

ChatCoT: Tool-Augmented Chain-of-Thought Reasoning on Chat-based Large Language Models

Zhipeng Chen^{1,3*}, Kun Zhou^{2,3*}, Beichen Zhang^{1,3}, Zheng Gong^{1,3},
Wayne Xin Zhao^{1,3†} and Ji-Rong Wen^{1,2,3}

¹Gaoling School of Artificial Intelligence, Renmin University of China.

²School of Information, Renmin University of China.

³Beijing Key Laboratory of Big Data Management and Analysis Methods.

{zhipeng_chen, gongzheng0109, jrwen}@ruc.edu.cn, francis_kun_zhou@163.com

{zhangbeichen724, batmanfly}@gmail.com

Abstract

Although large language models (LLMs) have achieved excellent performance in a variety of evaluation benchmarks, they still struggle in complex reasoning tasks which require specific knowledge and multi-hop reasoning. To improve the reasoning abilities, we propose **ChatCoT**, a tool-augmented chain-of-thought reasoning framework for chat-based LLMs (e.g., ChatGPT). In ChatCoT, we model the chain-of-thought (CoT) reasoning as multi-turn conversations, to utilize tools in a more natural way through chatting. At each turn, LLMs can either interact with tools or perform the reasoning. Our approach can effectively leverage the multi-turn conversation ability of chat-based LLMs, and integrate the thought chain following and tools manipulation in a unified way. Specially, we initialize the early turns of the conversation by the knowledge about tools, tasks, and reasoning format, and propose an iterative *tool-augmented reasoning* step to perform step-by-step tool-augmented reasoning. The experiment results on two complex reasoning datasets (MATH and HotpotQA) have shown the effectiveness of ChatCoT on complex reasoning tasks, achieving a 7.9% relative improvement over the state-of-the-art baseline. Our code and data are available at: <https://github.com/RUCAIBOX/ChatCoT>.

1 Introduction

Recently, large language models (LLMs) (Zhao et al., 2023a) have shown great potential as general-purpose task solvers in a variety of real-world applications. With excellent few-shot and zero-shot ability, LLMs, such as GPT-4 (OpenAI, 2023) and LLaMA (Touvron et al., 2023), can even outperform full-data supervised-tuned models on many tasks with suitable prompting strategies.

Among these prompting strategies, chain-of-thought (CoT) prompting (Wei et al., 2022; Ko-

jima et al., 2022) has been a prominent approach to eliciting the reasoning abilities of LLMs. It incorporates the intermediate reasoning steps of exemplars into the input prompt, to instruct LLMs to solve a question step by step. Despite the remarkable improvement by CoT prompting, LLMs still have difficulties in solving complex reasoning tasks that involve specific functionalities, such as arithmetic calculation and information retrieval (Lu et al., 2022; Qian et al., 2022). To address this issue, external tools (e.g., calculator, search engine) have been employed to fulfill the basic functionalities (Schick et al., 2023; Paranjape et al., 2023), easing the burden of LLMs. With proper interfaces, LLMs can be guided by prompts to manipulate tools when necessary.

However, as tools are not intrinsically integrated with LLMs, incorporating external tools would have to interrupt the CoT reasoning process of LLMs. Such an issue would become more intractable on complex reasoning tasks that frequently invoke the use of tools. To address it, existing work either relies on LLMs to prearrange the tool use plan for subsequent execution (Zhou et al., 2022; Jiang et al., 2023b), or needs to design formal actions pertaining to specific tasks (Dua et al., 2022; Khattab et al., 2022; Jiang et al., 2023a). Despite the effectiveness, the two types of methods still suffer from potential issues: the former one cannot interact with tools after generating the plan, even seeing obvious mistakes; while the latter one has to frequently switch between reasoning with LLMs and taking actions, hurting the continuity of the CoT reasoning process.

To overcome these disadvantages, we seek a more unified way to integrate CoT reasoning and tool manipulation. As the key idea, we consider tools manipulation by LLMs as the *interaction* between LLMs and tools, in which LLMs send the use requests and tools respond to support specific functions. Further, inspired by the recent progress

* Equal contributions.

† Corresponding author.

of ChatGPT-like LLMs (called *chat-based LLMs*), we model the interaction process between LLMs and tools as a multi-turn conversation, and leverage the excellent chatting capacities for manipulating tools by LLMs. At each turn, the LLM can freely interact with tools when in need, otherwise perform the reasoning by itself. The conversation continues until the final answer is derived by LLMs. In this process, as chat-based LLMs can well understand the multi-turn context, they can follow the thought chain in the whole conversation and naturally invoke the tools accordingly, thus keeping the continuity of the reasoning process.

To this end, in this paper, we propose ChatCoT, a tool-augmented chain-of-thought reasoning strategy for chat-based LLMs. As the major merit, ChatCoT can perform the CoT reasoning across multi-turn conversation, and freely interact with tools at immediate steps. Concretely, we first store the useful knowledge at early turns of the conversation, including tools, tasks, and multi-turn reasoning format, to help LLMs utilize task-specific knowledge to perform reasoning or manipulate tools. Then, we iterate a specially designed *tool-augmented reasoning* step in which LLMs interact with tools, to perform step-by-step tool-augmented reasoning, until obtaining the final answer.

To evaluate the effectiveness, we implement ChatCoT on ChatGPT, and conduct experiments on two complex reasoning benchmarks, *i.e.*, MATH (Hendrycks et al., 2021) and HotpotQA (Yang et al., 2018). Experimental results show that ChatCoT achieves very promising performance on MATH with 7.9% relative improvement in average over the SOTA baselines (*i.e.*, PHP (Zheng et al., 2023)). Besides, our approach can also be integrated with other strategies, *e.g.*, self-consistency, and ChatCoT can achieve better performance by incorporating these strategies.

