQA dev set.

	Accuracy (%)		
Models	**Bridge**	**Comparison**	**All**
OpenAI GPT-3.5	85.5	94.4	87.3
OpenAI GPT-4	97.4	90.3	96.0
Finetune-BERT$_{base}$	**98.9**	**98.1**	**98.7**

Note: The "all" column was calculated using micro average. Bold scores indicate the best performances.

This table provides empirical findings regarding (1) the performance variation among different LLM models, (2) the effect of the fine-tuned small LM, and (3) a detailed breakdown of performance by reasoning type.

Regarding (1), both GPT-3.5 and GPT-4 demonstrated performances greater than 87% even though they are few-shot models. We also confirmed that GPT-4 outperformed GPT-3.5 by roughly 9% in our tests. Regarding (2), the performance of the fine-tuned BERT model was approximately 2.7% better than that of the few-shot LLMs. Despite its small size, significant performance improvements were observed when the model was trained using task-specific data. Regarding (3), GPT-4 performed better than GPT-3.5 for bridge-type questions. However, for the comparison type, GPT-3.5 outperforms GPT-4. Due to the relatively higher number of bridge-type questions compared with comparison-type questions, enhancing the performance of bridge-type questions can substantially contribute to overall performance improvement. Finally, for both bridge- and comparison-type questions, the Finetune-BERT$_{base}$ model outperformed the LLMs.

4.7 | Ablation studies

Given that each type of reasoning pipeline contains unique components, there is leads to a range of configurations depending on the type. We describe additional analyses focusing on specific components of the reasoning pipeline. The effects of removing or altering these components were evaluated.

4.7.1 | Study for bridge-type configuration

Table 8 lists the performance impact outcomes when certain components were excluded or replaced in the final answer generator. We conducted an experimental comparison to assess the effect of the two approaches. The first involved solely using sub-Q2′, whereas the second approach utilized information from the original question instead of sub-Q2′. Ideally, in a bridge-type scenario, it should be possible to determine the answer using only sub-Q2′.

However, relying solely on sub-Q2′ resulted in lower EM scores. To investigate the cause of this underperformance, we conducted a human evaluation to confirm instances in which the original question did not fit the bridge type. For instance, "*Where did recording sessions take place for the Michael Jackson hit Beat It?*" is a single-hop type, and "*Who was the writer of These Boots Are Made for Walkin' and who died in 2007?*" is an intersection type; both of these are types that do not necessitate a bridge. In HotpotQA, it was stated that 31% of the questions comprised intersection or single-hop types and not the bridge type. Acknowledging the limitations of using sub-Q2′ in these instances, we opted for an alternative approach that utilizes the original question (Q) as the final question.

By replacing the partial question with the original question, the EM score improved to 34.27, an increase of 6.01 points. Moreover, when we integrated the original question and R_Q, the EM score further increased to 38.89, representing an additional gain of 4.26 points (totaling a 10.63-point improvement). In conclusion, our findings indicate that that use of inappropriate question types in an evaluation set can result in incorrect question decompositions. We demonstrated that incorporating information from the original question significantly enhances the overall performance of the bridge type.

TABLE 8 Comparison of EM outcomes for the bridge type based on Reader input configurations: including either Q or sub-Q2′ and with or without R_Q.

Question	R_Q	EM	Difference
Sub-Q2′	✓	28.26	
Q	x	34.27	+6.01
	✓	38.89	+10.63

Note: Herein, R_Q represents the retrieval results for the original question (Q), and sub-Q2′ denotes the augmented sub-question 2. The retrieval results for the sub-Q2′ is included in the input by default.
Abbreviation: EM, exact match.

TABLE 9 Performance comparison of EM for the comparison type, considering different Reader input configurations: with/without the augmented sub-As (sub-A′s) and with/without the retrieval results for the original question (R_Q).

sub-A′s	R_Q	EM	Difference
✓	✓	52.05	
✓	x	52.39	+0.34
x	x	40.08	−11.97

Note: By default, the original question (Q) and the retrieval results for sub-questions ($R_{sub\text{-}Qs}$) are always included in the input.
Abbreviation: EM, exact match.

