7-2023

Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models

Lei WANG
Singapore Management University, lei.wang.2019@phdcs.smu.edu.sg

Wanyu XU
Southwest Jiaotong University

Yihuai LAN

Zhiqiang HU
Singapore University of Technology and Design

Yunshi LAN
East China Normal University

See next page for additional authors

Citation

WANG, Lei; XU, Wanyu; LAN, Yihuai; HU, Zhiqiang; LAN, Yunshi; LEE, Roy Ka-Wei; and LIM, Ee-peng. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. (2023). *61st Annual Meeting of the Association for Computational Linguistics, ACL 2023: Toronto, July 9-14: Proceedings*. 2609-2634.

Available at: https://ink.library.smu.edu.sg/sis_research/8054

Plan-and-Solve Prompting: Improving Zero-Shot Chain-of-Thought Reasoning by Large Language Models

Lei Wang[1] **Wanyu Xu**[2] **Yihuai Lan** **Zhiqiang Hu**[3] **Yunshi Lan**[4]
Roy Ka-Wei Lee[3] **Ee-Peng Lim**[1]*
[1]Singapore Management University
[2]Southwest Jiaotong University
[3]Singapore University of Technology and Design
[4]East China Normal University

Abstract

Large language models (LLMs) have recently been shown to deliver impressive performance in various NLP tasks. To tackle multi-step reasoning tasks, few-shot chain-of-thought (CoT) prompting includes a few manually crafted step-by-step reasoning demonstrations which enable LLMs to explicitly generate reasoning steps and improve their reasoning task accuracy. To eliminate the manual effort, Zero-shot-CoT concatenates the target problem statement with "*Let's think step by step*" as an input prompt to LLMs. Despite the success of Zero-shot-CoT, it still suffers from three pitfalls: calculation errors, missing-step errors, and semantic misunderstanding errors. To address the missing-step errors, we propose Plan-and-Solve (PS) Prompting. It consists of two components: first, devising a plan to divide the entire task into smaller subtasks, and then carrying out the subtasks according to the plan. To address the calculation errors and improve the quality of generated reasoning steps, we extend PS prompting with more detailed instructions and derive PS+ prompting. We evaluate our proposed prompting strategy on ten datasets across three reasoning problems. The experimental results over GPT-3 show that our proposed zero-shot prompting consistently outperforms Zero-shot-CoT across all datasets by a large margin, is comparable to or exceeds Zero-shot-Program-of-Thought Prompting, and has comparable performance with 8-shot CoT prompting on the math reasoning problem. The code can be found at https://github.com/AGI-Edgerunners/Plan-and-Solve-Prompting.

1 Introduction

Large language models (LLMs) (Brown et al., 2020; Thoppilan et al., 2022; Chowdhery et al., 2022) have recently proven highly effective in various NLP tasks. Unlike the previous pre-trained language models (PTMs) (Devlin et al., 2019; Liu et al., 2019), these LLMs are typically provided as a service, with no access to model parameters due to commercial considerations and potential risks of misuse (Sun et al., 2022). Thus, it is challenging to fine-tune LLMs for downstream tasks (He et al., 2021; Houlsby et al., 2019; Devlin et al., 2019). Instead, we leverage LLMs to solve complex reasoning problems by eliciting their strong reasoning abilities over their embedded knowledge using instructions (or trigger sentences). So far, LLMs have shown impressive abilities to solve new reasoning problems by simply conditioning them on a few illustrative examples (i.e., few-shot learning) or a prompt to solve new problems without illustrative examples (i.e., zero-shot learning).

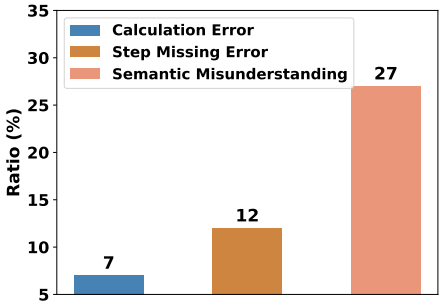

Figure 1: Error analysis of 46 GSM8K problems with incorrect answers returned by Zero-shot-CoT using GPT-3 LLM. Following Wei et al. (2022b) and Wang et al. (2022a), we assign "Calculation Error" (7%), "Step Missing Error" (12%), or "Semantic misunderstanding Error" (27%) to each incorrect answer.