2 Related Work

Tool-Augmented Large Language Models. With the large-scale parameters and pre-training corpus, large language models (LLMs) (*e.g.*, Flan T5 (Chung et al., 2022), ChatGPT (OpenAI, 2022) and LLaMA (Touvron et al., 2023)) have demonstrated strong zero-shot and few-shot ability in NLP tasks (*e.g.*, language generation, reasoning). However, LLMs have still struggled with complex reasoning tasks requiring task-specific knowledge and multi-step reasoning (*e.g.*, mathemati-

cal problem solving). Previous work (Zhao et al., 2022, 2023b; Luo et al., 2023) has constructed task-specific corpus and utilized continue pre-training and instruction tuning to inject relative knowledge into LLMs and enhance the complex reasoning ability of LLMs. In order to further reduce the mistakes made by LLMs, existing methods have explored to augment LLMs with external tools. They can be roughly categorized into the following two types. The first type of methods (Gao et al., 2023; Parisi et al., 2022; Qiao et al., 2023) train the model parameters to support the utilization of the external tools, where they collect or synthesize the tool-augmented examples to tune the model parameters (Schick et al., 2023; Patil et al., 2023; Hao et al., 2023). Another type of methods (Gao et al., 2022; Yao et al., 2022; Zhang et al., 2023) utilize carefully designed prompts to guide LLMs to use external tools. They focus on devising proper prompts or tools manipulation ways to select and use tools when necessary (Liang et al., 2023; Shen et al., 2023; Yao et al., 2022). In this work, we follow the second type of methods and propose a tool-augmented chain-of-thought reasoning strategy that can better solve complex reasoning tasks.

Chain-of-Thought Reasoning. To further enhance the reasoning capacity of LLMs, Chain-of-Thought (CoT) prompting strategy (Wei et al., 2022; Kojima et al., 2022) has been proposed to guide LLMs to generate intermediate reasoning steps which can boost the performance of LLMs. Through special instructions (*e.g.*, “Let us think step by step”) and in-context exemplars with detailed intermediate reasoning steps, LLMs can perform step-by-step reasoning to reach the final answer. Based on CoT, recent work has also proposed several methods to further improve the performance, including problem decomposition (Zhou et al., 2022; Dua et al., 2022), appropriate exemplars selection (Ye et al., 2022; Shi et al., 2023), results post-processing (Wang et al., 2022; Madaan et al., 2023; Zheng et al., 2023), and changing the reasoning format (Yao et al., 2023; Wu et al., 2023). However, as the generation process of CoT is one-pass, the utilization of tools in intermediate steps would have to interpret it, hurting the continuity of the generation process. In this work, we propose a unified way to integrate CoT reasoning and tool manipulation, which utilizes the excellent multi-turn chatting capacity of LLMs to perform CoT reasoning across multi-turn conversations.

3 Preliminary

In this section, we present the task setting, then introduce the Chain-of-Thought prompting strategy and the tool manipulation in reasoning tasks.

Task Setting. In this work, we focus on improving the reasoning ability of large language models (LLMs) on complex tasks, *e.g.*, solving mathematics competition problems. Unlike tasks that can be solved by humans via straightforward skills or tools, complex tasks require advanced knowledge (*e.g.*, mathematical theorem) and multi-step reasoning to reach the answer. Typically, a complex problem includes three types of texts, namely problem statement, solution text, and answer key, denoted as Q , S and A , respectively. The problem statement Q introduces the background and description of a complex problem, and the solution text illustrates the detailed solving process to obtain the answer key. All of them are composed of a sequence of tokens, where each token is either a text word or a mathematical symbol. Formally, given the problem statement Q , we aim to utilize LLMs to perform multi-step reasoning, to finally generate its accurate answer A .

Chain-of-Thought Prompting. To elicit the powerful reasoning ability of LLMs for complex tasks, Chain-of-Thought (CoT) prompt strategy (Wei et al., 2022) has been widely used to guide LLMs for performing step-by-step reasoning. Generally, the CoT prompt consists of few exemplars whose a series of intermediate reasoning steps $\{I_1, \dots, I_n\}$ are also involved. Each exemplar can be denoted as $E = \langle Q, \{I_1, \dots, I_n\}, A \rangle$. Formally, given the question and few exemplars, a CoT prompt is composed by integrating them as a long input of the LLM, which can prompt the LLM to generate a similar thought chain that leads to the final answer.

Tool Manipulation. Previous work has revealed that LLMs are struggling with basic functionality (*e.g.*, arithmetical calculation (Schick et al., 2023)), which can be solved by using specific external tools (*e.g.*, calculator), denoted as $\{T_1, \dots, T_n\}$. To manipulate tools, existing work mostly relies on writing a detailed prompt for describing how to use available tools for the LLM, then incorporates it to guide the selection of useful tools and generate the tool arguments, and finally calls the tool API to obtain the result. Following this way, in this work, we focus on three useful tools that have been widely

used by humans to solve complex problems:

- *Calculator:* Given a mathematical expression, the calculator can compute the value of it or simplify it according to arithmetic rules (*e.g.*, combining like terms and reduction of fractions).

- *Equation Solver:* Given the equations system and unknown variables, the equation solver can automatically calculate the value of the contained unknown variables through relative algorithms.

- *Retriever:* Given a query, the retriever aims to extract the most relevant information (*e.g.*, documents) from a number of candidates. According to the types of the retrieved corpus, it can be implemented by specialized models, *e.g.*, dense retrieval model.

We implement the first two tools by using different functions of *SymPy* (Meurer et al., 2017), a Python library for mathematical symbolic calculation. For the retriever, we adopt SimCSE (Gao et al., 2021), a sentence embedding model to measure the text semantic similarity. Note that when the input expression or equation is ill-formed or unsolved, the above tools would return an error.

4 Approach

In this section, we present our proposed ChatCoT, a new chain-of-thought (CoT) prompting framework based on multi-turn conversations, for improving chat-based LLMs on complex reasoning tasks with tools. The overall illustration of our proposed ChatCoT is shown in Figure 1.

4.1 Overview

For complex tasks (*e.g.*, advanced mathematical problems), LLMs require to frequently manipulate the tools when in need, to fulfill the intractable intermediate issues. However, as tools are not intrinsically integrated with LLMs, previous work mostly relies on the LLM to generate the plan of manipulating tools and then execution (Gao et al., 2022; Lu et al., 2023), or immediately call tools by stopping the continuous generation of LLMs (Yao et al., 2022). The above both ways are not suitable for the frequent interactions between LLMs and tools, due to the error accumulation in planning and frequent interruptions in LLM generation.