4.7.2 | Study for comparison-type configuration

In Table 9, similar to the bridge-type study, we conducted an experimental comparison between the impact of the retrieval results of the original question (R_Q) and the influence of the augmented sub-answers (sub-A′s).

First, we found that the retrieval results of the original question (R_Q) negatively affected the performance of the comparison type, in contrast to the bridge type. This is likely caused by the original question in the comparison type involving two different entities, which led to a high likelihood of noisy search results due to the mixing of entity information. Thus, avoiding the use of R_Q with this type appears to be beneficial to enhancing the overall performance (+0.34). Second, excluding the use of sub-A′s resulted in a notable decrease in performance. This indicates that the sub-A′s are crucial for determining the final answer in the comparison type.

5 | CONCLUSION

In this paper, we introduced *EffiChainQA*, an efficient chain-of-reasoning framework for multi-hop open-domain QA tasks that leverages small LMs. Our *EffiChainQA* consists of three modules: a reasoning-type classifier, a question decomposer, and the RecomposeNet for each reasoning type. Beginning with the classifier, the reasoning type for incoming questions was discerned, primarily categorizing them as bridges or comparisons. The decomposer then simplified these questions into sub-

questions by leveraging ONUS+L, a small LM, or ChatGPT, a larger LM. The final module, RecomposeNet, utilized both sub-questions and original questions to derive the final answer for the identified type. Instead of relying on LLMs, we designed a chain-of-reasoning pipeline that used smaller LMs.

The performance of the proposed method was analyzed using the HotpotQA dataset. Our method showed improved EM performances by more than 15 points compared with *CoT* and by more than 10 points compared with *SC*. Moreover, it exhibited slightly better or comparable performance over the retrieval-based LLM method. Our system, which utilizes ONUS+L, yielded better results than the Rewrite-Retrieve-Read method. Based on the results from the system utilizing ONUS+L, we observed that the chain-of-reasoning with small LMs has the potential to exhibit capabilities comparable to that of LLMs. Moreover, various comparative ablation experiments have been conducted. For bridge type, we observed that the use of the original question when generating the final answer had an impact on performance improvement. For the comparison type, the sub-answers had a significant impact on performance.

CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest.

ORCID

Jihyeon Roh https://orcid.org/0000-0002-1708-7259
Kyoungman Bae https://orcid.org/0000-0001-9007-4027

