LLMs can Find Mathematical Reasoning Mistakes by Pedagogical Chain-of-Thought

Zhuoxuan Jiang[1,2] , **Haoyuan Peng**[2] , **Shanshan Feng**[3,4] , **Fan Li**[5] and **Dongsheng Li**[6]
[1]Shanghai Business School, Shanghai, China
[2]Learnable.AI Inc., Shanghai, China
[3]Centre for Frontier AI Research, A*STAR, Singapore
[4]Institute of High-Performance Computing, A*STAR, Singapore
[5]The Hong Kong Polytechnic University, Hong Kong, China
[6]Microsoft Research Asia, Shanghai, China
jzx@sbs.edu.cn, penghy15@gmail.com, victor_fengss@foxmail.com, fan-5.li@polyu.edu.hk, dongshengli@fudan.edu.cn

Abstract

Self-correction is emerging as a promising approach to mitigate the issue of hallucination in Large Language Models (LLMs). To facilitate effective self-correction, recent research has proposed mistake detection as its initial step. However, current literature suggests that LLMs often struggle with reliably identifying reasoning mistakes when using simplistic prompting strategies. To address this challenge, we introduce a unique prompting strategy, termed the Pedagogical Chain-of-Thought (PedCoT), which is specifically designed to guide the identification of reasoning mistakes, particularly mathematical reasoning mistakes. PedCoT consists of pedagogical principles for prompts (PPP) design, two-stage interaction process (TIP) and grounded PedCoT prompts, all inspired by the educational theory of the Bloom Cognitive Model (BCM). We evaluate our approach on two public datasets featuring math problems of varying difficulty levels. The experiments demonstrate that our zero-shot prompting strategy significantly outperforms strong baselines. The proposed method can achieve the goal of reliable mathematical mistake identification and provide a foundation for automatic math answer grading. The results underscore the significance of educational theory, serving as domain knowledge, in guiding prompting strategy design for addressing challenging tasks with LLMs effectively.

1 Introduction

In recent years, Large Language Models (LLMs) have emerged as the leading force in advancing various Natural Language Processing (NLP) tasks, consistently achieving state-of-the-art performance. Despite the remarkable improvement in general AI abilities, LLMs still suffer from some essential problems, e.g., hallucination, which suggests that current LLMs may generate contents that look reasonable but intrinsically illogical [Huang *et al.*, 2023b].

To overcome hallucination, many efforts have been made, among which *self-correction* is one of the most recently focused directions. It empowers LLMs to correct their outputs by zero- or few-shot prompting [Pan *et al.*, 2023]. Some researchers treat self-correction as a single-process way [Chen *et al.*, 2023; Madaan *et al.*, 2023], while some researchers break down the task into two components, i.e., *mistake finding* and *output correction*, and achieve superior performance [Tyen *et al.*, 2023]. Since finding mistakes is a fundamental and precedent skill for right correction, it is reasonable that the two-step self-correction is more reliable.

Mistake finding is a prerequisite of self-correction, while how to well achieve it is still an open challenge. Some studies show that current LLMs struggle to find mistakes reliably using basic prompting strategies [Huang *et al.*, 2023a; Tyen *et al.*, 2023]. Most previous benchmark studies mainly adopt simple and general prompting strategies, e.g., Zero-shot CoT and its variants[1]. They do not fully leverage domain-related knowledge to design prompts. Domain-related knowledge is important because there are various kinds of reasoning problems, such as word sorting, tracking shuffled objects, logical deduction and multi-step arithmetic [Srivastava *et al.*, 2023], which require different kinds of fine-grained reasoning skills. For example, word sorting needs the skills of text and order understanding while multi-step arithmetic needs the skills of calculation and deduction. Hence, without domain-related knowledge, the potential of LLMs may be underestimated due to inadequately designed prompts.

In this paper, we investigate the challenge of creating efficient prompts for a specific domain to identify reasoning errors utilizing LLMs. As a starting point, we choose the mathematical domain, a basic subject in education. On the one hand, mathematical questions queries necessitate rigorous reasoning and logic, an area where current LLMs fall

[1]They are Direct trace-level prompting, Direct step-level prompting, and CoT step-level prompting. In this paper, we follow their task settings. The details of benchmark experiments with more prompting strategies will be introduced in Section 5.

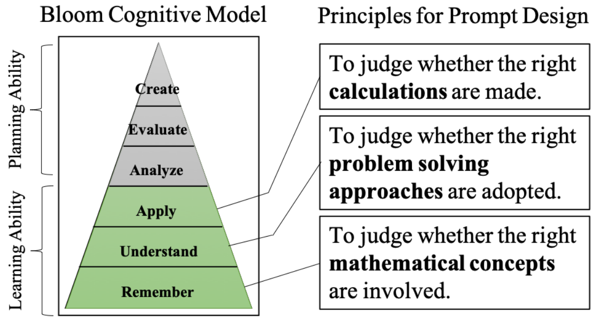

Figure 1: We develop the principles for prompt design for LLMs by leveraging the educational Bloom Cognitive Model and we focus on the *learning ability*. The bold parts are keywords used in prompts.

short. On the other hand, finding reasoning mistakes in mathematics can provide a foundation for automatic answer grading in terms of pedagogy. To this end, we conduct interdisciplinary research on LLMs prompting strategies by adopting pedagogical knowledge within this work to achieve reliable mathematical mistake finding.

In the field of pedagogy, the teaching and learning objectives are theoretically consistent with the Bloom Cognitive Model (BCM)[2] [Bloom *et al.*, 1984], which breaks the student abilities into six levels from a cognitive perspective, i.e., Remember, Understand, Apply, Analyze, Evaluate, and Create (as shown in Figure 1) [Anderson and Krathwohl, 2001]. According to this theory, the first three foundational levels are associated with *learning ability*, signifying that these skills are relatively easier to acquire, whereas the upper three levels involve *planning ability*, which are more complex to nurture and develop. We focus on the learning abilities in this work and leave the exploration of planning abilities as future work.