In our approach, we decompose the chain-of-thought reasoning process of LLMs into a multi-round conversation. In each turn, LLMs just require to concentrate on manipulating tools or accomplishing reasoning in the current step, and the whole

Vanilla CoT

Problem: A function f has the property that $f(3x - 1) = x^2 + x + 1$ for all real numbers x . What is $f(5)$? [Prompting] Context

Solution: Let's think step by step. Let $u = 3x - 1$. Then $x = \frac{u+1}{3}$, and $f(u) = \left(\frac{u+1}{3}\right)^2 + \frac{u+1}{3} + 1 = \frac{u^2+2u+1}{9} + \frac{u+1}{3} + 1 = \frac{u^2+2u+1}{9} + \frac{u^2+5u+13}{9}$. In particular, $f(5) = \frac{5^2+5 \times 5+13}{9} = 7$.

Problem: Q : Solve for e , given that $2d$ is 8 less than $17e$, and $2e$ is 9 less than d . [Reasoning] One-time Inference

Solution: Let's think step by step. I_1 : We begin with a system of two equations $2d = 17e - 8$, $2e = d - 9$. I_2 : Since the second equation can also be rewritten as $d = 2e + 9$, we can plug this expression for d back into the first equation and solve for e . I_3 : $2d = 17e - 8 \rightarrow 2(2e + 9) = 17e - 8 \rightarrow 4e + 18 = 17e - 8 \rightarrow -13e = -26 \rightarrow e = A$: 2.

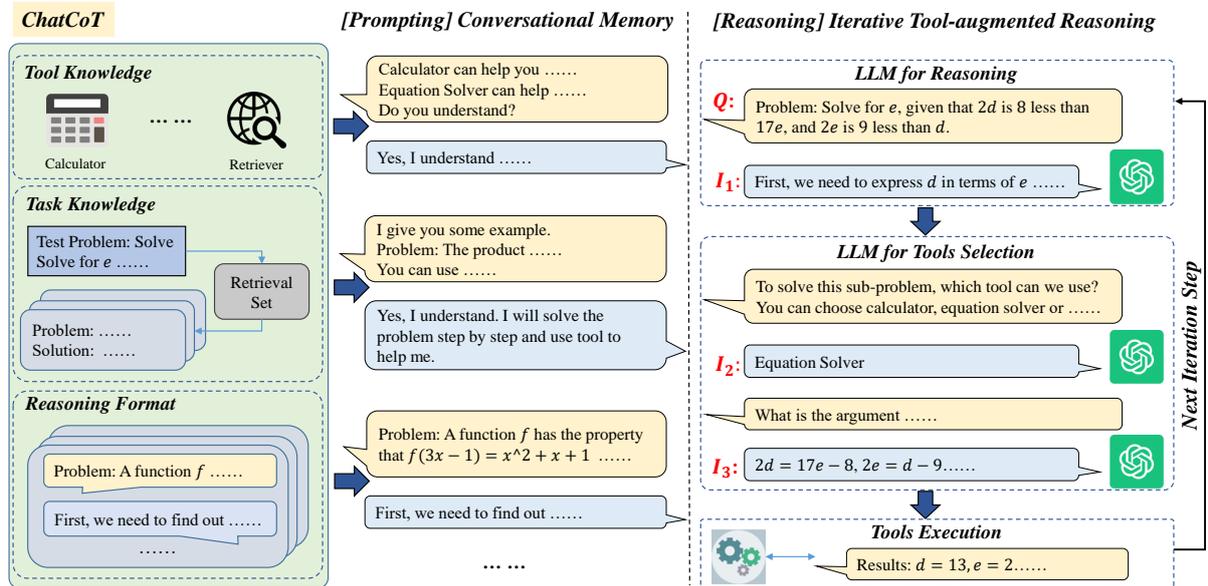


Figure 1: The comparison of vanilla CoT and ChatCoT, illustrated for a mathematical problem. For vanilla CoT, the content underlined are generated by LLMs. For ChatCoT, the conversational knowledge memory is initialized to provide tools, task and reasoning format knowledge. Then, the tool-augmented reasoning step is iterated multiple times to perform step-by-step reasoning, until obtaining the answer.

reasoning process would keep on pushing without premature planning and sudden interruption. In this way, the whole reasoning process would be converted to a conversation between LLMs and an agent, which follows pre-defined rules to guide LLMs and manipulate the tool. By designing proper chatting strategies, the agent would automatically elicit LLMs to perform reasoning and select a tool, or invoke the tool for execution.

In our approach, we first initialize the multi-turn conversation by feeding chat-based LLMs with the background knowledge, *i.e.*, the description of tools, relevant task exemplars, and the demonstration of decomposed chain-of-thought in chat, which are the *conversational knowledge memory* for supporting the following reasoning. Then, we propose the *tool-augmented reasoning* procedure that leverages LLMs to perform reasoning with tools in the current step and iterate it to fulfill all sub-tasks in the whole reasoning process, until reaching the answer. We introduce the details of the two components in the following.

4.2 Initializing Conversational Knowledge Memory

To guide chat-based LLMs to follow our proposed ChatCoT using external tools, it is essential to design proper prompts in context. In our approach, as we reformulate the chain-of-thought reasoning into a decomposed multi-turn conversation, we can also feed the essential prompts into LLMs at early turns as the context, to initialize the conversation background knowledge. It can be seen as the in-context *knowledge memory* in the format of dialogue that stores useful knowledge for helping chat-based LLMs manipulate tools or perform reasoning. Here, we consider three types of knowledge about tools, task, and multi-turn reasoning format, respectively. The details of prompts are in Appendix A.

Tools Knowledge. As LLMs have never seen tools during pre-training, for each tool in Section 3, we hand-craft its description in the following pattern: “[T] can help you [Y]”, where [T] is the tool name and [Y] shows its detailed functionality. Then, we merge all the descriptions and design the input

prompt to tell LLMs about the knowledge of all tools. We also hand-craft the expected response of the LLM. It will be also fed into the LLM, to indicate the LLM that it has accepted our prompt and should follow it.

Retrieval-Augmented Task Knowledge. Since LLMs can learn the task knowledge from in-context exemplars, we leverage a retriever to select the most relevant instance from the training dataset, to provide more useful knowledge for the given question. Concretely, we train SimCSE (Gao et al., 2021), a sentence embedding method that can measure the semantic similarity of texts, via the unsupervised training strategy on the training set. Then, we leverage it to retrieve the top- k most semantically similar exemplars, and concatenate their problem statement Q and solution S to compose the input prompt. Similarly, we also feed it with our expected response into the LLM.

