

INSTRUCTING LARGE LANGUAGE MODELS TO IDENTIFY AND IGNORE IRRELEVANT CONDITIONS

Anonymous authors

Paper under double-blind review

ABSTRACT

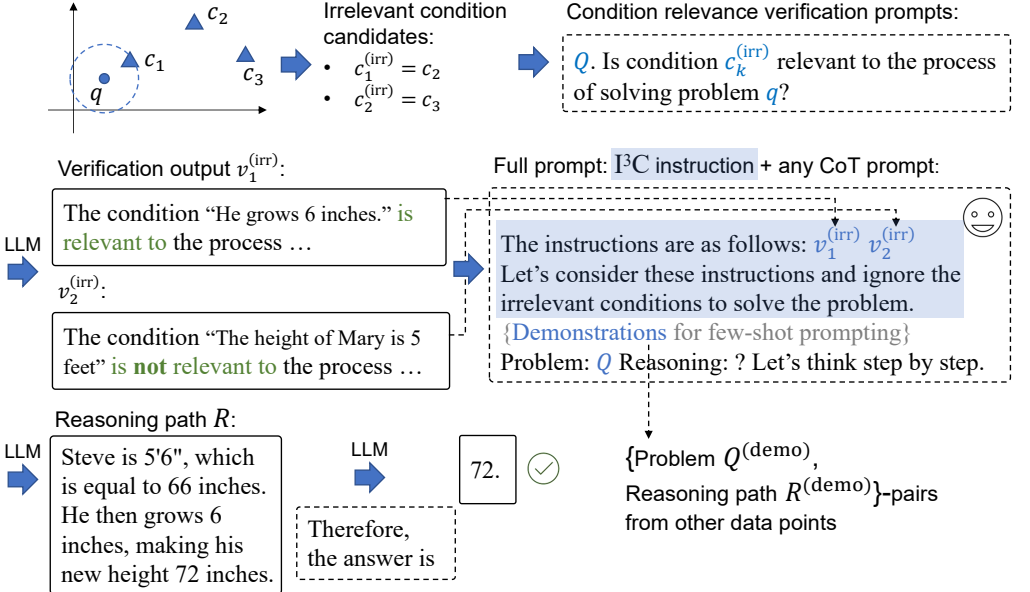
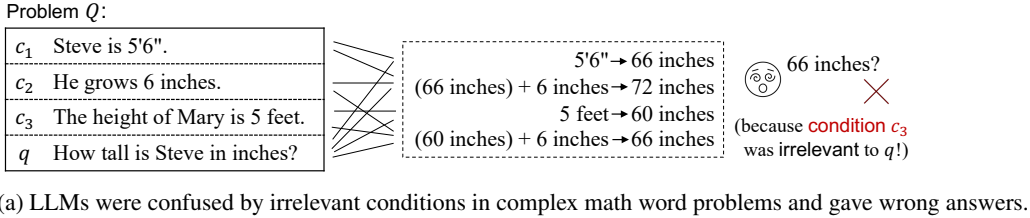
1 Math word problem (MWP) solving requires generating a reasoning path based
 2 on a given problem description that often contains *irrelevant conditions*. Existing
 3 chain-of-thought (CoT) prompting methods elicited multi-step reasoning abilities
 4 of large language models (LLMs) to solve MWPs. However, they were seriously
 5 confused by the irrelevant conditions, resulting in low accuracy. In this paper, we
 6 propose a novel approach named I³C that instructs LLMs to identify and ignore
 7 irrelevant conditions. It identifies a set of irrelevant condition candidates that have
 8 a weak semantic relevance with the question. Then it prompts LLMs to verify the
 9 irrelevant conditions. Lastly it instructs the LLMs with the verification on relevant
 10 and irrelevant conditions to avoid confusion and improve reasoning paths. More-
 11 over, we propose to select (problem, reasoning paths)-pairs as demonstrations to
 12 enhance I³C with few-shot reasoning. We develop I³C-Select that selects the most
 13 confusing problems based on the semantic relevance measurement. We conduct
 14 extensive experiments on six MWP datasets. I³C can be combined with any CoT
 15 prompting methods to improve the performance of solving MWPs. Notably, I³C-
 16 Select achieves an accuracy of 93.7 and 90.9 on GSM-IC2-1K and GSM-ICM-1K,
 17 respectively, significantly outperforming the state-of-the-art few-shot prompting
 18 method Auto-CoT by +19.4 and +25.7.

19 1 INTRODUCTION

20 Math word problem (MWP) solving is a task of developing algorithms to generate a reasoning path
 21 towards an unknown quantity based on a problem description. This task is challenging as it requires
 22 mathematical understanding and multi-step reasoning abilities. Chain-of-thought (CoT) prompting
 23 methods were able to guide large language models (LLMs) to perform complex multi-step reasoning
 24 (Kojima et al., 2022; Wang et al., 2023a). Adding demonstrations created manually Wei et al. (2022)
 25 or retrieved from a large training set Zhang et al. (2023) in CoT prompts enabled few-shot in-context
 26 learning and improved accuracy. However, Shi et al. found that LLMs could be seriously confused
 27 by irrelevant conditions which are specifications or data presented in a problem that are unrelated to
 28 the solution (Kellogg, 2016). For example, as shown in Figure 1a, the third condition “*The height of*
 29 *Mary is 5 feet.*” was irrelevant to the final question and misled the reasoning and prediction. Shi et al.
 30 added a plain instruction “*Feel free to ignore irrelevant conditions in the problem description.*” in
 31 the prompts, but the LLMs could not effectively ignore them in the problem solving process because
 32 they were not identified or specified in the instruction.

33 Improving the reasoning on MWPs that have irrelevant conditions is non-trivial. Self-consistency
 34 (Wang et al., 2023b) was proposed to repeatedly solve a problem multiple times (e.g., 10 times) and
 35 employ a majority vote strategy to determine the most consistent answer as the final answer. How-
 36 ever, it was computationally expensive and still confused by the irrelevant conditions. Moreover, the
 37 demonstrations would have to be re-designed to obtain the few-shot learning ability of identifying
 38 and ignoring the irrelevance, compared to those in (Wei et al., 2022; Zhang et al., 2023).

39 In this paper, we propose a novel approach named I³C to instruct LLMs to explicitly Identify and
 40 Ignore Irrelevant Conditions in the mathematical reasoning process. It creates effective instructions
 41 that can be added to any CoT prompts to improve their generated reasoning paths. Unlike self-
 42 consistency, I³C does not prompt LLMs many times. Its advanced variant I³C-Select uses the most
 43 confusing problems and their generated reasoning paths as demonstrations for few-shot learning.



(b) I³C performs three steps: (1) Identify irrelevant condition candidates by encoding and condition-question similarity scoring; (2) Use LLMs to verify if the candidates are relevant; (3) Leverage the verifications (and demonstrations) to generate accurate reasoning paths and find correct answers.

Figure 1: The proposed I³C approach instructs LLMs to Identify and Ignore Irrelevant Conditions.

44 First, we quantify the semantic relevance of each condition c_i in a MWP $Q = [\{c_i\}, q]$. Specifically,
 45 we use a pre-trained language model like SimCSE (Gao et al., 2021) to encode the conditions $\{c_i\}$
 46 and question sentence q . The semantic relevance is lower if the condition’s encoding is more distant
 47 from the encodings of question and other conditions, as shown in Figure 1b. Then we identify a set
 48 of irrelevant condition candidates, like c_2 and c_3 in this example, and we denote them by $\{c_k^{(irr)}\}$.

49 Next we use a LLM to verify if the candidates are indeed irrelevant. For each candidate $c_k^{(irr)}$, the
 50 verification prompt is a natural language question consisted of itself, Q , and q . The verification
 51 output usually has the explicit answers "... is (not) relevant to ...", denoted by $v_k^{(irr)}$.

52 Finally we put all the verification outputs $\{v_k^{(irr)}\}$ to create a novel instruction which helps the LLM
 53 to identify and ignore irrelevant conditions in the problem description, so-called I³C. The I³C in-
 54 struction is a plug-and-play module and can be added to any CoT prompting methods to help LLMs
 55 avoid confusion and improve their generated reasoning paths.

56 To enable few-shot in-context learning, we further develop I³C-Select that uses the pairs of solved
 57 problems and generated reasoning paths to automatically construct effective demonstrations. Specif-
 58 ically, it defines the confusion score of each solved problem: the score is higher, if the semantic
 59 relevance of its conditions is lower; and the problems of the highest confusion scores are selected.

60 Experiments on GPT-3 demonstrate that adding the I³C instruction to CoT prompting methods im-
 61 proves their performance. For example, adding I³C instruction to Manual-CoT improves the accu-
 62 racy by +5.1 on AddSub, +3.4 on SVAMP, +4.7 on GSM8K, +2.6 on SingleEq, +8.1 on GSM-
 63 IC2-1K, and +5.5 on GSM-ICM-1K. Moreover, I³C-Select beats existing prompting methods by a
 64 striking margin on six MWP datasets. Specifically, I³C-Select boosts the performance of Zero-Shot-
 65 CoT method by +23.0 on GSM-IC2-1K, +28.4 on GSM-ICM-1K, and +12.0 on GSM8K.

66 2 RELATED WORK

67 2.1 MATH WORD PROBLEM SOLVING

68 Our work is related to existing efforts on solving MWPs. Traditional methods used statistical learn-
69 ing to extract entities, quantities, and operators from a question and generated an arithmetic equation
70 to find the answer (Hosseini et al., 2014; Roy et al., 2015; Zhou et al., 2015; Mitra & Baral, 2016).
71 Later, sequence-to-sequence (Seq2Seq) model and recurrent neural networks directly transformed
72 the question into an arithmetic equation (Wang et al., 2017; 2019). Recently, fine-tuned pre-trained
73 language models have significantly improved the validity of generated equations and accuracy of an-
74 swers (Shen et al., 2021; Liang et al., 2022; 2023). However, these methods require a large amount
75 of human annotations, lacking the ability to generalize to new kinds of MWPs. In this work, we aim
76 to prompt LLMs to answer arbitrary MWPs without human annotations or task-specific fine-tuning.
77 Our approach generates reasoning paths so that researchers can investigate the behaviors of LLMs.