To align with BCM and bridge the gap in design prompting strategies for LLMs, we introduce a novel method named Pedagogical Chain-of-Thought (abbr. PedCoT). More specifically, the PedCoT strategy embodies two parts. First, following the BCM and the levels of *learning ability*, we propose the pedagogical principles of prompt (PPP) design for LLMs. The detailed content of PPP is consistent with *learning ability*, as shown in Figure 1. Second, concerning the content of prompts to be grounded when interacting with LLMs, we formulate the two-stage interaction process (TIP) and grounded prompts, consistent with the three principles.

To assess the efficacy of our approach, we gather two publicly available datasets, each containing mathematical problems of varying degrees of complexity. i.e., multi-step arithmetic and multi-step word problems. The experimental results, compared against strong baselines, consistently reveal a noteworthy contrast to what most existing studies have asserted. Specifically, we observe that current LLMs, such as GPT-4 Turbo, can effectively find mathematical reasoning mistakes through our proposed PedCoT. The result highlights the importance and value of domain knowledge in prompting the reasoning abilities of LLMs.

[2]Also called Bloom's Taxonomy of Educational Objectives in Cognitive Domain.

The main contributions of this paper include:

- We conduct an interdisciplinary study and investigate a new problem on how to leverage domain knowledge to guide prompt design for LLMs, i.e., leveraging pedagogical theories to find mathematical mistakes.
- We develop a novel zero-shot prompting strategy named PedCoT to bridge the gap between educational theory and prompts for LLMs, which is featured by (1) pedagogical principles for prompt (PPP) design, (2) two-stage interaction process (TIP) and (3) PedCoT prompts.
- Experiments on two public datasets with various complexity of math problems demonstrate that contrary to what most existing studies claim, current LLMs actually can find mathematical reasoning mistakes by using our PedCoT equipped with pedagogical theory.

2 Related Work

2.1 Automatic Answer Grading

The task of finding mistakes among mathematical reasoning steps is analogous to the task of automatic answer grading (ASG) from the perspective of pedagogy. The ASG has garnered significant attention within the educational technology community. The methods for ASG are manifold, which depend on specific subjects and/or question types, e.g., automatic scoring for essay [Xie *et al.*, 2022], reading comprehension [Fernandez *et al.*, 2022], open math response [Erickson *et al.*, 2020] and short math answering grading [Zhang *et al.*, 2022a]. The majority of these studies focus on improving semantic modeling by incorporating the most recent language models from the Natural Language Processing (NLP) community. As to the mathematical grading task, besides capturing the semantics, evaluating the rationale (intermediate reasoning steps) on the word problem step-by-step is still much more challenging [Cobbe *et al.*, 2021; Welleck *et al.*, 2022]. The emergence of LLMs significantly improves the level of machine intelligence and provides a potential direction for finding mistakes. However, the difference between ASG and mistake finding is that the former can utilize the standard answer information while the later cannot. To the best of our knowledge, this paper represents the first attempt to leverage the pedagogical theories from step-by-step grading of mathematical answers for LLMs.

2.2 Chain-of-Thought Prompting

LLMs indeed shown great potential in various tasks, but prompting them efficiently can be challenging. Recently, the concept of Chain-of-Thought (CoT) has been introduced [Wei *et al.*, 2022]. Then various zero- and few-shot CoT prompting methods are studied to demonstrate the ability of LLMs to solve challenging tasks [Wang *et al.*, 2022]. For example, in terms of zero-shot prompting, Zero-shot CoT simply adds 'Let's think step by step' after each question, which enables LLMs to generate step-by-step rationales and achieve better final answers [Kojima *et al.*, 2022]. Later, Plan-and-Solve Prompting instructs LLMs to devise a plan by breaking down the entire task into smaller ones and then executing each subtask to accomplish the plan, addressing the step missing and calculation issues which Zero-shot CoT may

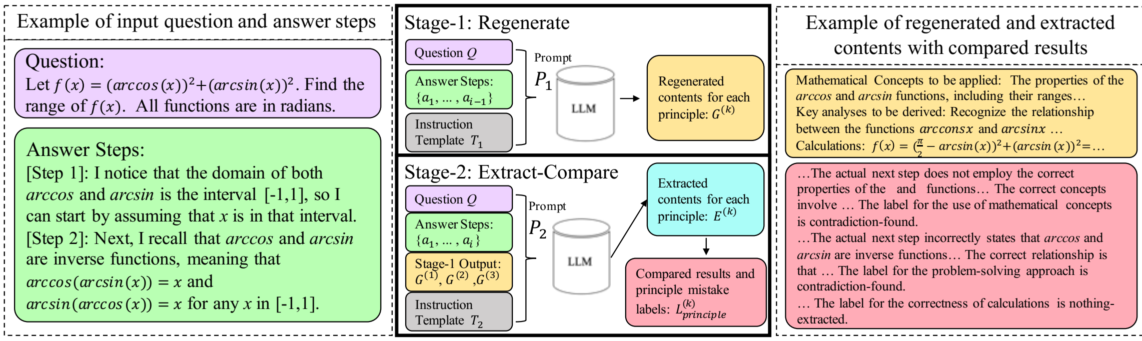

Figure 2: Diagram of two-stage interaction process (TIP) with LLMs for finding mistakes at the i-th step. The left and right parts are the exampling input and output contents. The detailed contents, as well as complete prompts, of the example can be referred to Appendix.

suffer from [Wang *et al.*, 2023]. Furthermore, AutoCoT automatically identifies the most similar questions from a corpus and uses Zero-shot CoT to generate several demonstrations to support few-shot learning on LLMs [Zhang *et al.*, 2022b].