Multi-turn Reasoning Format. To elicit LLMs following multi-turn reasoning format, we manually annotate the whole multi-round dialogue I_1, \dots, I_n of randomly sampled five questions from the training set, to create the exemplars. Then, we feed the dialogues of all the exemplars into the chat-based LLM round by round, as the context to guide LLMs to follow it for performing reasoning.

Summary. The above three types of multi-turn utterances are pre-defined with corresponding contents and formats, which compose the *conversational knowledge memory* of our approach. It would be leveraged to initialize the conversational context, and support the following step-by-step reasoning for answering the question.

4.3 Iterative Tool-augmented Reasoning

Based on the above conversational knowledge memory, we iterate the tool-augmented reasoning step to perform step-by-step tool-augmented reasoning, until finally obtain the answer.

4.3.1 Tool-augmented Reasoning Step

The tool-augmented reasoning step can be iterated in multiple times. In each iteration, based on the current results, we first leverage LLMs to perform reasoning, then select the proper tool by LLMs, and finally execute the selected tool to obtain the intermediate result in the current step.

LLM for Reasoning. Guided by the exemplars in the conversation history, LLMs are able to decom-

pose the whole reasoning process into multi-turn chat. Specially, LLMs would be elicited by the contextual exemplars to directly perform reasoning in natural language based on the current result, without specialized prompts or instructions. Consequently, LLMs can rely on the retrieval-augmented task knowledge in context, to generate the natural language solution till the point that needs the functionality of tools.

LLM for Tools Selection. After reasoning, we utilize the LLM to select a useful tool (e.g., calculator), which will be employed to provide the required functionality for the LLM. Here, the input prompt of the LLM is “*To solve this sub-problem, which tool can we use?*” After feeding it into the LLM, if the LLM requires to utilize tools, it will select a suitable one, and then we further ask the LLM to formulate the input arguments of the tool, e.g., mathematical expression. Otherwise, it will answer “*Do not use tool*”, and the LLM will continue to perform reasoning.

Tools Execution. Given the selected tool and formulated arguments by LLMs, we can execute the tool with the arguments to obtain the result in the current iteration. Here, we also consider that the results from the tool may be not satisfied by the LLM, e.g., irrelevant retrieved documents. In this case, we can also add several feedback rounds where the LLM judges if the result is useful or expected, and then reuse the tool to acquire a new result.

4.3.2 Iteration for Step-by-Step Reasoning

We iterate the above step based on the in-context conversation knowledge memory, to perform step-by-step reasoning on the given question Q . We start the whole iteration process using the following prompt: “*You should solve the problem step by step and you should follow the react in the history [Q]*”. Then, after reaching the answer key, the iteration process will be stopped by LLMs. In practice, we find that chat-based LLMs are prone to continue chatting although the answer key has appeared in the reasoning process. Thus, we set the maximum chat turns, and devise the following prompt to force LLMs to stop reasoning and conclude the answer: “*Base on the context, what is the answer?*”.

As our proposed approach only decomposes the one-pass chain-of-thought reasoning into multi-turn chat and adds the utilization of tools, it is agnostic to the task types and tools implementation. Therefore, it is a general framework that can

Dataset	Category	Train	Dev/Test
MATH	Algebra	1744	1187
	CP	771	474
	Precalculus	746	546
	Prealgebra	1205	871
	Geometry	870	479
	IA	1295	903
	NT	869	540
HotpotQA	Distractor	90477	7405

Table 1: Statistics of the two complex reasoning datasets. CP, IA, and NT denote *Counting and Probability*, *Intermediate Algebra*, and *Number Theory*, respectively.

be applied to a variety of complex reasoning tasks that require suitable tools. Besides, our approach also supports the recently proposed improvement strategies based on the chain-of-thought method, *e.g.*, self-consistency (Wang et al., 2022). We conduct corresponding experiments in Section 5.3 to validate it.

5 Experiment

In this section, we conduct experiments to evaluate the effectiveness of ChatCoT. The implementation details can be found in Appendix B.

5.1 Experimental settings

Datasets. We consider two complex reasoning datasets for evaluation, *i.e.*, MATH (Hendrycks et al., 2021) and HotpotQA (Yang et al., 2018). The details of these two datasets are shown in Table 1. We adopt accuracy as the evaluation metric.

- *MATH* is composed of challenging competition mathematical problems which require advanced mathematical knowledge. It is divided into seven categories, *i.e.*, Algebra, Counting and Probability, Precalculus, Prealgebra, Geometry, Intermediate Algebra, and Number Theory. We adopt the calculator and an equation solver as external tools to help LLMs.

- *HotpotQA* is a multi-hop question answering dataset, where each question is associated with a collection of paragraph candidates containing several golden contents which are useful for reasoning. We use the development set under the distractor setting of HotpotQA for evaluation, where the annotation of golden paragraphs is not aware to LLMs. We employ the retriever as the external tool.

Baselines. We mainly compare our approach with

the following prompting strategies based on ChatGPT (OpenAI, 2022):

- *Chain-of-Thought (CoT)* (Wei et al., 2022) is a prominent method to boost the performance of LLMs in reasoning tasks. In CoT, LLMs are prompted to generate the intermediate reasoning path and reasoning step by step to reach the final answer. Previous work has shown that the utilization of external tools and similar exemplars improves the performance of CoT. Therefore, we implement external tools to help LLMs reason and retrieve to help LLMs select exemplars, which are named *CoT w/ Tool*, and *CoT w/ Retri*, respectively.

- *Learning-to-Program (LP)* (Guo et al., 2023) guides LLMs to program in natural language by learning solutions in the training set, and elicits LLMs to solve tasks following the program.

- *Progressive-Hint Prompting (PHP)* (Zheng et al., 2023) proposes to iteratively refine the solution based on the answer hints from previous trials. The iterative method achieves SOTA on MATH.

To provide a more complete evaluation, we also report the performance of various LLM backbones with the vanilla CoT prompting, including PaLM (Chowdhery et al., 2022), PaLM 2 (Google, 2023), Minerva (Lewkowycz et al., 2022), Galactica (Taylor et al., 2022), LLaMA (Touvron et al., 2023) and GPT-3 (Brown et al., 2020).