REFERENCES

1. J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. H. Chi, Q. V. Le, and D. Zhou, *Chain-of-Thought prompting elicits reasoning in large language models*, Vol. **35**, 2022, pp. 24824–24837.
2. J. Maynez, S. Narayan, B. Bohnet, and R. McDonald, *On faithfulness and factuality in abstractive summarization*, arXiv preprint, 2020, DOI 10.48550/arXiv.2005.00661
3. A. Lazaridou, E. Gribovskaya, W. Stokowiec, and N. Grigorev, *Internet-augmented language models through few-shot prompting for open-domain question answering*, arXiv preprint, 2022, DOI 10.48550/arXiv.2203.05115
4. H. He, H. Zhang, and D. Roth, *Rethinking with retrieval: faithful large language model inference*, arXiv preprint, 2022, DOI 10.48550/arXiv.2301.00303
5. X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan, *Query rewriting for retrieval-augmented large language models*, arXiv preprint, 2023, DOI 10.48550/arXiv.2305.14283
6. W. Shi, S. Min, M. Yasunaga, M. Seo, R. James, M. Lewis, L. Zettlemoyer, and W. Yih, *REPLUG: retrieval-augmented black-box language models*, arXiv preprint, 2023, DOI 10.48550/arXiv.2301.12652
7. G. Izacard, P. Lewis, M. Lomeli, L. Hosseini, F. Petroni, T. Schick, J. Dwivedi-Yu, A. Joulin, S. Riedel, and E. Grave, *Atlas: Few-shot learning with retrieval augmented language models*, arXiv preprint, 2022, DOI 10.48550/arXiv.2208.03299
8. S. Min, W. Shi, M. Lewis, X. Chen, W. Yih, H. Hajishirzi, and L. Zettlemoyer, *Nonparametric masked language modeling*, (Findings of the Association for Computational Linguistics), 2023, pp. 2097–2118, DOI 10.18653/v1/2023.findings-acl.132
9. W. Yu, *Retrieval-augmented generation across heterogeneous knowledge*, (Proc. NAACL: Human Language Technologies: Student Research Workshop), 2022, pp. 52–58, DOI 10.18653/v1/2022.naacl-srw.7.
10. E. Perez, P. Lewis, W. Yih, K. Cho, and D. Kiela, *Unsupervised question decomposition for question answering*, (Proc. Empirical Methods in Natural Language Processing), 2020, pp. 8864–8880.
11. H. Dai, Z. Liu, W. Liao, X. Huang, Y. Cao, Z. Wu, L. Zhao, S. Xu, W. Liu, N. Liu, S. Li, D. Zhu, H. Cai, L. Sun, Q. Li, D. Shen, T. Liu, and X. Li, *AugGPT: leveraging ChatGPT for text data augmentation*, arXiv preprint, 2023, DOI 10.48550/arXiv.2302.13007
12. X. Wang, J. Wei, D. Schuurmans, Q. V. Le, E. H. Chi, S. Narang, A. Chowdhery, and D. Zhou, *Self-consistency improves chain of thought reasoning in language models*, (International Conference on Learning Representations, Kigali, Rwanda), 2022.
13. G. Izacard and E. Grave, *Leveraging passage retrieval with generative models for open domain question answering*, (Proc. of the 16th Conf. of the European Chapter of the Association for Computational Linguistics), 2021, pp. 874–880.
14. A. Asai, M. Gardner, and H. Hajishirzi, *Evidentiality-guided generation for knowledge-intensive NLP tasks*, (Proc. of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, WA, USA), 2022, pp. 2226–2243.
15. S. Hofstätter, J. Chen, K. Raman, and H. Zamani, *Fid-light: efficient and effective retrieval-augmented text generation*, arXiv preprint, 2022, DOI 10.48550/arXiv.2209.14290
16. Y. Levine, O. Ram, D. Jannai, B. Lenz, S. Shalev-Shwartz, A. Shashua, K. Leyton-Brown, and Y. Shoham, *Huge frozen language models as readers for open-domain question answering*, (ICML 2022 Workshop on Knowledge Retrieval and Language Models), 2022.
17. S. Zheng, J. Huang, and K. C.-C. Chang, *Why does ChatGPT fall short in answering questions faithfully? arXiv preprint*, 2023, DOI 10.48550/arXiv.2304.10513
18. Z. Deng, Y. Zhu, Y. Chen, M. Witbrock, and P. Riddle, *Interpretable AMR-based question decomposition for multi-hop question answering*, arXiv preprint, 2022, DOI 10.48550/arXiv.2206.08486
19. Y. Liu, S. Yavuz, R. Meng, D. Radev, C. Xiong, and Y. Zhou, *HPE: Answering complex questions over text by hybrid question parsing and execution*, arXiv preprint, 2023, DOI 10.48550/arXiv.2305.07789
20. J. Li, M. Ren, Y. Gao, and Y. Yang, *Ask to understand: question generation for multi-hop question answering*, arXiv preprint, 2022, DOI 10.48550/arXiv.2203.09073
21. S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, *Multi-hop reading comprehension through question decomposition and rescoring*, (Proc. Annual Meeting of the Association for Computational, Linguistics, Florence, Italy), 2019, pp. 6097–6109.