Besides zero-shot prompting, CoT methods can be enhanced by manually providing a few exemplars as part of the prompts. This approach can more effectively harness the reasoning capabilities of LLMs [Yao *et al.*, 2023]. For example, few-shot CoT methods are proved effective by transforming the natural language-formed rationales to executable Python codes [Chen *et al.*, 2022; Gao *et al.*, 2023]. Least-to-Most Prompting decomposes complex reasoning problems into a series of simpler sub-problems and provides specific exemplars to teach LLMs [Zhou *et al.*, 2022]. Recently, a new exemplar selection scheme has been proposed, showing that choosing good exemplars with complex reasoning CoTs is important [Fu *et al.*, 2022]. Our work is in line with the challenging zero-shot CoT prompting while we focus on finding reasoning mistakes, instead of solving math problems.

2.3 Detecting Reasoning Errors with LLMs

Self-correction is one of the recently focused topics among zero- and few-shot prompting methods to tackle hallucination problems [Saunders *et al.*, 2022; Chen *et al.*, 2023; Madaan *et al.*, 2023]. It can be divided into two steps: *mistake finding* and *output correction*. The former step is regarded as a fundamental skill and normally addressed by using some few-shot prompting methods [Weng *et al.*, 2023; Ling *et al.*, 2023]. Nevertheless, recent studies demonstrate that the current LLMs still struggle to identify mistakes even in objective, unambiguous cases [Huang *et al.*, 2023a; Tyen *et al.*, 2023]. Hence, some studies indicate that even the state-of-the-art LLMs perform poorly in fixing their own reasoning errors without external feedback and LLMs are only able to improve their outputs in terms of style and quality [Kim *et al.*, 2023; Shinn *et al.*, 2023].

A recent study proposed SelfCheck [Miao *et al.*, 2023], which promotes LLMs to detect reasoning mistakes in a zero-shot manner. SelfCheck can partially address the problem of detecting reasoning mistakes of LLMs, while it still has some problems: (1) SelfCheck may be faced with a bias when the regenerated and real current steps to be compared are consistently wrong. To deal with the bias, PedCoT considers more comprehensive and reliable factors, including pedagogical theories, various mistake types, and different mistake labels; (2) SelfCheck requires five LLM requests to complete a check, and it is inefficient; and (3) SelfCheck is evaluated by its performance in detecting reasoning mistakes without comparing with other related methods. We conduct more comprehensive experiments on various CoT prompting methods and achieve the state-of-the-art performance.

3 Problem Statement and Analysis

In this section, we will firstly provide a detailed definition of the research problem. Subsequently, we will undertake a comprehensive analysis of the research problem and introduce the proposed solution to address this problem.

3.1 Problem Definition

We propose a novel problem of finding mathematical reasoning mistakes by zero-shot prompting LLMs. Mathematical reasoning mistakes can be divided into a three-hierarchy decision-making mistakes: namely principle mistake, step mistake, and trace mistake respectively.

Given a question (i.e., math problem) Q and a trace of the top i answer steps $A_i = \{a_1, ..., a_i\}$, as variables, they are integrated into predefined instruction templates $T = \{T_j^k\}$ and then the combined prompts can be represented by $P = \{P_j\}$, where j means the j-th stage of request on a pre-selected LLM (e.g., GPT-4) and k corresponds to the k-th pedagogical principle. In this work, we set the $J = 2$ and $K = 3$ since we only need two stages of requests on LLM to complete the mistake finding task and we leverage three pedagogical principles for the time being. Therefore, at Stage-1, the input prompt P_1 is combined with Q, A_{i-1} and T_1 to request on the LLM to obtain the intermediate content $G^{(k)}$. The process is named as Regenerate and is introduced in the following section. At Stage-2, the input prompt P_2 is combined with Q, A_i, T_2 and G to request on the LLM again to extract some content $E^{(k)}$ and compare the consistency between the regenerated content and extracted content. This process is called as Extract-Compare. The LLM can finally output a principle mistake label in terms of principles which is defined as: $L^{(k)}_{principle} \in$ {correct-and-aligned, reasonable-but-incomplete, contradic-

<table>
<tr><th rowspan="2">Ability Levels</th><th rowspan="2">Principles</th><th colspan="2">Instruction Templates for Two-stage PedCoT Prompts</th></tr>
<tr><th>Stage-1: Regenerate</th><th>Stage-2: Extract-Compare</th></tr>
<tr><td>Remember</td><td>Whether the right mathematical concepts are involved.</td><td rowspan="3">You are given a math problem and several initial steps …
Question: $[Q]$
Initial steps: [Previous Answer Steps $\{a_1, ..., a_{i-1}\}$]
Execute the following instructions sequentially.
1. List the mathematical concepts that … in the next step …
2. List the key analyses that … in the next step …
3. List the mathematical expressions that … in the next step …</td><td rowspan="3">You are given a math problem and several initial steps …
Question: $[Q]$
Initial steps: [Previous Answer Steps $\{a_1, ..., a_{i-1}\}$]
The actual next step is: [Current Answer Step a_i]
Execute the following instructions sequentially.
1. First extract the mathematical concepts employed … Give a detailed comparison between the … [Stage-1 Output Content $G^{(1)}$]. Finally, output a label … to categorize the the use of mathematical concepts for the actual next step.
2. First extract the key analyses present … Give a detailed comparison between the … [Stage-1 Output Content $G^{(2)}$]. Finally, output a label to categorize the correctness of the problem solving approach for the actual next step.
3. First extract the calculation results present …, and give a detailed comparison between … [Stage-1 Output Content $G^{(3)}$]. Finally, output a label … to categorize the correctness of calculations for the actual next step.</td></tr>
<tr><td>Understand</td><td>Whether the right problem solving approaches are adopted.</td></tr>
<tr><td>Apply</td><td>Whether the right calculations are made.</td></tr>
</table>