5.2 Main Results

We present the evaluation results of our approach on MATH and HotpotQA datasets in Table 2 and Table 3 respectively.

First, for the comparison of backbones for CoT prompting, ChatGPT achieves the best performance, demonstrating its outstanding mathematical reasoning ability. Our method elicits the reasoning process by leveraging the strong multi-turn dialogue ability of ChatGPT, thus leading to a better release of the reasoning ability from ChatGPT.

Second, retrieval-augmented methods (*e.g.*, ChatCoT, CoT w/ Retri) outperform other baselines. The reason is that retrieved exemplars may contain more relevant knowledge and reasoning steps that are beneficial to solve the given problem. On Geometry tasks of MATH, CoT w/ Retri achieves the largest improvement over vanilla CoT than other sub-tasks. Another possible reason is that ChatGPT is more unfamiliar to the knowledge and symbol of geometry than others. Without similar exemplars, it is difficult for LLMs to well understand them.

Models	Prompt Strategy	MATH							
		Algebra	CP	PC	PA	Geometry	IA	NT	Avg.
GPT-3	CoT	6.0	4.7	4.0	7.7	3.1	4.4	4.4	5.2
PaLM	CoT	9.7	8.4	4.4	19.2	7.3	3.5	6.0	8.8
LLaMA	CoT	-	-	-	-	-	-	-	10.6
Galactica	CoT	29.0	13.9	12.8	27.2	12.3	9.6	11.7	20.4
Minerva	CoT	51.3	28.0	18.0	55.0	26.8	13.7	21.2	33.6
PaLM 2	CoT	-	-	-	-	-	-	-	34.3
ChatGPT	CoT	48.1	31.4	<u>21.1</u>	56.6	22.3	18.3	29.1	35.1
	CoT w/ Tool	35.9	22.6	9.3	40.5	13.6	9.4	19.4	23.8
	CoT w/ Retri	<u>52.7</u>	32.7	18.9	<u>58.4</u>	<u>29.2</u>	19.9	31.7	<u>37.7</u>
	LP	49.6	30.2	16.3	52.3	22.5	16.9	29.8	34.0
	PHP	51.1	<u>33.7</u>	16.1	57.7	25.4	17.1	35.1	36.5
	ChatCoT	56.1	34.2	23.8	59.2	29.9	<u>19.5</u>	<u>32.6</u>	39.4

Table 2: Experimental results on MATH dataset. PC and PA denote *Precalculus* and *Prealgebra*, respectively. Avg. is the average value of all categories. The best are denoted in bold and the second-best are underlined.

Methods	HotpotQA
CoT	38.0
CoT w/ Tool	31.4
ChatCoT w/o Feedback	<u>53.8</u>
ChatCoT	59.2

Table 3: The results on HotpotQA. We report the results of the development set under the distractor setting.

Third, given the results of CoT and CoT w/ Tool on MATH and HotpotQA, we can find that directly utilizing external tools during reasoning is not a suitable way, which may hurt the performance of LLMs. The reason may be that injecting tool usage into the CoT reasoning process will hurt the continuity of reasoning.

Finally, ChatCoT achieves state-of-the-art performance on MATH dataset based on ChatGPT and outperforms other baselines on HotpotQA. Compared with the previous SOTA method PHP, ChatCoT outperforms six of seven sub-tasks on MATH dataset and achieves 7.9% relative improvement on average accuracy over the PHP method. The experiment results have verified the effectiveness of ChatCoT on complex reasoning tasks. By leveraging conversational knowledge memory and multi-round dialogue to reasoning, ChatCoT has the advantage to utilize plug-and-play tools. Moreover, on the Number Theory tasks of MATH, we can find that PHP achieves the best performance. The reason may be that there are fewer equations that need to be computed or simplified. Thus, the advantage

Methods			MATH		
TK	RATK	MRF	PC	Geo	NT
✓	✓	✓	23.8	29.9	32.6
✗	✓	✓	23.3	29.2	30.6
✓	✗	✓	20.0	27.4	31.0
✓	✓	✗	21.6	24.2	32.2
✗	✗	✓	16.7	21.1	29.3

Table 4: The results of ablation study. TK, RATK, and MRF denote if using tool knowledge, retrieval-augmented task knowledge, and multi-turn reasoning format at early turns of the conversation, respectively. Geo is the abbreviation of *Geometry*.

of the utilization of tools becomes less obvious.

5.3 Detailed Analysis

In order to further verify the effectiveness of each component in ChatCoT, we conduct experiments about ablation, adaption, tools utilization and expense. We present the case study in Appendix C.1.

Ablation Study. In the ablation study, we evaluate the effectiveness of conversational memory, including tool knowledge memory, retrieval-augmented knowledge memory, and multi-turn reasoning format memory. As shown in Table 4, removing any type of conversational memory will reduce the performance of ChatCoT, which indicates the effectiveness of these memories in complex reasoning. In particular, removing retrieval-augmented knowledge memory or multi-turn reasoning format memory will lead to a large drop, which shows

Methods	CP	NT
CoT + SC	35.2 _{+3.8%}	34.4 _{+5.3%}
ChatCoT + SC	40.1 _{+5.9%}	38.3 _{+5.7%}

Table 5: The evaluated accuracy of combining our approach with self-consistency. SC denotes self-consistency. We also report the absolute improvement compared with vanilla methods on subscripts.

Methods	Frequency	Success
CoT w/ Tool	3.0%	85.7%
ChatCoT w/o TK	56.0%	93.0%
ChatCoT w/o MRF	10.0%	64.2%
ChatCoT	70.0%	92.0%

Table 6: Frequency and success rate of tool manipulation on Number Theory task of MATH. TK, MRF denote tool knowledge, multi-turn reasoning format at early turns of the conversation respectively.

that mathematical knowledge and reasoning format knowledge is important for LLMs in reasoning tasks, while LLMs can learn the usage of external tools from exemplars without descriptions.

Combination with Improvement Strategies.