78 2.2 CHAIN-OF-THOUGHT PROMPTING METHODS

79 CoT prompting methods have enabled LLMs to generate reasoning paths and solve complex MWPs
80 (Kojima et al., 2022). The reasoning paths could be more expressive if the prompts were added
81 with “*Let’s think step by step*”. To mitigate missing-step errors, Plan-and-Solve (PS) prompting
82 methods instructed the LLMs to devise a plan to break down the entire task into smaller subtasks,
83 and then carry out the subtasks according to the plan (Wang et al., 2023a). Manual-CoT, as a type
84 of few-shot prompting, manually designed demonstrations to elicit multi-step reasoning ability of
85 the LLMs (Wei et al., 2022). Program of Thought (PoT) generated programming language state-
86 ments and used a program interpreter to execute the generated program to get final answers (Chen
87 et al., 2022a). Zhang et al. (2023) designed Auto-CoT that sampled diverse questions from data and
88 minimized the manual effort on finding demonstrations. Aware of irrelevant conditions in the prob-
89 lem description, Instruct-CoT added the instruction “*Feel free to ignore irrelevant conditions in the*
90 *problem description*” in the prompt (Shi et al., 2023). These methods do not explicitly specify the
91 irrelevant conditions in the prompt, which makes it difficult for LLMs to identify and ignore irrel-
92 evant conditions in the problem solving process. Our method identifies irrelevant conditions in the
93 problem description, instructs the LLMs to ignore them, and achieves significantly higher accuracy.

94 2.3 IDENTIFY AND IGNORE IRRELEVANT INFORMATION

95 Jia & Liang (2017) have shown that neural question answering systems are confused when para-
96 graphs contain irrelevant information. Several studies have trained models to identify and filter out
97 the irrelevant information. For example, Roy & Roth (2015) trained a classifier and scored the like-
98 lihood of each quantity in the problem being an irrelevant quantity. Kim et al. (2022) employed a
99 new training loss to remove the attribute-irrelevant information from the semantic encoder output.
100 Li et al. (2022) proposed a multi-scale knowledge-aware transformer to eliminate identity-irrelevant
101 information. Yang et al. (2023) leveraged pre-extracted semantic information to improve the prepro-
102 cessor’s ability to accurately identify and filter out task-irrelevant information. All these methods
103 require massive human annotations. In contrast, our method does not require time-consuming train-
104 ing or fine-tuning. It employs large language models to automatically identify irrelevant conditions
105 and generate instructions to help the models ignore them.

106 3 PROPOSED APPROACH

107 3.1 OVERVIEW

108 In this section, we elaborate on how to instruct LLMs to identify and ignore irrelevant conditions in
109 the math word problem description. Given a complex problem, we first identify a set of irrelevant
110 condition candidates that have a weak semantic relevance with the question (§ 3.2). Then we prompt
111 LLMs to verify if the candidates are indeed irrelevant. Putting all the verification results together,
112 we create a novel I³C instruction to instruct the LLMs to ignore the irrelevant conditions in the
113 problem description. The I³C instruction can be added to any CoT prompting methods to help LLMs
114 avoid confusion and improve their generated reasoning paths. Furthermore, we develop a few-shot

115 prompting method I³C-Select that selects the most confusing problems and their reasoning paths as
 116 demonstrations, and adds the I³C instruction before the demonstrations in the prompt. Given the
 117 prompt and a target problem, the LLMs generate an accurate reasoning path to improve the solving
 118 process. We introduce the I³C instruction in § 3.3 and I³C-Select method in § 3.5.

119 3.2 IDENTIFY A SET OF IRRELEVANT CONDITION CANDIDATES

120 Given a math word problem Q , we first split it into n conditions $\{c_i\}_{i=1}^n$ and a question sentence
 121 q , where each condition describes at most one quantity. So we have $Q = [\{c_i\}, q]$. For example,
 122 in Figure 1a, the conditions are {"Steve is 5'6"}, {"He grows 6 inches"}, {"The height of Mary is 30
 123 feet"}, and the question sentence is "How tall is Steve in inches?"

124 Next, we use a pre-trained language model, e.g., SimCSE (Gao et al., 2021), to encode the con-
 125 ditions and question sentence into vector representations. So we have $\{c_i\}_{i=1}^n$ and \mathbf{q} which are
 126 d -dimensional vectors. We set $d = 1,024$.

127 Then for each condition c_i , we calculate the average similarity between c_i and all other conditions
 128 in Q using cosine similarity, because the SimCSE embeddings were trained on cosine similarity:

$$s_i^{(c)} = \frac{1}{n-1} \sum_{j=1, j \neq i}^n \cos(\mathbf{c}_i, \mathbf{c}_j) = \frac{1}{n-1} \sum_{j=1, j \neq i}^n \frac{\mathbf{c}_i^\top \mathbf{c}_j}{\|\mathbf{c}_i\| \cdot \|\mathbf{c}_j\|}. \quad (1)$$

129 We also calculate the similarity between c_i and q : $s_i^{(q)} = \cos(\mathbf{c}_i, \mathbf{q})$. So we have $\{s_i^{(c)}, s_i^{(q)}\}_{i=1}^n$.

130 Now we can define a set of *irrelevant condition candidates* $\mathcal{I} \subset \{c_i\}_{i=1}^n$ for each math word prob-
 131 lem. A condition c_i is potentially irrelevant if its semantic relevance is lower than expectation. In
 132 other words, if $s_i^{(c)} < \theta$ or $s_i^{(q)} < \theta$, \mathcal{I} has c_i . We re-index the conditions in the set: $\mathcal{I} = \{c_k^{(irr)}\}_{k=1}^{|\mathcal{I}|}$.
 133 The threshold θ is a hyperparameter. We set $\theta = 0.5$.

134 We can further define the *confusion score* of a math word problem Q . We assume that the problem
 135 is more confusing if its conditions are less relevant with the final question. So the confusion score
 136 is defined as the inverse of the average similarity between any condition and the question:

$$\text{conf}(Q) = \left[\frac{1}{n} \sum_{i=1}^n \cos(\mathbf{c}_i, \mathbf{q}) \right]^{-1}. \quad (2)$$

137 The most confusing problems, i.e., the problems of the highest confusion scores, and their generated
 138 reasoning paths, will be automatically used as demonstrations in a few-shot setting. The demos
 139 teach LLMs to better solve confusing problems. Later sections give details.

140 3.3 CONSTRUCT I³C INSTRUCTION

141 Given a set of irrelevant condition candidates \mathcal{I} , we use a LLM to verify if the candidates are indeed
 142 irrelevant. For a math word problem Q , its final question q , and a condition candidate $c_k^{(irr)} \in \mathcal{I}$,
 143 we construct a verification prompt: " Q . Is condition $c_k^{(irr)}$ relevant to the process of solving problem
 144 q ?" We feed the prompt to a LLM and receive a piece of text $v_k^{(irr)}$ justifying if $c_k^{(irr)}$ is relevant or
 145 indeed irrelevant. So we have a set of verification outputs (size $|\mathcal{I}|$): $\{v_k^{(irr)}\}_{k=1}^{|\mathcal{I}|}$.

146 Now we can create a novel instruction to help LLMs identify and ignore irrelevant conditions in the
 147 problem description. In a zero-shot setting, the instruction starts with all the verification outputs.
 148 Specifically, this I³C instruction, simply denoted by I , is "*The instructions are as follows: $v_1^{(irr)} \dots$
 149 $v_{|\mathcal{I}|}^{(irr)}$. Let's consider these instructions and ignore the irrelevant conditions to solve the problem*".
 150 In case where \mathcal{I} is an empty set, we follow the Instruct-CoT method (Shi et al., 2023) and use the
 151 sentence "*Feel free to ignore irrelevant conditions in the problem description*" as the instruction.

152 3.4 GENERATE REASONING PATHS AND ANSWERS WITH I³C INSTRUCTION

153 The I³C instruction can be added to any CoT prompting methods such as Zero-Shot-CoT (Kojima
 154 et al., 2022), PS (Wang et al., 2023a), Instruct-CoT (Shi et al., 2023), Manual-CoT (Wei et al., 2022),

Table 1: Accuracy (%) comparison on six MWP datasets. I³C indicates that instructs LLMs to identify and ignore irrelevant conditions. Adding the I³C instruction to CoT prompting methods effectively improves performance. Selecting the most confusing problems and their generated reasoning paths as demonstrations for few-shot learning (i.e., I³C-Select) achieves state-of-the-art performance on all six MWP datasets. The best performance for each dataset is shown in bold.