Table 1: The proposed PedCoT prompting method which evolved from left two columns of pedagogical ability levels and principles for prompt design. We design the instruction templates for PedCoT prompts that coordinate with the two-stage interaction process (TIP). The bold parts are keywords to embody the corresponding educational meanings. We refine the wording of PedCoT prompts to avoid misunderstanding by LLMs through numerous empirical attempts. The contents in [] are variables.

tion-found, nothing-extracted}, where *correct-and-aligned* means the extracted contents $E^{(k)}$ are correct and aligned with the regenerated contents $G^{(k)}$. *reasonable-but-incomplete* means the extracted contents $E^{(k)}$ are partially correct but not completely aligned with the regenerated contents $G^{(k)}$. *contradiction-found* means the extracted contents $E^{(k)}$ are totally contrary to the regenerated contents $G^{(k)}$. *nothing-extracted* means no related content is extracted from the current answer step.

With the acquired principle mistake labels in terms of each principle, then we can obtain the 1-0 binary step mistake label for the i-th step defined as the following:

$$L_{step}^{(i)} = \begin{cases} 1, & \text{if} \quad \forall k \in \{1,2,3\}, st. L_{principle}^{(k)} \neq \text{contradiction-found}, \\ 0, & \text{if} \quad \exists k \in \{1,2,3\}, st. L_{principle}^{(k)} = \text{contradiction-found}. \end{cases}$$

With the acquired step mistake labels in terms of each answer step, then we can obtain the 1-0 binary trace mistake label defined as the following:

$$L_{trace} = \begin{cases} 1, & \text{if} \quad \forall i \in \{1, ..., n\}, st. L_{step}^{(i)} = 1, \\ 0, & \text{if} \quad \exists i \in \{1, ..., n\}, st. L_{step}^{(i)} = 0, \end{cases}$$

where n is the number of steps in an answer trace. Actually, the answer steps are in a sequential order, and the step mistake labels are obtained orderly. Once the first step mistake is found, the trace is identified as mistake found and the iteration can stop. We believe that the above proposed principle mistake label schema is aligns well with educational objectives for answer grading. It provides explainable information for understanding the logic of mistake identification, as opposed to the binary judgment employed by most existing methods.

3.2 Analysis

The basic idea on addressing the problem of finding mathematical reasoning mistakes is to leverage pedagogical theories and bridge the gap between pedagogy and LLMs. Pedagogical theories are essential in the field of education as they provide a framework for understanding how and why students learn. They are widely used to assist teachers in evaluating the students' learning outcomes. Hence, it is believed that the pedagogical theories can be transferred to instruct LLMs in mistake finding and self-correction. Following the framework of pedagogical theories, we first develop the principles for prompt design, and then the grounded prompts can be designed in a systematical manner. In this paper, we develop three pedagogical principles for prompt (PPP) design according to ability levels that are defined by Bloom Cognitive Model. The detail analysis of the developed PPP is discussed as below:

Ability Level. As shown in Figure 1, six ability levels are defined by the Bloom Cognitive Model which explains the teaching objectives from the cognitive perspective of pedagogy. From bottom (low) to up (high), they are Remember, Understand, Apply, Analyze, Evaluate, and Create. Along with the theory, the lower three levels are called *learning ability* which are easier to obtain, while the higher three levels, *planning ability*, are notably more complex to nurture and develop. This paper focuses on the lower three ability levels.

Pedagogical Principles for Prompt Design. To fill the gap between pedagogical theories and grounded prompts for LLMs to execute, we propose three pedagogical principles for prompt (PPP) design, corresponding to the three ability levels. The PPP can guide the later grounded prompt design on what aspects should be involved. The right part of Figure 1 shows the content of principles. Admittedly, based on the BCM, each ability level contains various aspects of fine-grained skills, and the proposed principles cannot fully reflect them. For example, remembering *mathematical concepts* is just a representative skill among 14 counterparts at the level of Remember. The detail of ability levels can be referred to the theory [Anderson and Krathwohl, 2001]. As a starting point, we propose to defer the exploration of more intricate principles to subsequent research endeavours.

4 Methodology

Contrary to existing methodologies that solely rely on the inherent capabilities of LLMs to identify errors - an approach that often proves ineffective - we instead take a pioneer study to utilize external domain knowledge on the targeted problem.

In this paper, we design the novel PedCoT prompts for mathematical reasoning mistake finding based on the previously developed PPP design. The PedCoT prompts are also coordinated with the newly-proposed two-stage interaction process (TIP) on LLMs. We introduce the TIP first.

4.1 Two-Stage Interaction Process

Assume to find mistake at the i-th answer step, Figure 2 shows the two-stage interaction process with LLMs, including Stage-1 Regenerate and Stage-2 Extract-Compare.

Stage-1: Regenerate. Given the prompt P_1 for Stage-1, which is a combination of a math question Q, the previous answer steps $A_{i-1} = \{a_1, ..., a_{i-1}\}$ and the instruction template T_1, LLMs regenerate the potential mathematical concepts, key analysis (i.e., problem-solving approaches), and calculation results (i.e., calculations) about the expected current step, denoted as $G \in \{G^{(k)}\}$. We use regular expressions to split the output contents G into three segments to obtain each $G^{(k)}$. Unlike the majority of existing approaches that necessitate LLMs to explicitly regenerate the current step and compare it with the actual current step, our method can generate a pedagogical analysis of the step while keeping the actual current step, a_i, unseen to LLMs for the sake of objectivity.