22. M. Bevilacqua, R. Blloshmi, and R. Navigli, One spring to rule them both: symmetric AMR semantic parsing and generation without a complex pipeline, (Proc. AAAI Technical Track on Speech and Natural Language Processing), Vol. **35**, 2021, pp. 12564–12573.
23. E. Chung and J. G. Park, *Sentence-chain based seq2seq model for corpus expansion*, EERI J. **39** (2017), no. 4, 455–466.
24. H. You, R. Sun, Z. Wang, L. Chen, G. Wang, H. A. Ayyubi, K.-W. Chang, and S.-F. Chang, *IdealGPT: iteratively decomposing vision and language reasoning via large language models*, arXiv preprint, 2023, DOI 10.48550/arXiv.2305.14985
25. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, *ReAct: Synergizing reasoning and acting in language models*, (International Conference on Learning Representations), 2022.
26. W. Yu, Z. Zhang, Z. Liang, M. Jiang, and A. Sabharwal, *Improving language models via plug-and-play retrieval feedback*, arXiv preprint, 2023, DOI 10.48550/arXiv.2305.14002
27. P. Lu, B. Peng, H. Cheng, M. Galley, K.-W. Chang, Y. N. Wu, S.-C. Zhu, and J. Gao, *Chameleon: Plug-and-play compositional reasoning with large language models*, arXiv preprint, 2023, DOI 10.48550/arXiv.2304.09842
28. Y. Qin, S. Hu, Y. Lin, W. Chen, N. Ding, G. Cui, Z. Zeng, Y. Huang, C. Xiao, C. Han, and Y. R. Fung, *Tool learning with foundation models*, arXiv preprint, 2023, DOI 10.48550/arXiv.2304.08354
29. T. Schick, J. Dwivedi-Yu, R. Dessì, R. Raileanu, M. Lomeli, L. Zettlemoyer, N. Cancedda, and T. Scialom, *Toolformer: Language models can teach themselves to use tools*, arXiv preprint, 2023, DOI 10.48550/arXiv.2302.04761
30. K. Ma, H. Cheng, X. Liu, E. Nyberg, and J. Gao, *Open-domain question answering via chain of reasoning over heterogeneous knowledge*, (Findings of the Association for Computational Linguistics: EMNLP), 2022, pp. 5360–5374.
31. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT*: pre-training of Deep Bidirectional Transformers for Language Understanding, (Proc. NAACL-HLT, Minneapolis, MN, USA), 2019, pp. 4171–4186.
32. T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, *Language models are few-shot learners*, Adv. Neural Info. Process. Syst. **33** (2020), 1877–1901.
33. G. Lample and A. Conneau, *Cross-lingual language model pretraining*, arXiv preprint, 2019, DOI 10.48550/arXiv.1901.07291
34. T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, and K. Toutanova, *Natural questions: a benchmark for question answering research*, Trans. Assoc. Comput. Linguist. **7** (2019), 452–466.
35. Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, *HotpotQA: a dataset for diverse, explainable multi-hop question answering*, (Proc. Conf. Empirical Methods in Natural Language Processing), 2018, pp. 2369–2380, DOI 10.18653/v1/D18-1259.

AUTHOR BIOGRAPHIES

Jihyeon Roh received her B.S. degree in Electronic and Electrical Engineering from Sungkyunkwan University in 2013, and her Ph.D. degree in Electrical Engineering at Korea Advanced Institute of Science and Technology in 2022. Since 2022, she has been a researcher at the Language Intelligence Lab in Electronics and Telecommunications Research Institute. Her primary research interests include neural language models, natural language processing, language generation system, and machine learning for LMs.

Minho Kim received his B.S. degree in Electronics from Korea University, Seoul, Republic of Korea, in 1997, and M.S. and Ph.D. degrees in Information and Communications Engineering from Gwangju Institute of Science and Technology, Gwangju, Republic of Korea, in 1999 and 2006, respectively. Since 2006, he has worked at the Electronics and Telecommunications Research Institute as a principal researcher. His research interests include natural language processing, deep neural network language models, and the reasoning process in artificial intelligence.

Kyoungman Bae received his B.S., M.S., and Ph.D. degree in Computer Engineering from the Department of Computer Engineering, Dong-A University, Busan, Republic of Korea, in 2004, 2006, and 2016, respectively. Since 2016, he has worked at the Language Intelligence Lab in Electronics and Telecommunications Research Institute, Daejeon, Republic of Korea. His main research interests are LLM, natural language processing, explainable AI, and generative AI.

How to cite this article: J. Roh, M. Kim, and K. Bae, *Towards a small language model powered chain-of-reasoning for open-domain question answering*, ETRI Journal **46** (2024), 11–21, DOI 10.4218/etrij.2023-0355