Method (text-davinci-003)	Dataset					
	AddSub	SVAMP	GSM8K	SingleEq	GSM-IC2-1K	GSM-ICM-1K
Direct	89.3	65.2	15.0	84.6	22.8	9.0
Direct + I ³ C	92.4 (+3.1)	74.5 (+9.3)	49.7 (+34.7)	92.7 (+8.1)	82.6 (+59.8)	66.9 (+57.9)
Zero-Shot-CoT	84.8	74.3	60.8	89.5	70.7	62.5
Zero-Shot-CoT + I ³ C	91.7 (+6.9)	75.9 (+1.6)	61.3 (+0.5)	93.7 (+4.2)	84.7 (+14.0)	71.4 (+8.9)
PS	88.1	72.0	58.2	89.2	70.9	63.5
PS + I ³ C	91.4 (+3.3)	75.6 (+3.6)	61.1 (+2.9)	93.1 (+3.9)	84.8 (+13.9)	69.4 (+5.9)
Instruct-CoT	90.4	76.3	57.8	91.1	82.4	64.3
Instruct-CoT + I ³ C	91.8 (+1.4)	77.0 (+0.7)	61.0 (+3.2)	92.7 (+1.6)	84.7 (+2.3)	71.3 (+7.0)
Manual-CoT	87.8	76.7	56.9	91.3	73.9	60.6
Manual-CoT + I ³ C	92.9 (+5.1)	80.1 (+3.4)	61.6 (+4.7)	93.9 (+2.6)	82.0 (+8.1)	66.1 (+5.5)
Auto-CoT	90.6	77.8	58.9	90.9	74.3	65.2
Auto-CoT + I ³ C	93.7 (+3.1)	80.0 (+2.2)	61.9 (+3.0)	93.5 (+2.6)	83.9 (+9.6)	68.2 (+3.0)
I ³ C-Select (Ours)	96.0	80.9	72.8	94.3	93.7	90.9

155 and Auto-CoT (Zhang et al., 2023). The goal is to generate a reasoning path and answer a math word
 156 problem Q . For example, in Zero-Shot-CoT (Kojima et al., 2022), the prompt was “ Q : q . A : *Let’s*
 157 *think step by step.*” where q is the final question of Q . By adding the I³C instruction to the Zero-
 158 Shot-CoT method, denoted by Zero-Shot-CoT+I³C in our experiments, the prompt becomes “ I . Q :
 159 q . A : *Let’s think step by step*”. The full prompts in experiments can be found in Appendix A.4.

160 Finally, after the reasoning path is generated, we use prompt “*Therefore, the answer is*” to get the
 161 quantity prediction as the final answer for evaluation.

162 3.5 I³C-SELECT: SELECT CONFUSING PROBLEMS AS AUTOMATIC DEMONSTRATIONS

163 Fu et al. (2023) indicated that prompts with higher reasoning complexity achieve better performance
 164 on multi-step reasoning tasks. To further enhance the ability of LLMs to address the irrelevance of
 165 conditions, we develop a novel few-shot prompting method I³C-Select. As presented in § 3.2, it
 166 first calculates the confusion score of solved problems, defined in Eq.(2), and selects the K most
 167 confusing problems ($K = 8$ in our experiments). Next, it uses the most confusing problems and
 168 their reasoning paths as demonstrations, denoted by $\{Q_1^{(\text{demo})}, R_1^{(\text{demo})}; \dots; Q_K^{(\text{demo})}, R_K^{(\text{demo})}\}$.

169 I³C-Select puts the demonstrations after the I³C instruction to construct the full prompt. Specifically,
 170 the prompt is “ I . Q : $Q_1^{(\text{demo})}$ A : $R_1^{(\text{demo})}$ \dots Q : $Q_K^{(\text{demo})}$ A : $R_K^{(\text{demo})}$ Q : Q . A :”. With the prompt and
 171 the target problem Q , the LLMs generate a reasoning path for Q . Figure 1b illustrates the details.

172 4 EXPERIMENTS

173 4.1 EXPERIMENTAL SETUP

174 **Datasets.** We use six math word problem (MWP) datasets for evaluation. AddSub (Hosseini et al.,
 175 2014), SingleEq (Koncel-Kedziorski et al., 2015), SVAMP (Patel et al., 2021), and GSM8K (Cobbe
 176 et al., 2021) are classical MWP datasets in which part of the problem description contains irrelevant
 177 conditions. GSM-IC2-1K (Shi et al., 2023) and GSM-ICM-1K (Shi et al., 2023) are challenging
 178 datasets that require multi-step reasoning, and each problem description contains irrelevant condi-
 179 tions. More detailed dataset information can be found in Appendix A.1.

180 **Baselines.** We compare our proposed I³C-Select prompting method with two types of prompting
 181 baselines: (1) Zero-shot baselines. We include Zero-Shot-CoT (Kojima et al., 2022), PS (Wang et al.,
 182 2023a), Instruct-CoT (Shi et al., 2023), and Direct (Kojima et al., 2022). The Direct baseline uses

Table 2: Accuracy (%) on GSM-IC-2K dataset, broken down by the number of reasoning steps required in the standard answer. The GSM-IC-2K dataset is formed by merging the GSM-IC2-1K dataset and the GSM-ICM-1K dataset. The best performance for each dataset is shown in bold.

Method (text-davinci-003)	Accuracy by Steps (GSM-IC-2K)				
	2 Steps	3 Steps	4 Steps	≥ 5 Steps	All
Direct	22.8	15.3	5.7	4.9	15.9
Direct + I ³ C	82.6 (+59.8)	74.3 (+59.0)	66.8 (+61.1)	59.0 (+54.1)	74.8 (+58.9)
Zero-Shot-CoT	70.7	67.8	62.9	56.4	66.6
Zero-Shot-CoT + I ³ C	84.7 (+14.0)	75.9 (+8.1)	73.9 (+11.0)	64.5 (+8.1)	78.1 (+11.5)
PS	70.9	69.4	63.3	57.3	67.2
PS + I ³ C	84.8 (+13.9)	73.7 (+4.3)	71.7 (+8.4)	62.8 (+5.5)	77.1 (+9.9)
Instruct-CoT	82.4	68.1	65.7	59.0	73.4
Instruct-CoT + I ³ C	84.7 (+2.3)	78.3 (+10.2)	69.3 (+3.6)	65.4 (+6.4)	78.0 (+4.6)
Manual-CoT	73.9	68.1	52.3	59.3	67.3
Manual-CoT + I ³ C	82.0 (+8.1)	72.1 (+4.0)	64.3 (+12.0)	61.1 (+1.8)	74.1 (+6.8)
Auto-CoT	74.3	80.4	53.7	58.1	69.8
Auto-CoT + I ³ C	83.9 (+9.6)	73.2 (-7.2)	68.6 (+14.9)	62.5 (+4.4)	76.1 (+6.3)
I ³ C-Select (Ours)	93.7	93.3	90.1	89.0	92.3

183 the prompt “*The answer is*” to get the final answer. (2) Few-shot baselines. We include Manual-
 184 CoT (Wei et al., 2022) and Auto-CoT (Zhang et al., 2023). The demonstrations of the few-shot
 185 baselines are from their original papers. Details of all baselines are shown in Appendix A.2.

186 **Implementation.** We use GPT-3 (text-davinci-003) as the backend LLM, which is one of the most
 187 widely-used LLMs with public APIs¹. Following (Shi et al., 2023), we set the temperature to 0.7
 188 throughout our experiments. To evaluate the model performance, we follow (Chen et al., 2022b) to
 189 adopt accuracy as our evaluation metric. An answer is considered correct if and only if the absolute
 190 error between the answer and the gold answer is less than 1×10^{-5} . See Appendix A.3 for detail.

191 4.2 EXPERIMENTAL RESULTS

192 **Overall performance on MWP datasets.** As shown in Table 1, I³C-Select consistently outper-
 193 forms the baseline methods across all MWP datasets by a significant margin. Specifically, it im-
 194 proves the accuracy over Zero-Shot-CoT by at least +6.6 for all datasets, except for SingleEq,
 195 which has a +4.8 improvement. This exception can be attributed to the fact that the problems in
 196 SingleEq do not contain irrelevant conditions, and our proposed I³C-Select method mainly instructs
 197 LLMs to identify and ignore irrelevant conditions in the problem description. It is worth noting
 198 that even in the SingleEq dataset, using the most confusing problems and their reasoning paths as
 199 demonstrations effectively enhances MWP solving performance.

200 In comparison to the competitive zero-shot baseline, Instruct-CoT, the performance of I³C-Select re-
 201 mains impressive. It enhances the average accuracy by +11.0 across six MWP datasets, surpassing
 202 the Instruct-CoT prompting method. Furthermore, our analysis demonstrates that I³C-Select outper-
 203 forms few-shot baselines on all datasets. Specifically, when compared to the Auto-CoT prompting
 204 method, I³C-Select exhibits superior performance in GSM-ICM-1K, GSM-IC2-1K, and GSM8K,
 205 with improvements of +25.7, +19.4, and +13.9, respectively. These findings indicate that incorpo-
 206 rating more detailed instructions (e.g., I³C instruction) and the most confusing problems and their
 207 reasoning paths into the prompt can achieve better performance.

208 **Does adding the I³C instruction work?** As shown in Table 1, adding the I³C instruction to
 209 the CoT prompting methods significantly improves the MWP solving performance. Specifically,
 210 adding the I³C instruction to the Zero-Shot-CoT method (i.e., Zero-Shot-CoT+I³C) improves the
 211 average accuracy by +6.0 on six MWP datasets, compared to the original Zero-Shot-CoT prompting
 212 method. For GSM-IC2-1K and GSM-ICM-1K, which contain irrelevant conditions in each problem
 213 description, Zero-Shot-CoT+I³C improves the accuracy by +14.0 and +8.9, respectively. Even for

¹Public API available at <https://openai.com/api/>.

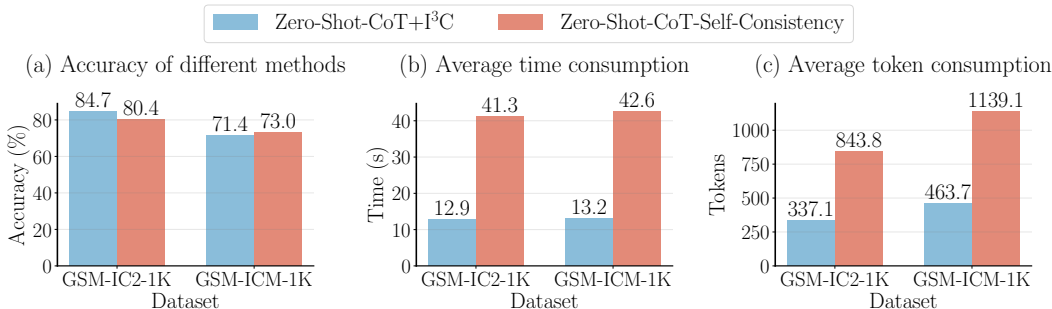


Figure 2: Performance comparison of Zero-Shot-CoT with I³C instruction (i.e., Zero-Shot-CoT+I³C) and Zero-Shot-CoT with self-consistency (i.e., Zero-Shot-CoT-Self-Consistency).