Stage-2: Extract-Compare. Given the prompt P_2 for Stage-2, which is a combination of a math question Q, the current answer steps $A_i = \{a_1, ..., a_i\}$, the outputs G from Stage-1 Regenerate and the instruction template T_2, LLMs firstly extract the real mathematical concepts, key analysis (i.e., problem-solving approaches), and calculation results (i.e., calculations) within the current step a_i, denoted as $E \in \{E^{(k)}\}$. Then, LLMs compare the extracted contents E with regenerated contents G at Stage-1 along with the respective principle. At last, LLMs make the conclusions on whether the mathematical concepts, problem-solving approaches, and/or calculations are rightly involved in the current step a_i, i.e., giving the principle mistake label $L^{(k)}_{principle}$. Then the step mistake label $L^{(i)}_{step}$ can be acquired.

To find mistake at any step, our method only requires two requests. To the best of our knowledge, no existing other work with less than two requests can fulfil complex reasoning tasks using LLMs currently.

4.2 Pedagogical Chain-of-Thought Prompts

Based on pedagogical principles mentioned in Section 3.2, we develop the Pedagogical Chain-of-Thought (PedCoT) prompts for each stage of TIP. Each prompt consists of the instruction templates (i.e., T_1 or T_2) and the variables (i.e., Q, A and/or G). As shown in Table 1, the right two columns list the grounded contents of developed prompts for each stage. The bold parts are keywords to embody the corresponding educational meanings. We refine the wording of grounded prompts to avoid misunderstanding by LLMs through numerous empirical attempts.

By utilizing the proposed PedCoT prompts, the LLMs can perform a binary assessment of the current step's accuracy. Additionally, they can provide explanatory feedback for educational purposes by generating a textual principle mistake label for each ability level. Thus, our method can bridge the gap between pedagogical theories and prompt design for LLMs, and fulfil the task of step-by-step reasoning mistake finding in the mathematical domain.

5 Experiment

5.1 Datasets

To evaluate our proposed PedCoT, we collect two public datasets containing step-level correctness labels for mathematical problems with different difficulties.

BIG-Bench Mistake [Tyen *et al.*, 2023]: The original dataset consists 2,186 reasoning traces generated by PALM-2 [Anil *et al.*, 2023], encompassing five reasoning tasks derived from diverse domains. With an emphasis on empowering LLMs to identify reasoning errors within the realm of mathematics, we chose samples exclusively related to this field. In total, these samples encompass 1,506 reasoning steps across 300 answer traces. The type of mathematical problem consistently involves multi-step arithmetic calculations. These are relatively straightforward to solve and necessitate basic mathematical concepts and problem-solving strategies, such as the fundamental rules of arithmetic. 62 correct traces are involved to avoid the bias that LLMs may treat every trace as faulty.

PRM800K [Lightman *et al.*, 2023]: The original dataset contains about 800,000 step-level labels over 75,000 solutions for math word problems in the MATH dataset [Hendrycks *et al.*, 2021], which consists of 12,500 problems from high-school math competitions. The problems necessitate a comprehensive understanding of intricate mathematical concepts and problem-solving methodologies. We adopt this challenging dataset to verify the effectiveness and robustness of our method. We randomly select a set of 300 pairs of problems and their affiliated reasoning answer traces[3], containing 85 correct traces and 3,736 reasoning steps in total. The steps in the PRM800K dataset are categorized as positive, negative, and neutral. To use the proposed method, we adjust the labels to align with our label schema. Specifically, the positive or neutral steps are treated as right (i.e., $L_{step} = 1$) while the negative steps are labeled as wrong (i.e., $L_{step} = 0$) in our experiments.

5.2 Models, Baselines, and Metrics

Models. Our experiments are conducted with two state-of-the-art LLMs: GPT-4 [OpenAI, 2023] and its latest generation GPT-4 Turbo[4]. The temperature for generation is consistently set to 0 for both models to minimize the diversity of model outputs. We do not employ any ensemble strategies, such as self-consistency [Wang *et al.*, 2022], to maintain a pure comparison with different CoT baseline methods.

Baselines. We compare our proposed PedCoT with the following zero-shot prompting methods: (1) **Zero-shot CoT** [Kojima *et al.*, 2022]; (2) **Plan-and-Solve Prompting** [Wang *et al.*, 2023]; (3) **SelfCheck** [Miao *et al.*, 2023]. We re-implement these baselines by prompting LLMs with

[3]Accessible on github.com/HaoyuanPeng/PedCoT-IJCAI24/

[4]We used gpt-4-1106-preview, the latest version then.

Datasets	LLMs	CoT methods	MF. Acc.	Avg. F1	Cls. Acc.	P_+	R_+	F_+	P_-	R_-	F_-
BIG-Bench Mistake	GPT-4	Direct-Step†	42.67	78.67	-	-	-	-	-	-	-
		Vanilla	46.67	75.19	76.00	40.38	33.87	36.84	83.47	86.97	85.19
		Zero-shot CoT	46.33	72.82	73.33	33.93	30.65	32.20	82.38	84.45	83.40
		Plan-and-Solve	47.00	73.17	74.33	34.69	27.42	30.63	82.07	86.55	84.25
		SelfCheck	81.00	88.37	88.33	71.43	**72.58**	**72.00**	**92.83**	92.44	92.63
		PedCoT (ours)	**83.00**	**88.81**	**89.33**	**81.25**	62.90	70.91	90.87	**96.22**	**93.47**
	GPT-4 Turbo	Direct-Step†	43.33	86.24	-	-	-	-	-	-	-
		Vanilla	64.67	85.84	87.33	85.29	46.77	60.42	87.59	97.90	92.46
		Zero-shot CoT	70.00	88.73	89.67	89.74	56.45	69.31	89.66	98.32	93.79
		Plan-and-Solve	77.00	87.33	88.67	**91.18**	50.00	64.58	88.35	**98.74**	93.25
		SelfCheck	80.33	87.84	87.67	68.66	**74.19**	71.32	**93.13**	91.18	92.14
		PedCoT (ours)	**85.33**	**89.33**	**90.00**	86.36	61.29	**71.70**	90.62	97.48	**93.93**
PRM800K	GPT-4	Direct-Step	33.33	67.10	66.00	42.20	54.12	47.42	79.58	70.70	74.88
		Vanilla	37.00	73.23	73.00	52.22	55.29	53.71	81.90	80.00	80.94
		Zero-shot CoT	41.33	74.89	74.00	53.15	69.41	60.20	86.24	75.81	80.69
		Plan-and-Solve	38.00	74.79	74.00	53.27	67.06	59.37	85.49	76.74	80.88
		SelfCheck	40.00	71.79	70.33	48.59	**81.18**	60.79	**89.87**	66.05	76.14
		PedCoT (ours)	**45.33**	**79.82**	**79.33**	**61.39**	72.94	**66.67**	88.44	**81.86**	**85.02**
	GPT-4 Turbo	Direct-Step	35.00	72.76	72.67	51.72	52.94	52.33	81.22	80.47	80.84
		Vanilla	34.00	75.05	78.33	77.78	32.94	46.28	78.41	**96.28**	86.43
		Zero-shot CoT	38.67	78.96	80.67	76.47	45.88	57.35	81.53	94.42	87.50
		Plan-and-Solve	38.33	81.01	82.67	**83.67**	48.24	61.19	82.47	**96.28**	**88.84**
		SelfCheck	41.67	70.21	68.67	46.90	**80.00**	59.13	89.03	64.19	74.59
		PedCoT (ours)	**44.33**	**83.45**	**83.33**	69.66	72.94	**71.26**	**89.10**	87.44	88.26