ChatCoT is a general method to enhance the ability of tool manipulation of LLMs. It can be integrated with improvement strategies and further boost the performance of LLMs on reasoning tasks. To evaluate the applicability of ChatCoT to improvement strategies designed for CoT, we compare ChatCoT with CoT on two subtasks of MATH, where both methods are augmented with self-consistency (Wang et al., 2022), a representative improvement strategy for CoT prompting. Concretely, we sample 5 outputs for majority voting in self-consistency. As shown in Table 5, self-consistency brings improvement in both CoT and ChatCoT. In particular, the absolute improvement of ChatCoT is slightly higher than CoT, showing that ChatCoT can adapt to self-consistency well. The reason is that, with the decomposing of reasoning procedures, the intermediate steps of ChatCoT are more confident, and small mistakes will be corrected easily. Moreover, we construct the case study about the combination with ChatCoT and Self-Refine (Madaan et al., 2023) in Appendix C.2.

Tools Utilization Analysis. As we mentioned above, in complex reasoning tasks, infrequently or incorrectly utilizing external tools might lead to wrong answers. Thus, we conduct the experiment

Methods	Generated Tokens
CoT	224.6
CoT w/ Tool	296.2
Self-Consistency	1017.4
ChatCoT	355.2

Table 7: The comparison of the number of generated tokens from LLMs among different prompt strategies.

about whether LLMs can frequently or correctly leverage based on different methods. Table 6 expresses the performance of tools utilization in the Number Theory task of MATH of baseline and our approach. ‘‘Frequency’’ denotes the ratio of problems where LLMs correctly leverage tools. ‘‘Success’’ denotes the rate of LLMs utilizing tools successfully among all the times of invoking tools. We can observe that ChatCoT achieves a balance of frequency and ratio of success. Tool knowledge provides the function of tools for LLMs and improves the frequency that LLMs utilize the tools. LLMs can learn how to leverage external tools through the multi-turn reasoning format and boost the ratio of successful utilization of tools. Without any of them, the frequency and ratio of success will drop which might not be conducive to reasoning.

Number of Generated Tokens Analysis. Despite guiding LLMs to reason through multi-turn dialogue, the computation expense of ChatCoT is not significantly larger than the CoT method. In Table 7, we present the average number of generated tokens from LLMs in several methods on MATH dataset, which reflects the computation expense to a certain degree. We can observe that ChatCoT is on the same order of magnitude with other baselines (e.g., CoT and CoT w/ Tool). Therefore, ChatCoT does not bring significant expenses compared to existing prompting methods.

6 Conclusion

In this paper, we have proposed ChatCoT, a new framework to manipulate the tools for the CoT reasoning. It naturally integrates the reasoning process and manipulating tools through a form of multi-turn conversations. At each turn, LLMs can either interact with tools or perform the reasoning by itself. Our approach can effectively leverage the multi-turn conversation ability of chat-based LLMs. Experimental results on two complex reasoning tasks including MATH and HotpotQA have verified the

effectiveness of ChatCoT.

Currently, our experiments are mainly conducted on mathematical reasoning tasks, and we will test the effectiveness of the proposed approach to more types of reasoning tasks. Besides, we will also consider extending the number of available tools for solving different tasks.

Limitations

In this section, we discuss the limitations of our work. First, we do not utilize GPT-4 in our experiment or evaluate the performance of GPT-4 in the ChatCoT framework. That is because our application for GPT-4 has not been accepted. Second, ChatCoT is designed for chat-based LLMs and it is hardly compatible with other LLMs. However, most LLMs support multi-turn conversation currently and they perform well on reasoning tasks. Besides, although LLMs have achieved strong ability in reasoning tasks, the requirement of computation expense and GPU resource is higher than other pre-trained language models which have millions of parameters. The utilization of LLMs will produce more carbon dioxide and pollute the environment.

Acknowledgement

This work was partially supported by National Natural Science Foundation of China under Grant No. 62222215, Beijing Natural Science Foundation under Grant No. L233008 and 4222027, and Beijing Outstanding Young Scientist Program under Grant No. BJJWZYJH012019100020098. And this work is also partially supported by the Outstanding Innovative Talents Cultivation Funded Programs 2021 of Renmin University of China. Xin Zhao is the corresponding author.

References

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. In *Advances in Neural Information Processing Systems* 33:

Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayanan Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. 2022. Palm: Scaling language modeling with pathways. *CoRR*, abs/2204.02311.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 1251–1265.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2022. PAL: program-aided language models. *CoRR*, abs/2211.10435.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

- Google. 2023. Palm 2 technical report. *Google*.
- Yiduo Guo, Yaobo Liang, Chenfei Wu, Wenshan Wu, Dongyan Zhao, and Nan Duan. 2023. Learning to program with natural language. *CoRR*, abs/2304.10464.
- Shibo Hao, Tianyang Liu, Zhen Wang, and Zhiting Hu. 2023. Toolkengpt: Augmenting frozen language models with massive tools via tool embeddings. *CoRR*, abs/2305.11554.
- Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.
- Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Wayne Xin Zhao, and Ji-Rong Wen. 2023a. Structgpt: A general framework for large language model to reason over structured data. *arXiv preprint arXiv:2305.09645*.
- Xue Jiang, Yihong Dong, Lecheng Wang, Qiwei Shang, and Ge Li. 2023b. Self-planning code generation with large language model. *CoRR*, abs/2303.06689.
- Omar Khattab, Keshav Santhanam, Xiang Lisa Li, David Hall, Percy Liang, Christopher Potts, and Matei Zaharia. 2022. Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP. *CoRR*, abs/2212.14024.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.
- Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay V. Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models. In *NeurIPS*.
- Yaobo Liang, Chenfei Wu, Ting Song, Wenshan Wu, Yan Xia, Yu Liu, Yang Ou, Shuai Lu, Lei Ji, Shaoguang Mao, Yun Wang, Linjun Shou, Ming Gong, and Nan Duan. 2023. Taskmatrix.ai: Completing tasks by connecting foundation models with millions of apis. *CoRR*, abs/2303.16434.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.
- Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. Chameleon: Plug-and-play compositional reasoning with large language models. *arXiv preprint arXiv:2304.09842*.
- Pan Lu, Liang Qiu, Wenhao Yu, Sean Welleck, and Kai-Wei Chang. 2022. A survey of deep learning for mathematical reasoning. *CoRR*, abs/2212.10535.
- Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *CoRR*, abs/2308.09583.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. *CoRR*, abs/2303.17651.
- Aaron Meurer, Christopher P. Smith, Mateusz Pa-procki, Ondrej Certík, Sergey B. Kirpichev, Matthew Rocklin, Amit Kumar, Sergiu Ivanov, Jason Keith Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Stepán Roucka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony M. Scopatz. 2017. Sympy: symbolic computing in python. *PeerJ Comput. Sci.*, 3:e103.
- OpenAI. 2022. Introducing chatgpt. *OpenAI Blog*.
- OpenAI. 2023. Gpt-4 technical report. *OpenAI*.
- Bhargavi Paranjape, Scott Lundberg, Sameer Singh, Hannaneh Hajishirzi, Luke Zettlemoyer, and Marco Tulio Ribeiro. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.
- Aaron Parisi, Yao Zhao, and Noah Fiedel. 2022. TALM: tool augmented language models. *CoRR*, abs/2205.12255.
- Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2023. Gorilla: Large language model connected with massive apis. *CoRR*, abs/2305.15334.
- Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. 2022. Limitations of language models in arithmetic and symbolic induction. *CoRR*, abs/2208.05051.
- Shuofei Qiao, Honghao Gui, Huajun Chen, and Ningyu Zhang. 2023. Making language models better tool learners with execution feedback. *CoRR*, abs/2305.13068.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *CoRR*, abs/2302.04761.

- Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. 2023. Hugging-gpt: Solving ai tasks with chatgpt and its friends in hugging face.
- Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael Schärli, and Denny Zhou. 2023. Large language models can be easily distracted by irrelevant context. *CoRR*, abs/2302.00093.
- Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science. *CoRR*, abs/2211.09085.
- Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *CoRR*.
- Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *CoRR*, abs/2203.11171.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.
- Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. 2023. An empirical study on challenging math problem solving with GPT-4. *CoRR*, abs/2306.01337.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2369–2380.
- Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *CoRR*, abs/2305.10601.
- Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *CoRR*, abs/2210.03629.
- Xi Ye, Srinivasan Iyer, Asli Celikyilmaz, Ves Stoyanov, Greg Durrett, and Ramakanth Pasunuru. 2022. Complementary explanations for effective in-context learning. *CoRR*, abs/2211.13892.
- Beichen Zhang, Kun Zhou, Xilin Wei, Wayne Xin Zhao, Jing Sha, Shijin Wang, and Ji-Rong Wen. 2023. Evaluating and improving tool-augmented computation-intensive math reasoning. *CoRR*, abs/2306.02408.
- Wayne Xin Zhao, Kun Zhou, Zheng Gong, Beichen Zhang, Yuanhang Zhou, Jing Sha, Zhigang Chen, Shijin Wang, Cong Liu, and Ji-Rong Wen. 2022. Jiuzhang: A chinese pre-trained language model for mathematical problem understanding. In *KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022*, pages 4571–4581.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023a. A survey of large language models. *CoRR*, abs/2303.18223.
- Xin Zhao, Kun Zhou, Beichen Zhang, Zheng Gong, Zhipeng Chen, Yuanhang Zhou, Ji-Rong Wen, Jing Sha, Shijin Wang, Cong Liu, and Guoping Hu. 2023b. Jiuzhang 2.0: A unified chinese pre-trained language model for multi-task mathematical problem solving. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*.
- Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. 2023. Progressive-hint prompting improves reasoning in large language models. *CoRR*, abs/2304.09797.
- Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed H. Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *CoRR*, abs/2205.10625.

A Details of Conversation Memory

In this part, we present the details of the prompt in conversation Memory.

Tool Knowledge. The two turns of utterances are:

User: “You can use tool to help you solve the problem and I give you the instruction of tools usage. $[T_1]$ can help you $[Y_1]$... Do you understand?”

LLM: “Yes, I understand. I will use tool to help me solve the problem.”.

Retrieval-Augmented Task Knowledge. The two-turn utterances are:

User: “I give you some example. Problem: $[Q_1]$ Solution: $[S_1]$... You can use the knowledge and theory in these problem. Do you understand?”

LLM: “Yes, I understand. I will solve the problem step by step and use tool to help me.”.

Multi-turn Reasoning Format. The multi-turn utterances are based on the following pattern:

User: “Problem: $[Q'_1]$ Let’s think step by step and use knowledge in similar problem to solve this problem.”

LLM: “ $[I_1]$ ”.

...

LLM: “ $[I_n]$ ”.

B Implementation Details

During the evaluation, we utilize ChatGPT (gpt-3.5-turbo) (OpenAI, 2022) as our backbone model, and fine-tune RoBERTa (Liu et al., 2019) following SimCSE (Gao et al., 2021) on the training sets of MATH and HotpotQA separately as the retriever in corresponding tasks.

For MATH, we leverage 5-shot setting. The exemplars of CoT and CoT w/ Tool are randomly sampled, while exemplars of CoT w/ Retri are retrieved top-5 similar problems by the retriever. For ChatCoT, 2 retrieval exemplars and 3 annotated exemplars will be adopted. For HotpotQA, we leverage 4-shot setting which is similar to MATH, due to the length limitation of input. For the CoT method, we retrieve the top-3 relevant paragraphs from the paragraph collection as evidence of the given question. In ChatCoT, as the retrieved paragraphs might be not useful for LLMs, LLMs can send feedback to the retriever to show other results at most 5 times.

C Case Study

C.1 Framework of ChatCoT

In order to better present the reasoning process of ChatCoT, we conduct the case study of two problems in MATH dataset, which is shown in Figure 2.

The beginning prompt contains knowledge of tools, tasks, and reasoning format. In the tool knowledge, we introduce the usage and function of external tools. For task knowledge, we retrieve similar problems and corresponding solutions from the training set as retrieval exemplars through semantics similarity, which might contain relevant knowledge. The reasoning format is used to guide LLMs to reason through multi-round dialogue. The beginning prompt and corresponding response are pre-set to enhance the sensitivity of LLMs to instructions.