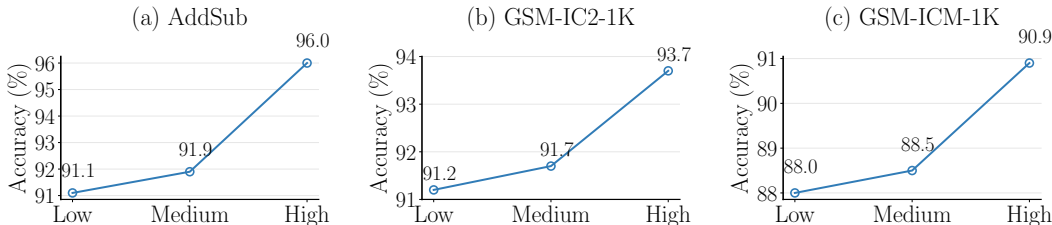


Figure 3: Comparison to other demonstration construction methods. “Low” indicates selecting eight problems with the lowest confusion scores. “Medium” indicates randomly selecting eight problems. “High” indicates selecting eight problems with the highest confusion scores.

214 prompting methods (e.g., Auto-CoT) that already achieve high accuracy on most MWP datasets, the
 215 addition of the I³C instruction (i.e., Auto-CoT+I³C) still leads to significant improvements. Auto-
 216 CoT+I³C improves accuracy by +9.6 on GSM-IC2-1K, and +3.0 on GSM8K.

217 **Does I³C instruction work for complex problems?** We analyze the breakdown accuracies for
 218 problems with respect to the reasoning steps² in Table 2. The GSM-IC-2K dataset is formed by
 219 merging the GSM-IC2-1K and GSM-ICM-1K datasets. Each problem in GSM-IC-2K contains
 220 irrelevant conditions and requires multiple steps to solve. Prompts with I³C instruction outperform
 221 baseline methods in solving problems that require at least 4 steps. Moreover, compared to Manual-
 222 CoT, I³C-Select significantly improves the performance on GSM-IC-2K: from 67.3 to 92.3. These
 223 results indicate that adding I³C instruction to the prompt can effectively solve complex problems.

224 **Efficiency and effectiveness of I³C instruction.** Self-consistency (Wang et al., 2023b) is the
 225 process of repeatedly solving a problem M times and using a majority vote strategy to determine
 226 the most consistent answer as the final answer. We evaluate the performance of Zero-Shot-CoT
 227 with self-consistency (i.e., Zero-Shot-CoT-Self-Consistency) on the GSM-IC2-1K and GSM-ICM-
 228 1K datasets. Following (Wang et al., 2023a), we set M to 10. As shown in Figure 2, adding the
 229 I³C instruction to Zero-Shot-CoT (i.e., Zero-Shot-CoT+I³C) consumes much fewer computational
 230 resources compared to Zero-Shot-CoT-Self-Consistency, while maintaining comparable accuracy.

231 4.3 ABLATION STUDIES

232 **How does demonstration construction affect I³C-Select?** In I³C-Select, we select the K most
 233 confusing problems and their reasoning paths as demonstrations and named this demonstration con-
 234 struction method “High”. To verify the effectiveness of the demonstration construction method, we
 235 also consider: (1) “Low”, where we select the K problems with the lowest confusion scores and
 236 their reasoning paths as demonstrations, and (2) “Medium”, where we randomly select K problems
 237 and their reasoning paths as demonstrations. For a fair comparison, we set K to 8 throughout our

²The number of reasoning steps of a problem is given by the number of sentences in its standard answer. (Cobbe et al., 2021)

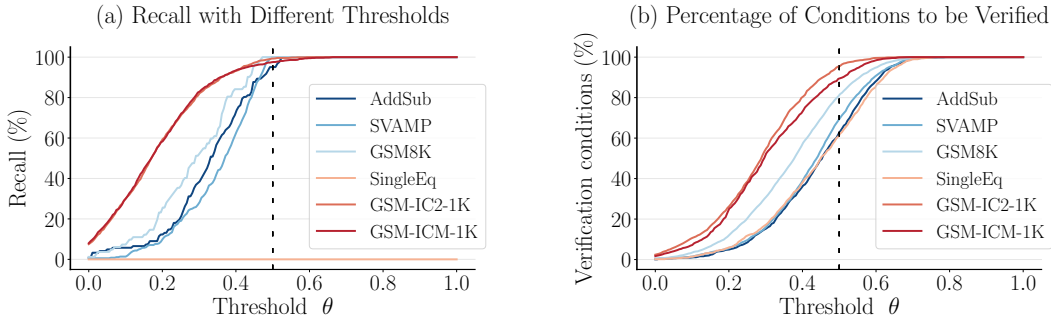


Figure 4: Hyperparameter analysis. (a) As the threshold increases, the recall scores of identified irrelevant condition candidates first increase and then remain unchanged for all datasets except SingleEq. (b) As the threshold increases, the percentage of conditions to be verified first increases and then remains unchanged for all datasets.

Table 3: Accuracy (%) when solving MWP’s using I³C-Select with different LLMs.

Method	LLM	AddSub	SingleEq	GSM-IC2-1K	GSM-ICM-1K
I ³ C-Select	text-davinci-002	80.0	84.8	87.2	87.3
	text-davinci-003	96.0(+16.0)	94.3(+9.5)	93.7(+6.5)	90.9(+3.6)

238 experiments. As shown in Figure 3, selecting more confusing problems and their reasoning paths as
 239 demonstrations can effectively improve the model’s performance.

240 **How does LLM selection affect I³C-Select?** Table 3 shows that I³C-Select works better when
 241 the LLM is more powerful. Specifically, on the AddSub dataset, the text-davinci-003 model demon-
 242 strates a +16.0 increase in accuracy compared to the text-davinci-002 model. Similarly, on the
 243 GSM-IC2-1K dataset, using the text-davinci-003 model leads to a +6.5 improvement in accuracy
 244 over the text-davinci-002 model. Notably, the text-davinci-002 model is fine-tuned using supervised
 245 instruction tuning, while the text-davinci-003 is fine-tuned with reinforcement learning (Zheng et al.,
 246 2023). The improved performance with text-davinci-003 can be attributed to its enhanced power,
 247 making it better at understanding and employing the given prompt.

248 **Instructing to ignore irrelevant conditions vs. refining problems to eliminate irrelevant con-**
 249 **ditions.** In Zero-Shot-CoT+I³C, we use I³C instruction to instruct LLMs to identify and ignore
 250 irrelevant conditions in the MWP solving process. In addition, we can refine the given problem to
 251 eliminate irrelevant conditions based on the verification outputs generated in § 3.3, and solve the
 252 refined problem using the Zero-Shot-CoT method (i.e., Zero-Shot-CoT+Refine). As shown in Table
 253 4, Zero-Shot-CoT+Refine (87.6, 77.4, and 64.8) substantially outperforms Zero-Shot-CoT (84.8,
 254 70.7, and 62.5) on AddSub, GSM-IC2-1K, and GSM-ICM-1K, respectively. This highlights that
 255 the generated verification outputs can explicitly identify irrelevant conditions in the problem de-
 256 scription. Furthermore, Zero-Shot-CoT+I³C consistently outperforms Zero-Shot-CoT+Refine. This
 257 is mainly because the identified irrelevant conditions may contain some useful conditions. When we
 258 refine the given problem, we may eliminate some useful conditions, resulting in an incorrect answer.
 259 Instructing the LLM to ignore irrelevant conditions can effectively alleviate the problem of losing
 260 useful conditions during problem refinement. Case studies are provided in Appendix A.5.

261 **Hyperparameter setup for I³C instruction.** To compare the identification accuracy of irrelevant
 262 conditions at different thresholds θ , we employ the recall metric. Recall is computed by dividing
 263 the number of correctly identified irrelevant conditions by the total number of irrelevant conditions.
 264 As shown in Figure 4a, as the threshold increases, the recall score for the irrelevant condition iden-
 265 tification first increases and then remains unchanged in all datasets except SingleEq. The exception
 266 is due to the problems in SingleEq not containing irrelevant conditions, the recall score remains 0
 267 regardless of the threshold. Notably, when the threshold is 0.5, we can accurately identify 98.6%
 268 of irrelevant conditions. Furthermore, as shown in Figure 4b, the percentage of conditions to be
 269 verified initially increases and then stabilizes across all datasets as the threshold increases. The per-

Table 4: Accuracy (%) comparison of different methods that help LLMs ignore irrelevant conditions.

Method (text-davinci-003)	Dataset		
	AddSub	GSM-IC2-1K	GSM-ICM-1K
Zero-Shot-CoT	84.8	70.7	62.5
Zero-Shot-CoT + Refine	87.6	77.4	64.8
Zero-Shot-CoT + I ³ C	91.7	84.7	71.4

Table 5: Case study of answers generated by different LLM methods. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color. The gold answer is indicated in bold.