Table 2: Comparison of various CoT methods on the BIG-Bench Mistake and PRM800K datasets. †The scores are from the original paper.

the same prompts and multi-stage pipelines. To better evaluate the abilities of LLMs on detecting reasoning tasks with CoT prompting, we also consider **Direct-Step Prompting** [Tyen *et al.*, 2023] and **Vanilla Two-stage Prompting** as our baselines. Direct-Step Prompting instructs LLMs to generate a single word *yes* for each step if it is correct, or *no* otherwise. Vanilla Two-stage Prompting first prompts LLMs to analyze each step's correctness without specific instructions. Subsequently, LLMs are prompted again to generate a label word based on their analyses.

Metrics. Similar to the previous work that evaluates the performance of mistake finding [Tyen *et al.*, 2023], we also utilize the mistake finding accuracy (abbr. MF. Acc.) and weighted average F1 (abbr. Avg. F1) scores as our metrics. Specifically, the MF. Acc. is strict for step-level accuracy which means mistakes should be found at the right step. Only the first mistake among a trace is concerned. The average F1 score provides a lenient measure of trace-level accuracy, indicating that mistakes within a trace scope can be identified. However, it does not account for whether these errors are pinpointed at the correct step. In addition, the binary classification accuracy (abbr. Cls. Acc.), as well as the precision (P_+ and P_-), recall (R_+ and R_-) and F1 (F_+ and F_-) for each label, are reported in the experiment.

5.3 Results and Analysis

Table 2 reports the comparison among different zero-shot CoT methods applied with GPT-4 and GPT-4 Turbo on the two datasets. Consistent with previous research, directly prompting the LLMs to output a single word for the correctness of each reasoning step (i.e., Direct-Step) is ineffective even with the state-of-the-art LLMs. Vanilla Two-stage Prompting allows LLMs to generate several analyses before being prompted again for the final label word, leading to a slightly better performance than Direct-Step. Zero-shot CoT and Plan-and-Solve Prompting explicitly instruct LLMs to generate intermediate analyses, and show improvements compared to Vanilla Two-stage Prompting. The above results indicate that the more analysis contents before giving the final judgment are generated by prompting LLMs, the better performance is obtained, especially with GPT-4 Turbo.

Our PedCoT consistently outperforms all the other baselines by a remarkable margin on the two datasets and two LLMs. Particularly, on the BIG-Bench Mistake dataset, PedCoT surpasses SelfCheck across both Acc. and Avg. F1 metrics. On the PRM800K dataset, PedCoT achieves a higher MF. Acc. score ranging from 4.00% to 7.33% compared to two-stage CoT baselines when using GPT-4. Additionally, its Avg. F1 and Cls. Acc. scores surpass these baselines by about 4.93% to 5.33%. Compared to SelfCheck, PedCoT exhibits a 5.33% leading in MF. Acc., and a considerable leading of 8.03% and 9.00% in Avg. F1 and Cls. Acc. scores respectively. When using GPT-4 Turbo, PedCoT exhibits a MF. Acc. advantage of 5.66% compared to the two-stage CoT baselines, along with improvements in Avg. F1 and Cls. Acc. In comparison to SelfCheck, PedCoT achieves a higher MF. Acc. by 2.66%, and a leading of 13.24% and 14.66% in Avg. F1 and Cls. Acc. respectively. All of the above results demonstrate the effectiveness of leveraging pedagogical knowledge in our proposed methods.

Additionally, although SelfCheck outperforms other baselines on BIG-Bench Mistake dataset, as displayed in Table 2, its Avg. F1 and Cls. Acc. scores on PRM800K dataset fall behind, as shown in Table 2. The results may be caused by the following reasons: During the process of SelfCheck prompting, one of the LLM requests is to summarize the target of the current answer step, and then regenerate a new current step for subsequent comparison. For the BIG-Bench Mistake dataset, due to the easy arithmetic ability required to solve math problems, SelfCheck can perform well. However, the PRM800K dataset presents more complex math problems requiring advanced reasoning skills. In such cases, SelfCheck