Instructed by the beginning prompt, LLMs iterate the tool-augmented reasoning step for step-by-step reasoning in ChatCoT. We illustrate the flow of ChatCoT with the example on the left in Figure 2. Given the math problem, LLMs first reason out that d should be expressed in terms of e . Then, LLMs communicate with the agent to select the appropriate external tool *Equation Solver* and specify the corresponding arguments to invoke the tool, including *the unknown symbols* and *the system of equations*. Based on the selection, The assistant executes the tool invocation and returns *the solution of the system of equations* to LLMs. After execution, LLMs can perform the next reasoning step based on the given result. The iterative tool-augmented reasoning process can be iterated several times until LLMs find out the answer. In this case, LLMs find that the original problem has been solved, thus obtaining the final answer and ending the reasoning process.

C.2 Combination with Self-Refine

To better show how LLMs correct mistakes through ChatCoT, we present a case study of the combination of ChatCoT and Self-Refine in the following:

User: “Problem: What is the remainder when $13^{13} + 5$ is divided by 6?

Let’s think step by step”

LLM: First, we need to find out the value of $13^{13} + 5$ using the properties of exponents.

.....

LLM: “Second, we need to find out the remainder when 302875106592258 is divided by 6.”

User: “To solve this sub-problem, which tool can we use?”

LLM: “Calculator”

User: “Give me the equation to calculate”

LLM: “302875106592258 mod 6”

User: “Results: 302875106592258 mod 6 = 1.81725063955355e + 15 * mod.

Continue reasoning”

LLM: “Finally, we know that the remainder when $13^{13} + 5$ is divided by 6 is the same as the remainder when 302875106592258 is divided by 6, which is $\boxed{2}$ ”

User: “The solution above might some mistake, you should check the solution and get the final answer.”

LLM: “Apologies for the mistake. Let’s correct the solution.

To find the remainder when $13^{13} + 5$ is divided by 6, we can simplify the expression 13^{13} first.

Using the property of remainders, we can observe that $13 \equiv 1 \pmod{6}$. Therefore, $13^{13} \equiv 1^{13} \equiv 1 \pmod{6}$.

Now, we can rewrite the expression as $13^{13} + 5 \equiv 1 + 5 \equiv 6 \equiv \boxed{0} \pmod{6}$.

Hence, the remainder when $13^{13} + 5$ is divided by 6 is 0.”

In this example, we can observe that LLMs try to calculate the exact value of $13^{13} + 5$ at first, and then get the remainder. However, the calculator does not understand the expression “302875106592258 mod 6” and returns the wrong result. Through refining, LLM changes the solution and gets the correct answer.

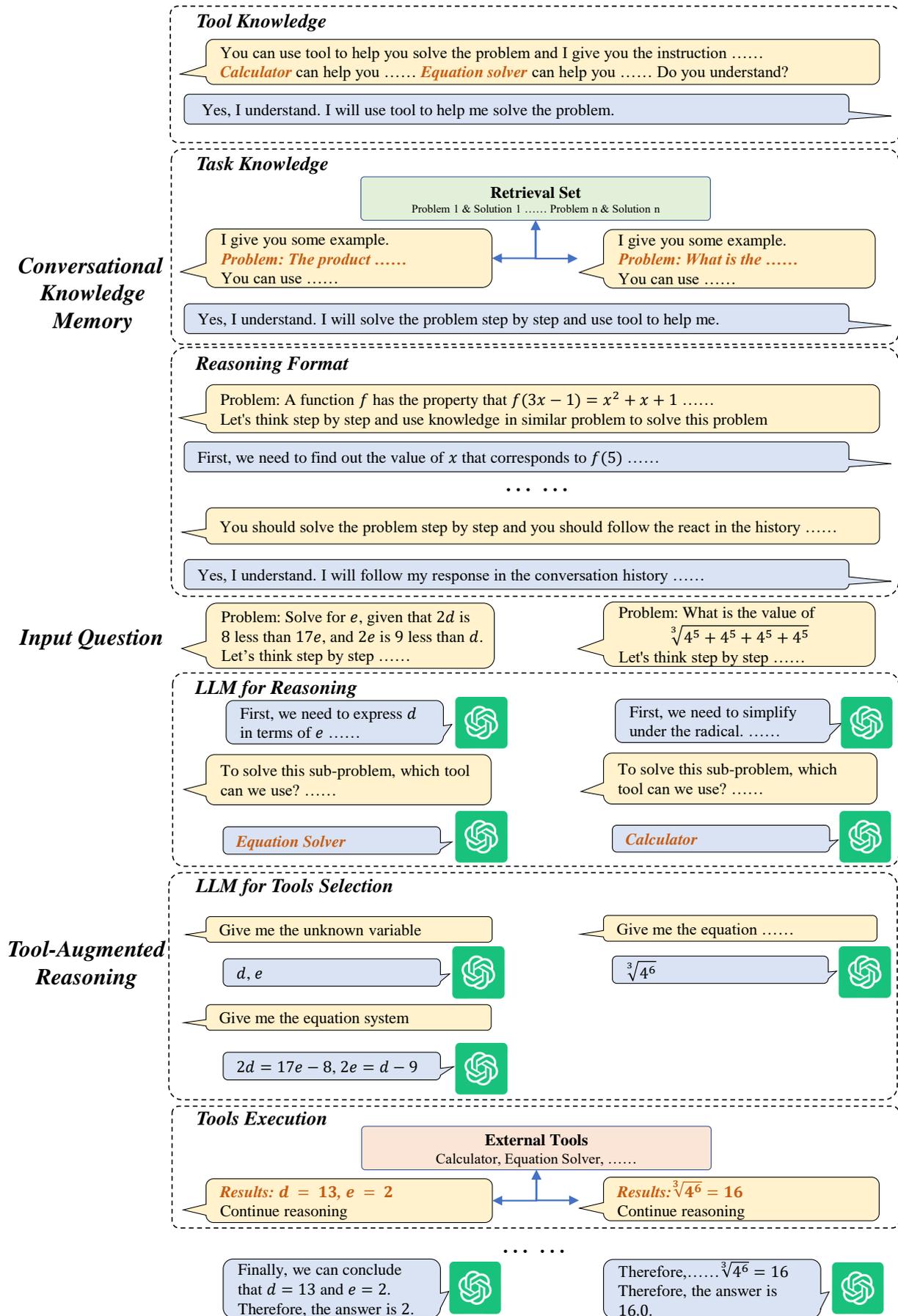


Figure 2: An illustration example for ChatCoT from MATH.