Problem	Granger went to the grocery store. He saw that the Spam is \$3 per can, the peanut butter is \$5 per jar, and the bread is \$2 per loaf. Jack owns 5 bottles of water with each worth \$2. If Granger bought 12 cans of spam, 3 jars of peanut butter, and 4 loaves of bread, how much is the total amount he paid?
Gold Answer	$3 \times 12 + 5 \times 3 + 2 \times 4 = \mathbf{59}$
Auto-CoT	Granger bought 12 cans of spam. $12 * \$3 = \36 . Granger bought 3 jars of peanut butter. $3 * \$5 = \15 . Granger bought 4 loaves of bread. $4 * \$2 = \8 . Jack owns 5 bottles of water. $5 * \$2 = \10 . The total amount he paid is $\$36 + \$15 + \$8 + \$10 = \$69$. The answer is 69 . ✗
Instruct-CoT	Granger bought 12 cans of Spam, so that's $12 \times \$3 = \36 . He bought 3 jars of peanut butter, so that's $3 \times \$5 = \15 . He bought 4 loaves of bread, so that's $4 \times \$2 = \8 . Jack owned 5 bottles of water, so that's $5 \times \$2 = \10 . Adding them all up, the total amount Granger paid is $\$36 + \$15 + \$8 + \$10 = \mathbf{\$69}$. ✗
I³C-Select	First, calculate the cost for each item that Granger bought: 12 cans of Spam = $12 \times \$3 = \36 ; 3 jars of peanut butter = $3 \times \$5 = \15 ; 4 loaves of bread = $4 \times \$2 = \8 . Then, calculate the total cost of all the items: $\$36 + \$15 + \$8 = \mathbf{\$59}$. ✓

270 centage of conditions to be verified is calculated by dividing the number of conditions in the set of
 271 irrelevant condition candidates by the total number of conditions. When the threshold is 0.5, the
 272 LLM is required to verify 66.6% of conditions on average. Overall, we set the threshold θ to 0.5 to
 273 achieve a trade-off between efficiency and effectiveness.

274 4.4 CASE STUDY

275 We present a real case study from the GSM-ICM-1K dataset in Table 5. It is evident that, apart from
 276 I³C-Select, the other methods fail to provide accurate answers to the given problem. Instruct-CoT
 277 and Auto-CoT produce incorrect answers due to the incorporation of irrelevant conditions in the
 278 MWP solving process. In contrast, I³C-Select explicitly identifies and ignores irrelevant conditions
 279 in the MWP solving process. Additional case studies can be found in Appendix A.6.

280 5 CONCLUSION

281 In this paper, we present a novel approach named I³C to instruct LLMs to explicitly identify and
 282 ignore irrelevant conditions in the mathematical reasoning process. It first identifies a set of irrele-
 283 vant condition candidates that have a weak semantic relevance with the question, and then prompts a
 284 LLM to generate verification outputs to verify if candidates are indeed irrelevant. By incorporating
 285 all the verification outputs, we obtained the I³C instruction. The I³C instruction is a plug-and-play
 286 module that can be added to any CoT prompting methods to help LLMs avoid confusion and im-
 287 prove their generated reasoning paths. Moreover, we present a novel few-shot prompting method,
 288 I³C-Select, which selects the most confusing problems and their reasoning paths as demonstrations,
 289 and adds the I³C instruction before the demonstrations to construct the prompt. Extensive experi-
 290 ments demonstrate that adding the I³C instruction to CoT prompting methods effectively improves
 291 MWP solving performance, achieving new state-of-the-art performance on all MWP datasets.

292 REFERENCES

- 293 Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*, 2022a.
- 294
295
- 296 Wenhui Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2022b.
- 297
- 298 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. *CoRR*, abs/2110.14168, 2021. URL <https://arxiv.org/abs/2110.14168>.
- 299
300
301
- 302 Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. Complexity-based prompting for multi-step reasoning, 2023.
- 303
- 304 Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.552. URL <https://aclanthology.org/2021.emnlp-main.552>.
- 305
306
307
308
- 309 Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. Learning to solve arithmetic word problems with verb categorization. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 523–533, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1058. URL <https://aclanthology.org/D14-1058>.
- 310
311
312
313
- 314 Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1215. URL <https://aclanthology.org/D17-1215>.
- 315
316
317
- 318 Ronald T. Kellogg. Sage Publications, Inc, 2016. ISBN 978-1-4833-4758-5.
- 319
- 320 Junhan Kim, Kyuhong Shim, and Byonghyo Shim. Semantic feature extraction for generalized zero-shot learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 1166–1173, 2022. doi: <https://doi.org/10.1609/aaai.v36i1.20002>.
- 321
- 322 Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 22199–22213. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/8bb0d291acd4acf06ef112099c16f326-Paper-Conference.pdf.
- 323
324
325
326
327
- 328 Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597, 2015. doi: 10.1162/tacl.a.00160. URL <https://aclanthology.org/Q15-1042>.
- 329
330
331
- 332 Hongchao Li, Chenglong Li, Aihua Zheng, Jin Tang, and Bin Luo. Mskat: Multi-scale knowledge-aware transformer for vehicle re-identification. *IEEE Transactions on Intelligent Transportation Systems*, 23(10):19557–19568, 2022. doi: 10.1109/TITS.2022.3166463.
- 333
334
- 335 Zhenwen Liang, Jipeng Zhang, Lei Wang, Wei Qin, Yunshi Lan, Jie Shao, and Xiangliang Zhang. MWP-BERT: Numeracy-augmented pre-training for math word problem solving. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 997–1009, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.74. URL <https://aclanthology.org/2022.findings-naacl.74>.
- 336
337
338
339
- 340 Zhenwen Liang, Jipeng Zhang, Lei Wang, Yan Wang, Jie Shao, and Xiangliang Zhang. Generalizing math word problem solvers via solution diversification. In *AAAI*, 2023.
- 341

- 342 Arindam Mitra and Chitta Baral. Learning to use formulas to solve simple arithmetic problems.
343 In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*
344 (*Volume 1: Long Papers*), pp. 2144–2153, Berlin, Germany, August 2016. Association for Com-
345 putational Linguistics. doi: 10.18653/v1/P16-1202. URL [https://aclanthology.org/
346 P16-1202](https://aclanthology.org/P16-1202).
- 347 Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are NLP models really able to solve simple
348 math word problems? In *Proceedings of the 2021 Conference of the North American Chapter*
349 *of the Association for Computational Linguistics: Human Language Technologies*, pp. 2080–
350 2094, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.
351 naacl-main.168. URL <https://aclanthology.org/2021.naacl-main.168>.
- 352 Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015*
353 *Conference on Empirical Methods in Natural Language Processing*, pp. 1743–1752, Lisbon, Por-
354 tugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1202.
355 URL <https://aclanthology.org/D15-1202>.
- 356 Subhro Roy, Tim Vieira, and Dan Roth. Reasoning about quantities in natural language. *Transac-*
357 *tions of the Association for Computational Linguistics*, 3:1–13, 2015. doi: 10.1162/tacl.a.00118.
358 URL <https://aclanthology.org/Q15-1001>.
- 359 Jianhao Shen, Yichun Yin, Lin Li, Lifeng Shang, Xin Jiang, Ming Zhang, and Qun Liu. Generate &
360 rank: A multi-task framework for math word problems. In *Findings of the Association for Compu-*
361 *tational Linguistics: EMNLP 2021*, pp. 2269–2279, Punta Cana, Dominican Republic, November
362 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.195.
363 URL <https://aclanthology.org/2021.findings-emnlp.195>.
- 364 Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H. Chi, Nathanael
365 Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context.
366 In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and
367 Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*,
368 volume 202 of *Proceedings of Machine Learning Research*, pp. 31210–31227. PMLR, 23–29 Jul
369 2023. URL <https://proceedings.mlr.press/v202/shi23a.html>.
- 370 Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won
371 Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metz-
372 zler. UL2: Unifying language learning paradigms. In *The Eleventh International Confer-*
373 *ence on Learning Representations*, 2023. URL [https://openreview.net/forum?id=
374 6ruVLB727MC](https://openreview.net/forum?id=6ruVLB727MC).
- 375 Lei Wang, Dongxiang Zhang, Jipeng Zhang, Xing Xu, Lianli Gao, Bing Tian Dai, and Heng Tao
376 Shen. Template-based math word problem solvers with recursive neural networks. In *Proceed-*
377 *ings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative*
378 *Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational*
379 *Advances in Artificial Intelligence*, AAAI’19/IAAI’19/EAAI’19. AAAI Press, 2019. ISBN 978-
380 1-57735-809-1. doi: 10.1609/aaai.v33i01.33017144. URL [https://doi.org/10.1609/
aaai.v33i01.33017144](https://doi.org/10.1609/
381 aaai.v33i01.33017144).
- 382 Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim.
383 Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language
384 models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguis-*
385 *tics (Volume 1: Long Papers)*, pp. 2609–2634, Toronto, Canada, July 2023a. Association for Com-
386 putational Linguistics. URL <https://aclanthology.org/2023.acl-long.147>.
- 387 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha
388 Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language
389 models. In *The Eleventh International Conference on Learning Representations*, 2023b. URL
390 <https://openreview.net/forum?id=1PL1NIMMrw>.
- 391 Yan Wang, Xiaojiang Liu, and Shuming Shi. Deep neural solver for math word problems. In
392 *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp.
393 845–854, Copenhagen, Denmark, September 2017. Association for Computational Linguistics.
394 doi: 10.18653/v1/D17-1088. URL <https://aclanthology.org/D17-1088>.

- 395 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V
396 Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models.
397 In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in*
398 *Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc.,
399 2022. URL [https://proceedings.neurips.cc/paper_files/paper/2022/
400 file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf).
- 401 Mingyi Yang, Luis Herranz, Fei Yang, Luka Murn, Marc Gorriz Blanch, Shuai Wan, Fuzheng Yang,
402 and Marta Mrak. Semantic preprocessor for image compression for machines. In *ICASSP 2023 -*
403 *2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp.
404 1–5, 2023. doi: 10.1109/ICASSP49357.2023.10096472.
- 405 Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. Automatic chain of thought prompting in
406 large language models. In *The Eleventh International Conference on Learning Representations*
407 *(ICLR 2023)*, 2023.
- 408 Chuanyang Zheng, Zhengying Liu, Enze Xie, Zhenguo Li, and Yu Li. Progressive-hint prompting
409 improves reasoning in large language models, 2023.
- 410 Lipu Zhou, Shuaixiang Dai, and Liwei Chen. Learn to solve algebra word problems using quadratic
411 programming. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language*
412 *Processing*, pp. 817–822, Lisbon, Portugal, September 2015. Association for Computational Lin-
413 guistics. doi: 10.18653/v1/D15-1096. URL <https://aclanthology.org/D15-1096>.