LLMs	Methods	BIG-Bench Mistake			PRM800K		
		MF. Acc.	Avg. F1	Cls. Acc.	MF. Acc.	Avg. F1	Cls. Acc.
GPT-4	PedCoT	90.67	91.78	92.00	46.67	77.87	77.33
	-Principle 1	89.33	90.54	90.67	42.67	74.06	73.33
	-Principle 2	89.33	90.54	90.67	42.67	76.00	76.00
	-Principle 3	28.00	37.44	36.00	32.00	57.92	57.33
GPT-4 Turbo	PedCoT	92.00	91.51	92.00	53.33	84.32	84.00
	-Principle 1	90.67	91.18	92.00	52.00	82.77	82.67
	-Principle 2	90.67	92.00	92.00	52.00	84.18	82.67
	-Principle 3	21.33	32.22	30.67	36.00	62.29	61.33

Table 3: Ablation experiment of PedCoT by ablating one principle.

may summarize an incorrect target, leading to a incidental alignment with an incorrect actual current step. This bias in SelfCheck can result in erroneously classifying the real current step as correct, impacting the precision for correct traces (P_+) and reducing recall for incorrect traces (R_-). The reason of our method can surmount these challenges met with SelfCheck, by our analysis, is that the PedCoT is grounded in pedagogical principles and allows LLMs to remain blind to the current step. This approach enables the LLMs to render judgments from an impartial and unbiased perspective.

5.4 Ablation Study

We design ablation experiments to assess the contribution of each principle. Specifically, we deactivate the three principles one by one, and observe the performance change patterns of our proposed methods. In this study, deactivating a principle means that this principle's corresponding prompt contents are removed. We select 25% of the test set for ablation experiments. The results are reported in Table 3.

Ablating prompt contents of Principle 1 (Remember). We observe a slight decrease in metrics when Principle 1 is deactivated, which is about mathematical concepts. Hence, a positive contribution of Principle 1 on both BIG-Bench Mistake dataset and PRM800K dataset is validated. Meanwhile, the experimental results on the BIG-Bench Mistake dataset exhibit a relatively smaller decrease, which leads us to believe the easier arithmetic problems of BIG-Bench Mistake dataset require only the concept of the rules of arithmetic.

Ablating prompt contents of Principle 2 (Understand). We observe some fluctuations in the performance of the proposed method on BIG-Bench Mistake, while there is a slight decrease in performance on the PRM800K dataset after deactivating Principle 2, which is about problem-solving approaches. The observed variations can also be attributed to the varying complexities of the problems. Our analysis suggests that the strategies required to solve arithmetic problems are less intricate compared to those needed for complex mathematical word problems.

Ablating prompt contents of Principle 3 (Apply). When the portion of the prompt representing Principle 3, which is about calculations, is removed from the prompt, there is a significant decline in the performance of the proposed method across both datasets and both LLMs. This indicates that verifying the correctness of calculations, including mathematical expression transformations and simplifications, is the most fundamental and crucial ability required for prompting the LLMs to detect mathematical reasoning errors.

Dataset	Strategy	MF. Acc.	Cls. Acc.	P_+	R_-
BIG-Bench Mistake	PedCoT	83.00	89.33	81.25	96.22
	One-Stage	74.33	81.00	53.33	85.29
PRM800K	PedCoT	45.33	79.33	61.39	81.86
	One-Stage	35.00	61.00	38.41	60.47

Table 4: Two-stage PedCoT vs. its one-stage variant on GPT-4.

Since the datasets have varying complexities of mathematical problems, the performance change patterns are not always significant. Regardless the performance stays stable or shows a significant decrease, it can be concluded that each principle has a positive impact on the PedCoT method. This demonstrates the effectiveness of incorporating the pedagogical theory of each *learning ability* from BCM.

5.5 Study on One-Stage vs. Two-Stage Prompting

In this section, we investigate the necessity of two-stage prompting. A distinctive feature of our two-stage methodology is that, at Stage-1, the current step remains undisclosed to LLMs and thus they can make an unbiased judgement at Stage-2. To verify the necessity, we construct a version of one-stage prompt by combining our two-stage prompts.

We compare the performance between the two-stage PedCoT and its one-stage variant, utilizing two datasets with GPT-4. As presented in Table 4, the precision of correct traces (P_+) and the recall of incorrect traces (R_-) significantly degrade when employing one-stage prompting. This suggests that the LLMs would be misled by the real current step present in the one-stage prompt. Based on our analysis, it appears that when the real current step is shared with the LLMs, they may encounter difficulties in independently generating analyses. Instead, they seem to rely heavily on referencing the real current step. We call this phenomenon 'lazy LLMs' since LLMs are inclined to cater to the ideas present in prompts. Consequently, finding mistakes by giving the real current step becomes difficult to accomplish. This experiment demonstrates that our two-stage prompting is necessary.

6 Conclusion

In this paper, we investigate a novel problem of how to leverage domain knowledge of pedagogy to enhance the complex reasoning abilities of the zero-shot prompted LLMs. We learn from pedagogical theories to instruct LLMs and to guide the prompts design for LLMs. We propose a novel strategy, namely PedCoT method, which consists of three pedagogy principles for prompt design, two-stage interaction process, and pedagogical Chain-of-Thought prompts. We experiment two public datasets of math problems with various complexity levels. The state-of-the-art performance of our proposed method indicates that it can achieve efficient reasoning mistaking finding with only two requests to LLMs. In addition, the ablation experiment and study on one-stage method validate the importance of the three pedagogical principles and the necessity of two-stage prompting. Overall, this finding indicates that LLMs actually can find mathematical reasoning mistakes by resorting to domain knowledge.

Acknowledgments

This work was supported by the Humanities and Social Science Fund of Ministry of Education (No. 23YJC790187). We thank the anonymous reviewers for their insightful and constructive comments.

Contribution Statement

Authors: Zhuoxuan Jiang, Haoyuan Peng, Shanshan Feng, Fan Li and Dongsheng Li

Zhuoxuan Jiang: Conceptualized the study, designed experiments, wrote the manuscript, supervised the project and secured funding.

Haoyuan Peng: Contributed to data collection, conducted experiments, performed result analyses, crafted the prompts and critically revised the manuscript.