414 A APPENDIX

415 A.1 DATASETS

416 We use six math word problem datasets for assessing prompting method quality. The statistics of
417 the datasets are shown in Table 6. We give a brief description of the datasets used below:

- 418 – SingleEq (Koncel-Kedziorski et al., 2015) contains a set of grade-school algebra word prob-
419 lems. Every problem may involve multiple math operations including multiplication, division,
420 subtraction, and addition.
- 421 – AddSub (Hosseini et al., 2014) consists of math word problems on addition and subtraction for
422 third, fourth, and fifth graders.
- 423 – SVAMP (Patel et al., 2021) consists of one-unknown math word problems which can be solved
424 by expressions requiring no more than two operators.
- 425 – GSM8K (Cobbe et al., 2021) consists of high quality grade school math word problems created
426 by human problem writers. These problems take between 2 and 8 steps to solve, and solutions
427 primarily involve performing a sequence of elementary calculations using basic arithmetic oper-
428 ations to reach the final answer.
- 429 – GSM-IC (Shi et al., 2023) is an arithmetic reasoning dataset with irrelevant conditions in the
430 problem description. It is divided into two splits: GSM-IC2, consisting of problems requiring
431 two steps to solve, and GSM-ICM, consisting of problems requiring more than two steps to
432 solve. Being mindful of the experiment costs, we uniformly sample 1,000 examples from the
433 GSM-IC2 dataset (denoted by GSM-IC2-1K) and 1,000 examples from the GSM-ICM dataset
434 (denoted by GSM-ICM-1K) for evaluation and analysis purposes throughout this paper.

435 A.2 BASELINES

436 As we study how to prompt large language models to solve math word problems, we employ seven
437 prompting baselines. We give a brief description of the baselines used below:

- 438 – Direct (Kojima et al., 2022) is a baseline that utilizes the symbolic reasoning ability of large
439 language models. By simply adding the sentence “*The answer is*” after the problem of interest,
440 which instructs the large language model to generate the answer to the problem.
- 441 – Zero-Shot-CoT (Kojima et al., 2022) is a Chain-of-Thought prompting method. By adding
442 “*Let’s think step by step*” to the problem to elicit the large language model to generate reasoning
443 path leading to the final answer.
- 444 – Plan-and-Solve (PS) (Wang et al., 2023a) replaces the sentence “*Let’s think step by step*” with
445 “*Let’s first understand the problem and devise a plan to solve the problem. Then let’s carry out
446 the plan and solve the problem step by step*” to address the missing step issue in Zero-Shot-CoT.
- 447 – Instruct-CoT (Shi et al., 2023) adds the sentence “*Feel free to ignore irrelevant conditions in
448 the problem description.*” before the problem of interest, which instructs the large language
449 model to ignore irrelevant information in the problem description.
- 450 – Manual-CoT (Wei et al., 2022) is a few-shot prompting method. By representing manual de-
451 signed demonstrations that solve the corresponding problems with intermediate reasoning steps
452 in the prompts, Manual-CoT elicits multi-step reasoning ability of large language models.
- 453 – Auto-CoT (Zhang et al., 2023) automatically constructs demonstrations with questions and
454 reasoning paths to eliminate manual designs in Manual-CoT.

455 A.3 METRICS

We use accuracy to evaluate the performance of different prompting methods. Since large language models cannot perform the computation precisely (especially with high-precision floats), we consider an answer to be correct if and only if the absolute error between the answer and the gold answer

Table 6: Dataset description. The last column indicates the percentage of problems with irrelevant conditions in the problem description.

Dataset	Number of Problems	Average Words	Irrelevant Condition
SingleEq	508	27.4	0.0%
AddSub	395	31.5	30.9%
SVAMP	1,000	31.8	36.7%
GSM8K	1,319	46.9	6.2%
GSM-IC2-1K	1,000	41.8	100.0%
GSM-ICM-1K	1,000	61.4	100.0%

Table 7: Accuracy (%) comparison on six MWP datasets. I³C indicates that instructs LLMs to identify and ignore irrelevant conditions. Adding the I³C instruction to CoT prompting methods effectively improves performance. Selecting the most confusing problems and their generated reasoning paths as demonstrations for few-shot learning (i.e., I³C-Select) achieves state-of-the-art performance on all six MWP datasets. The best performance for each dataset is shown in bold.

Method (UL2-20B)	Dataset					
	AddSub	SVAMP	GSM8K	SingleEq	GSM-IC2-1K	GSM-ICM-1K
Direct	28.6	16.9	5.0	21.7	12.9	9.5
Direct + I ³ C	33.9(+5.3)	27.8(+10.9)	9.8(+4.8)	32.7(+11.0)	21.3(+8.4)	13.2(+3.7)
Zero-Shot-CoT	32.9	29.5	22.7	38.8	29.6	25.5
Zero-Shot-CoT + I ³ C	36.7(+3.8)	30.5(+1.0)	22.7(+0.0)	40.0(+1.2)	40.6(+11.0)	27.6(+2.1)
PS	30.0	26.7	21.2	36.6	27.4	24.9
PS + I ³ C	31.9(+1.9)	28.4(+1.7)	21.3(+0.1)	40.0(+3.4)	32.4(+5.0)	26.0(+1.1)
Instruct-CoT	34.7	31.2	23.5	40.0	33.8	26.4
Instruct-CoT + I ³ C	35.4(+0.7)	31.5(+0.3)	21.2(-2.3)	41.1(+1.1)	40.0(+6.2)	28.6(+2.2)
Manual-CoT	34.9	31.7	25.2	43.3	35.4	28.0
Manual-CoT + I ³ C	39.0(+4.1)	28.1(-3.6)	22.2(-3.0)	42.9(-0.4)	43.0(+7.6)	28.5(+0.5)
Auto-CoT	36.7	31.9	24.5	41.9	35.0	29.4
Auto-CoT + I ³ C	39.5(+2.8)	28.7(-3.2)	24.7(+0.2)	43.6(+1.7)	41.1(+6.1)	30.1(+0.7)
I ³ C-Select (Ours)	39.7	34.6	27.5	44.1	46.0	35.9

is less than 1×10^{-5} . Let \mathcal{P} be a set of problems, the accuracy of the prompting method is

$$\text{Accuracy} = \frac{1}{|\mathcal{P}|} \sum_{Q \in \mathcal{P}} \mathbb{1} \left(a^{(\text{final})}, a^{(\text{gold})} \right)$$

$$\mathbb{1} \left(a^{(\text{final})}, a^{(\text{gold})} \right) = \begin{cases} 1, & \text{if } \text{Abs} \left(a^{(\text{final})} - a^{(\text{gold})} \right) < 1 \times 10^{-5} \\ 0, & \text{if } \text{Abs} \left(a^{(\text{final})} - a^{(\text{gold})} \right) \geq 1 \times 10^{-5} \end{cases}$$

456 where $a^{(\text{gold})}$ is the gold answer to question Q , $a^{(\text{final})}$ is the model-generated answer to question Q ,
457 and $\text{Abs}(\cdot)$ is the absolute value function.

458 A.4 FULL PROMPTS IN EXPERIMENTS

459 We list the prompts for all experiments in Table 8.

460 A.5 ADDITIONAL EXPERIMENTAL RESULTS

461 **Does I³C instruction work with weaker LMs?** In all our experiments in § 4, we use GPT-3
462 (text-davinci-003) as the backend LLM, but can I³C instruction work with weaker LMs? We com-
463 pare CoT prompting methods with adding the I³C instruction to CoT prompting methods when
464 use the UL2-20B (Tay et al., 2023) as backend LM. Note that UL2-20B is a weaker LMs with 20
465 billion parameters, but GPT3 has 175 billion parameters. As shown in Table 7, even though the
466 absolute accuracies of UL2-20B are lower, adding the I³C instruction to CoT prompting methods
467 effectively improves MWP solving performance, and I³C-Select achieves consistent performance
468 improvements on MWP datasets. This shows that I³C instruction can work with weaker LMs.

Table 8: All prompts used in experiments. Q represents the problem to be solved. I represents the I³C instruction that instructs LLMs to identify and ignore irrelevant conditions in the problem description. The demonstrations of Manual-CoT is from its original paper (Wei et al., 2022).

Method	Prompt
Direct	$Q: Q$
	$A: \text{The answer is}$
Direct + I ³ C	I
	$Q: Q$ $A: \text{The answer is}$
Zero-Shot-CoT	$Q: Q$
	$A: \text{Let's think step by step}$
Zero-Shot-CoT + I ³ C	I
	$Q: Q$ $A: \text{Let's think step by step}$
PS	$Q: Q$
	$A: \text{Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step}$
PS + I ³ C	I
	$Q: Q$ $A: \text{Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan and solve the problem step by step}$
Instruct-CoT	I
	$Q: Q$ $A: \text{Let's think step by step}$
Instruct-CoT + I ³ C	I
	$Q: Q$ $A: \text{Let's think step by step}$
Manual-CoT	{hand-crafted demonstrations}
	$Q: Q$ $A:$
Manual-CoT + I ³ C	I
	{hand-crafted demonstrations} $Q: Q$ $A:$
Auto-CoT	{automatically constructed demonstrations}
	$Q: Q$ $A:$
Auto-CoT + I ³ C	I
	{automatically constructed demonstrations} $Q: Q$ $A:$
I ³ C-Select (Ours)	I
	{the most confusing problems and their reasoning paths} $Q: Q$ $A:$

469 **Instructing to ignore irrelevant conditions vs. refining problems to eliminate irrelevant con-**
470 **ditions.** In Zero-Shot-CoT+I³C, we use I³C instruction to instruct LLMs to identify and ignore
471 irrelevant conditions in the MWP solving process. In addition, we can refine the given problem to
472 eliminate irrelevant conditions based on the verification outputs generated in § 3.3, and solve the
473 refined problem using the Zero-Shot-CoT method (i.e., Zero-Shot-CoT+Refine). For example, as
474 shown in Table 9, the condition “On Friday, he only works from 8am to 11am.” and the condition
475 “Last week, Tom repaired 10 more umbrellas than the cobbler.” are identified as the irrelevant con-
476 ditions. By eliminating these identified irrelevant conditions, we get the refined problem “A cobbler
477 can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours
478 each day. How many pairs of shoes can the cobbler mend in a week?”. Obviously, in this case,
479 we incorrectly identified the condition “On Friday, he only works from 8am to 11am.” as an irrel-
480 evant condition. Eliminating this condition would result in losing useful conditions in the problem
481 refinement process, resulting in an incorrect answer. In contrast, instructing the LLM to ignore ir-

Table 9: Example outputs by Zero-Shot-CoT + Refine and Zero-Shot-CoT + I³C for GSM-ICM-1K. Irrelevant conditions in the problem description is highlighted with red color.