Shanshan Feng, Fan Li and Dongsheng Li: Provided critical feedback on the manuscript, and edited for clarity.

Equal Contribution: Zhuoxuan Jiang and Haoyuan Peng contributed equally to this work and listed in alphabetical order. Zhuoxuan Jiang is the corresponding author.

Order of Authors: Authors are listed in order of decreasing contribution.

References

[Anderson and Krathwohl, 2001] Lorin Anderson and David Krathwohl. *Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. Longman, New York, 2001.

[Anil *et al.*, 2023] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. https://arxiv.org/abs/2305.10403, 2023. Accessed: 2024-01-05.

[Bloom *et al.*, 1984] Benjamin S. Bloom, David R. Krathwohl, and Bertram B. Masia. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Longman, New York, 1984.

[Chen *et al.*, 2022] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. https://arxiv.org/abs/2211.12588, 2022. Accessed: 2024-01-05.

[Chen *et al.*, 2023] Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. Iterative translation refinement with large language models. https://arxiv.org/abs/2306.03856, 2023. Accessed: 2024-01-05.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. https://arxiv.org/abs/2110.14168, 2021. Accessed: 2024-01-05.

[Erickson *et al.*, 2020] John A. Erickson, Anthony F. Botelho, Steven McAteer, Ashvini Varatharaj, and Neil T. Heffernan. The automated grading of student open responses in mathematics. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, LAK '20, pages 615–624, 2020.

[Fernandez *et al.*, 2022] Nigel Fernandez, Aritra Ghosh, Naiming Liu, Zichao Wang, Benoît Choffin, Richard Baraniuk, and Andrew Lan. Automated scoring for reading comprehension via in-context BERT tuning. In *23rd International Conference on Artificial Intelligence in Education (AIED 2022)*, 2022.

[Fu *et al.*, 2022] Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning. https://arxiv.org/abs/2210.00720, 2022. Accessed: 2024-01-05.

[Gao *et al.*, 2023] Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. PAL: Program-aided language models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 10764–10799, 2023.

[Hendrycks *et al.*, 2021] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. https://arxiv.org/abs/2103.03874, 2021. Accessed: 2024-01-05.

[Huang *et al.*, 2023a] Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. https://arxiv.org/abs/2310.01798, 2023. Accessed: 2024-01-05.

[Huang *et al.*, 2023b] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. https://arxiv.org/abs/2311.05232, 2023. Accessed: 2024-01-05.

[Kim *et al.*, 2023] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks. https://arxiv.org/abs/2303.17491, 2023. Accessed: 2024-01-05.

[Kojima *et al.*, 2022] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[Lightman *et al.*, 2023] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. https://arxiv.org/abs/2305.20050, 2023. Accessed: 2024-01-05.

[Ling *et al.*, 2023] Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. Deductive verification of chain-of-thought reasoning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[Madaan *et al.*, 2023] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Sean Welleck, Bodhisattwa Prasad Majumder, Shashank Gupta, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. https://arxiv.org/abs/2303.17651, 2023. Accessed: 2024-01-05.

[Miao *et al.*, 2023] Ning Miao, Yee Whye Teh, and Tom Rainforth. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning. https://arxiv.org/abs/2308.00436, 2023. Accessed: 2024-01-05.

[OpenAI, 2023] OpenAI. Gpt-4 technical report. https://arxiv.org/abs/2303.08774, 2023. Accessed: 2024-01-05.

[Pan *et al.*, 2023] Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang. Automatically correcting large language models: Surveying the landscape of diverse self-correction strategies. https://arxiv.org/abs/2308.03188, 2023. Accessed: 2024-01-05.

[Saunders *et al.*, 2022] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. https://arxiv.org/abs/2206.05802, 2022. Accessed: 2024-01-05.

[Shinn *et al.*, 2023] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. https://arxiv.org/abs/2303.11366, 2023. Accessed: 2024-01-05.

[Srivastava *et al.*, 2023] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, and et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.

[Tyen *et al.*, 2023] Gladys Tyen, Hassan Mansoor, Peter Chen, Tony Mak, and Victor Cărbune. Llms cannot find reasoning errors, but can correct them! https://arxiv.org/abs/2311.08516, 2023. Accessed: 2024-01-05.

[Wang *et al.*, 2022] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. https://arxiv.org/abs/2203.11171, 2022. Accessed: 2024-01-05.

[Wang *et al.*, 2023] Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. https://arxiv.org/abs/2305.04091, 2023. Accessed: 2024-01-05.

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

[Welleck *et al.*, 2022] Sean Welleck, Jiacheng Liu, Ximing Lu, Hannaneh Hajishirzi, and Yejin Choi. Naturalprover: Grounded mathematical proof generation with language models. In *Advances in Neural Information Processing Systems*, 2022.

[Weng *et al.*, 2023] Yixuan Weng, minjun zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. Large language models are better reasoners with self-verification. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2550–2575, 2023.

[Xie *et al.*, 2022] Jiayi Xie, Kaiwei Cai, Li Kong, Junsheng Zhou, and Weiguang Qu. Automated essay scoring via pairwise contrastive regression. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2724–2733, 2022.

[Yao *et al.*, 2023] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

[Zhang *et al.*, 2022a] Mengxue Zhang, Sami Baral, Neil Heffernan, and Andrew Lan. Automatic short math answer grading via in-context meta-learning. In *Proceedings of the 15th International Conference on Educational Data Mining (EDM 2022)*, pages 122–132, 2022.

[Zhang *et al.*, 2022b] Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in large language models. https://arxiv.org/abs/2210.03493, 2022. Accessed: 2024-01-05.

[Zhou *et al.*, 2022] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. Least-to-most prompting enables complex reasoning in large language models. https://arxiv.org/abs/2205.10625, 2022. Accessed: 2024-01-05.