Problem	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day, and on Friday, he only works from 8am to 11am. Last week, Tom repaired 10 more umbrellas than the cobbler. How many pairs of shoes can the cobbler mend in a week?	
Gold Answer	$(4 \times 8 + 11 - 8) \times 3 = 105$	
Verification Outputs	<p>■ Yes, the condition A cobbler can mend 3 pairs of shoes in an hour is relevant to the calculation process of the problem How many pairs of shoes can the cobbler mend in a week? Without the condition, it would be impossible to calculate the number of pairs of shoes the cobbler can mend in a week.</p> <p>■ The condition on Friday, he only works from 8am to 11am is not relevant to the calculation process.</p> <p>■ The condition Last week, Tom repaired 10 more umbrellas than the cobbler. is not relevant to the calculation process of the problem.</p>	
Refined Problem	A cobbler can mend 3 pairs of shoes in an hour. From Monday to Thursday, the cobbler works for 8 hours each day. How many pairs of shoes can the cobbler mend in a week?	
Reasoning Path	Zero-Shot-CoT + Refine	Zero-Shot-CoT + I³C
	First, we need to calculate how many pairs of shoes the cobbler can mend in 4 days (Monday to Thursday). Since he works 8 hours a day, and can mend 3 pairs of shoes per hour, he can mend $8 \times 3 = 24$ pairs of shoes in 4 days. This gives us 24 pairs of shoes in a week.	Step 1: Calculate the number of hours the cobbler works in a week. From Monday to Thursday, the cobbler works for 8 hours each day. That's $8 \times 4 = 32$ hours. On Friday, he only works from 8am to 11am. That's 3 hours. So in a week, the cobbler works $32 + 3 = 35$ hours. Step 2: Calculate the number of pairs of shoes the cobbler can mend in a week. The cobbler can mend 3 pairs of shoes in an hour. So in 35 hours, the cobbler can mend $35 \times 3 = 105$ pairs of shoes. Step 3: Determine if the condition is relevant to the calculation process. The fact that Tom repaired 10 more umbrellas than the cobbler does not affect the calculation process. Therefore, the cobbler can mend 105 pairs of shoes in a week.
Final Answer	24 ✗	105 ✓

482 relevant conditions can effectively alleviate the problem of losing useful conditions during problem
483 refinement, and can effectively enhance the MWP solving performance.

484 **I³C Instruction Improves Robustness to Irrelevant Conditions.** We present case studies in Ta-
485 bles 10 to 15. It is evident that I³C instructs LLMs to explicitly identify and ignore irrelevant
486 conditions in the mathematical reasoning process. Adding I³C instruction to the CoT prompting
487 methods effectively helps LLMs avoid confusion and improve reasoning paths.

488 A.6 SAMPLE PREDICTIONS ON MWP DATASETS

Table 10: Example outputs by Zero-Shot-CoT, Zero-Shot-CoT + I³C and I³C-Select. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color.

Problem	Megan is an actress. She was the lead actress in 80% of her work. In total, Megan participated in 100 plays. Tom bought the tickets for 300 plays. How many times Megan was not the lead actress?
Gold Answer	$100 \times (1 - 80\%) = 20$
Verification Outputs	<p>■ Megan participated in 100 plays. That means she was the lead actress in 80 of them. That leaves 20 plays where she was not the lead actress. Tom bought tickets to 300 plays. That means Megan was the lead actress in 240 of them, and she was not the lead actress in 60 of them. So, the answer is 60 times. The condition She was the lead actress in 80% of her work is relevant to the calculation process, as it tells us that 80 out of 100 plays she was the lead actress.</p> <p>■ Megan was the lead actress in 80% of her plays, so she was the lead actress in 80 plays. Since she participated in a total of 100 plays, that means she was not the lead actress in the remaining 20 plays. The condition that Tom bought tickets for 300 plays is irrelevant to the calculation process.</p>
Reasoning Path	<p>Zero-Shot-CoT Megan was the lead actress in 80 plays. $100 - 80 = 20$ Tom bought the tickets for 300 plays. Megan was not the lead actress in the remaining 200 plays. ✗</p> <p>Zero-Shot-CoT + I³C Megan was the lead actress in 80% of her work, which means she was the lead actress in 80 plays. That leaves 20 plays in which she was not the lead actress. Tom bought tickets for 300 plays, but that information does not affect the calculation of how many times Megan was not the lead actress. Therefore, the answer is 20 times. ✓</p> <p>I³C-Select Megan was the lead actress in 80% of her work, so she was the lead actress in 80 plays. That leaves 20 plays in which she was not the lead actress. Tom bought the tickets for 300 plays, but that information does not affect the calculation of how many times Megan was not the lead actress. Therefore, the answer is 20 times. ✓</p>

Table 11: Example outputs by Direct, Direct + I³C and I³C-Select. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color.

Problem	Sally eats 2 sandwiches on Saturday and 1 sandwich on Sunday. Jack bought 2 more pieces of bread than Sally yesterday. If each sandwich uses 2 pieces of bread, how many pieces of bread does Sally eat across Saturday and Sunday?
Gold Answer	$(2 + 1) \times 2 = 6$
Verification Outputs	<p>■ Sally eats 2 sandwiches on Saturday, so she has eaten 4 pieces of bread. On Sunday, she eats 1 sandwich, so she has eaten 2 more pieces of bread. Jack bought 2 more pieces of bread than Sally, so Sally has eaten 6 pieces of bread across Saturday and Sunday. Therefore, the answer is 6 pieces of bread. The condition of 1 sandwich on Sunday is relevant to the calculation process, since it tells us how many pieces of bread Sally has eaten on Sunday.</p> <p>■ Sally ate 2 sandwiches on Saturday, so she used 4 pieces of bread on Saturday. On Sunday, Sally ate 1 sandwich, so she used 2 pieces of bread on Sunday. Therefore, Sally ate 6 pieces of bread across Saturday and Sunday. The condition Jack bought 2 more pieces of bread than Sally is not relevant to the calculation process, as it does not affect the number of pieces of bread Sally ate across Saturday and Sunday.</p>
Reasoning Path	<p>Direct Sally eats a total of 4 pieces of bread across Saturday and Sunday. ✗</p> <p>Direct + I³C 6 pieces of bread. Sally ate 2 sandwiches on Saturday, which used 4 pieces of bread, and 1 sandwich on Sunday, which used 2 pieces of bread, so she ate 6 pieces of bread across Saturday and Sunday. The condition that Jack bought 2 more pieces of bread than Sally is not relevant to the calculation process. ✓</p> <p>I³C-Select Sally ate 2 sandwiches on Saturday, so she used 4 pieces of bread on Saturday. On Sunday, Sally ate 1 sandwich, so she used 2 pieces of bread on Sunday. Therefore, Sally ate 6 pieces of bread across Saturday and Sunday. The condition Jack bought 2 more pieces of bread than Sally is not relevant to the calculation process, as it does not affect the number of pieces of bread Sally ate across Saturday and Sunday. ✓</p>

Table 12: Example outputs by PS, PS + I³C and I³C-Select. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color.

Problem	Holden’s current master bedroom is 309 sq ft and his master bath is 150 sq ft. The basketball field of Tom is 200 sq ft larger than Holden’s bathroom. If Holden wants to add a home office/personal gym divided space that is twice as large as his bedroom and bathroom, how much sq ft will this new room have?
Gold Answer	$(309 + 105) \times 2 = 918$
Verification Outputs	<p>■ The master bedroom is 309 sq ft and the master bathroom is 150 sq ft. Therefore, the combined area of the bedroom and bathroom is 459 sq ft. If this space is to be doubled for the new home office/personal gym, then the new room will have 918 sq ft. The condition Holden’s current master bedroom is 309 sq ft is relevant to the problem, as it is used to determine the combined area of the bedroom and bathroom (459 sq ft). This combined area is then doubled to get the total area of the new room (918 sq ft).</p> <p>■ Holden’s bedroom is 309 sq ft. Holden’s bathroom is 150 sq ft. The total sq ft of Holden’s bedroom and bathroom is 459 sq ft. We need to find a space that is twice as large as Holden’s bedroom and bathroom combined. Therefore, the new room will be $2 \times 459 \text{ sq ft} = 918 \text{ sq ft}$. The basketball field of Tom is 200 sq ft larger than Holden’s bathroom so it is not relevant to the calculation process of this problem.</p>
Reasoning Path	<p>PS Plan: 1. Find the combined sq ft of the bedroom and bathroom. 2. Add 200 sq ft to the combined sq ft from step 1. 3. Multiply the combined sq ft from step 2 by 2. Solution: 1. 309 sq ft (bedroom) + 150 sq ft (bathroom) = 459 sq ft 2. 459 sq ft + 200 sq ft = 659 sq ft 3. 659 sq ft x 2 = 1318 sq ft Therefore, the new room will have 1318 sq ft. ✗</p> <p>PS + I³C First, we need to calculate the combined area of Holden’s bedroom and bathroom, which is equal to 309 sq ft (bedroom) + 150 sq ft (bathroom) = 459 sq ft. Next, we need to double this combined area to get the total area of the new room. This can be done by multiplying 459 sq ft (the total sq ft of Holden’s bedroom and bathroom combined) by 2, which gives us 918 sq ft. Therefore, the new room will have 918 sq ft. The basketball field of Tom is 200 sq ft larger than Holden’s bathroom so it is not relevant to the calculation process of this problem. ✓</p> <p>I³C-Select Holden’s bedroom is 309 sq ft and his bathroom is 150 sq ft. The total sq ft of Holden’s bedroom and bathroom combined is 459 sq ft. We need to find a space that is twice as large as Holden’s bedroom and bathroom combined. Therefore, the new room will be $2 \times 459 \text{ sq ft} = \mathbf{918}$ sq ft. The condition The basketball field of Tom is 200 sq ft larger than Holden’s bathroom is not relevant to the calculation process of this problem, as it does not involve any calculation involving Holden’s bedroom and bathroom. ✓</p>

Table 13: Example outputs by Instruct-CoT, Instruct-CoT + I³C and I³C-Select. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color.

Problem	James buys 6 t-shirts for 50% off. They each cost \$20. Emma works at the mall with an hourly salary of \$10. How much did James pay?
Gold Answer	$(20/2) \times 6 = 60$
Verification Outputs	<p>■ First, James paid 50% off the normal price for the 6 t-shirts, which would normally cost \$20 each. This means he paid \$10 per t-shirt. So, James paid \$60 total for the 6 t-shirts. The condition James buys 6 t-shirts for 50% off is relevant to the calculation process, as it determined the amount that James paid for the 6 t-shirts. Emma’s hourly salary of \$10 is not relevant to the calculation process of this problem.</p> <p>■ James bought 6 t-shirts, so he paid for 6 times \$20. Then, he got 50% off, so 50% of \$20 is \$10, so he paid for 6 times \$10. The condition They each cost \$20 is relevant to the problem, because it tells us how much James paid for each t-shirt. So, the answer is James paid \$60.</p> <p>■ James bought 6 t-shirts at 50% off, which means the t-shirts were 50% cheaper than their original price. This means that each t-shirt cost \$20. Multiplying 6 t-shirts by \$20, James paid a total of \$120 for the 6 t-shirts. The condition that Emma works at the mall with an hourly salary of \$10 is not relevant to the calculation process of this problem.</p>
Reasoning Path	<p>Instruct-CoT</p> <p>Since James bought 6 t-shirts for 50% off, the price of each t-shirt is reduced to \$20. Therefore, James paid \$120 in total for the 6 t-shirts. Emma’s hourly salary of \$10 is irrelevant to the question. ✗</p> <hr/> <p>Instruct-CoT + I³C</p> <p>James bought 6 t-shirts that were 50% off, so the original price of each t-shirt was \$20. Therefore, James paid \$10 for each t-shirt, and he bought 6 t-shirts, so he paid \$60 in total. The condition that Emma works at the mall with an hourly salary of \$10 is not relevant to the calculation process of this problem. ✓</p> <hr/> <p>I³C-Select</p> <p>First, James paid 50% off the normal price for the 6 t-shirts, which would normally cost \$20 each. This means he paid \$10 per t-shirt. So, James paid \$60 total for the 6 t-shirts. The condition James buys 6 t-shirts for 50% off is relevant to the calculation process, as it determined the amount that James paid for the 6 t-shirts. Emma’s hourly salary of \$10 is not relevant to the calculation process of this problem. ✓</p>

Table 14: Example outputs by Manual-CoT, Manual-CoT + I³C and I³C-Select. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color.

Problem	James collects all the fruits from his 2 trees. Each tree has 20 plants. Each plant has 1 seed and he plants 60% of those. James' sister is 10 years old. How many trees did James plant?
Gold Answer	$2 \times 20 \times 60\% = 24$
Verification Outputs	<p>■ James collected all the fruits from his 2 trees. This means he did not plant any new trees. Each tree has 20 plants. This means James had 40 plants in total. Each plant has 1 seed. This means James had 40 seeds in total. He planted 60% of those seeds. This means he planted 24 seeds. Therefore, James planted 24 seeds, but he did not plant any new trees. The condition James collects all the fruits from his 2 trees is relevant to the calculation process because it indicates that James did not plant any new trees, only seeds.</p> <p>■ First, James collected all the fruits from his two trees. Each tree has 20 plants, so James collected 40 plants in total. Second, each plant has 1 seed, so James collected 40 seeds in total. Third, he planted 60% of those seeds, so James planted 24 seeds in total. Finally, we can conclude that James planted two trees, since each tree had 20 plants and he planted 24 seeds in total. So, to answer the question, James planted two trees. Yes, the condition Each tree has 20 plants is relevant to the calculation process of the problem How many trees did James plant?, since it helps us determine how many plants he collected in total and how many seeds he planted in total.</p> <p>■ James collects all the fruits from his 2 trees. Each tree has 20 plants. Each plant has 1 seed. This means there are 40 seeds in total. James then plants 60% of those, which means he plants 24 seeds. The condition James' sister is 10 years old is not relevant to the calculation process of this problem. Therefore, James planted 24 trees.</p>
Reasoning Path	<p>Manual-CoT</p> <p>James collected all the fruits from his 2 trees. Each tree has 20 plants. Each plant has 1 seed. He planted 60% of those. 60% is 0.6, so he planted $0.6 \times 20 = 12$ of those. James' sister is 10 years old. This has nothing to do with the question, so the answer is still 12. ✗</p> <p>Manual-CoT + I³C</p> <p>James collected all the fruits from his two trees, so he has 40 plants. Each plant has 1 seed, so he has 40 seeds. He plants 60% of those, so he plants 24 seeds. Therefore, James planted 24 trees. The condition James' sister is 10 years old is not relevant to the calculation process of this problem. ✓</p> <p>I³C-Select</p> <p>First, James collected all the fruits from his two trees, so he has 40 plants. Second, each plant has 1 seed, so he has 40 seeds. Third, he plants 60% of those, so he plants 24 seeds. Therefore, James planted 24 trees. The condition that James' sister is 10 years old is not relevant to the calculation process of this problem. The relevant information to answer this question is that James collected all the fruits from his two trees, each tree has 20 plants, and he plants 60% of those. To calculate how many trees James planted, we need to calculate the number of seeds he planted. So the answer would be: 24 trees. ✓</p>

Table 15: Example outputs by Auto-CoT, Auto-CoT + I³C and I³C-Select. Irrelevant conditions in the problem description is highlighted with red color, and the LLM-generated final answer is highlighted with yellow color.

Problem	Hamza has several empty buckets of different sizes, holding either 3, 5, or 6 liters. She fills the 5-liter bucket and pours as much as she can into the 3-liter bucket. Then, she pours the remainder into the 6-liter bucket. Jack drinks 7 liters of milk every day. How much more water, in liters, can Hamza put into the 6-liter bucket, without overflowing?
Gold Answer	$6 - (5 - 3) = 4$
Verification Outputs	<p>■ First, Hamza fills the 5-liter bucket and then pours as much as she can into the 3-liter bucket. Since the 3-liter bucket has a capacity of 3 liters, she can pour all 5 liters into it. Then, she pours the remainder (2 liters) into the 6-liter bucket. That means she has put a total of 7 liters into the 6-liter bucket. Now, Jack drinks 7 liters of milk every day. That means the 6-liter bucket is emptied every day. Therefore, Hamza can put up to 6 liters more water, in liters, into the 6-liter bucket, without overflowing. Yes, the condition Hamza has several empty buckets of different sizes, holding either 3, 5, or 6 liters. is relevant to the calculation process of the problem. This condition gives us the capacity of each bucket, which is necessary to calculate the amount of water Hamza can put into the 6-liter bucket without overflowing.</p> <p>■ First, Hamza filled the 5-liter bucket with water. She then poured as much of the water as she could into the 3-liter bucket. Since the 3-liter bucket can hold a maximum of 3 liters of water, Hamza would have been able to pour 2 liters of water from the 5-liter bucket into the 3-liter bucket. The remaining 3 liters of water from the 5-liter bucket would then be poured into the 6-liter bucket. This means that the 6-liter bucket currently holds 3 liters of water. Based on this information, we can calculate that Hamza can put an additional 3 liters of water into the 6-liter bucket, without overflowing. This is because the 6-liter bucket can hold a maximum of 6 liters of water, and it currently holds 3 liters of water. The condition Jack drinks 7 liters of milk every day is not relevant to the calculation process of this problem.</p>
Reasoning Path	<p>Auto-CoT</p> <p>The 5-liter bucket is filled. She pours as much as she can into the 3-liter bucket. That leaves 2 liters in the 5-liter bucket. She then pours the remainder into the 6-liter bucket. That leaves 2 liters in the 6-liter bucket. Jack drinks 7 liters of milk every day, so she has to leave 7 liters of water for him. Therefore, she can put 5 liters of water in the 6-liter bucket without overflowing. The answer is 5. ✗</p> <p>Auto-CoT + I³C</p> <p>Hamza first fills the 5-liter bucket. She then pours as much as she can into the 3-liter bucket, filling it to capacity. The remaining 2 liters of water from the 5-liter bucket are then poured into the 6-liter bucket, leaving 4 liters of space for more water (6 liters - 2 liters = 4 liters). Hamza can add 4 more liters of water into the 6-liter bucket without overflowing. The answer is 4. ✓</p> <p>I³C-Select</p> <p>Hamza has a 5-liter bucket. This is relevant to the calculation process as it tells us the size of the bucket Hamza has. She fills the 5-liter bucket and pours as much as she can into the 3-liter bucket. This is relevant to the calculation process as it tells us how much water Hamza is transferring from the 5-liter bucket to the 3-liter bucket. The 3-liter bucket can only hold 3 liters of water, so Hamza would be able to pour 3 liters of water from the 5-liter bucket into the 3-liter bucket. She then pours the remainder of the water (2 liters) into the 6-liter bucket. This is relevant to the calculation process as it tells us how much water Hamza is transferring from the 5-liter bucket to the 6-liter bucket. The 6-liter bucket currently has 2 liters of water in it, so it can hold 4 more liters of water without overflowing. So the answer would be 4 liters. ✓</p>