To tackle multi-step complex reasoning tasks using LLMs, Wei et al. (2022b) proposes few-shot chain-of-thought (CoT) prompting, which enables LLMs to explicitly generate the intermediate reasoning steps before predicting the final answer with a few manual step-by-step reasoning demonstration examples. In (Kojima et al., 2022), Zero-shot CoT eliminates the need for manually crafted examples in prompts by appending "*Let's think step by step*" to the target problem fed to LLMs such

*Corresponding author.

Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics
Volume 1: Long Papers, pages 2609–2634

as GPT-3. This simple prompting strategy surprisingly enables LLMs to yield performance similar to few-shot CoT prompting.

Despite the remarkable success of Zero-shot-CoT in solving multi-step reasoning tasks, its results on a sample of 100 arithmetic test examples still point to three pitfalls (as shown in Figure 1): (i) Calculation errors (in 7% of test examples): These are errors in the calculation leading to wrong answers; (ii) Missing Step errors (in 12% of test examples): These occur when some intermediate reasoning step(s) is missed-out especially when there are many steps involved; (iii) Semantic misunderstanding (in 27% of test examples): There are other errors in semantic understanding of the problem and coherence of reasoning steps likely to be caused by the insufficient capability of LLMs.

To address the issue of Zero-shot-CoT caused by missing reasoning steps, we propose Plan-and-Solve (PS) Prompting. It consists of two components: first, devising a plan to divide the entire task into smaller subtasks, and then carrying out the subtasks according to the plan. In our experiments, we simply replace "*Let's think step by step*" of Zero-shot-CoT with "*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*" (see Figure 2 (b)).

To address the calculation errors of Zero-shot-CoT and improve the quality of generated reasoning steps, we add more detailed instructions to PS prompting. Specifically, we extend it with "*extract relevant variables and their corresponding numerals*" and "*calculate intermediate results (pay attention to calculation and commonsense)*" instructions. This prompting variant is called the PS+ prompting strategy (see Figure 3 (b)). Despite its simplicity, PS+ strategy greatly improves the quality of the generated reasoning process. Moreover, this prompting strategy can be easily customized to solve a variety of problems other than math reasoning, such as commonsense and symbolic reasoning problems.

We evaluate our proposed prompting on six math reasoning datasets, including AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021), MultiArith, AddSub, SingleEq, and SVAMP (Patel et al., 2021), two commonsense reasoning datasets (CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021)), and two symbolic reasoning datasets (Last Letter and Coin Flip (Wei et al., 2022b)). The results of our experiments with GPT-3 show that our proposed Zero-shot-PS+ prompting consistently outperforms Zero-shot-CoT across all reasoning problems and datasets by a large margin, and is comparable to or exceeds Zero-shot-Program-of-Thought (PoT) Prompting (Chen et al., 2022)). Furthermore, although PS+ prompting does not require manual demonstration examples, it has a performance similar to an 8-shot CoT prompting in arithmetic reasoning.

Overall, our results suggest that (a) Zero-shot PS prompting is capable of generating a higher-quality reasoning process than Zero-shot-CoT prompting, as the PS prompts provide more detailed instructions guiding the LLMs to perform correct reasoning tasks; (b) Zero-shot PS+ prompting outperforms Few-shot manual-CoT prompting on some datasets, indicating that in some instances it has the potential to outperform manual Few-shot CoT prompting, which hopefully will spark further development of new CoT prompting approaches to elicit reasoning in LLMs.

2 Plan-and-Solve Prompting

Overview. We introduce PS prompting, a new zero-shot CoT prompting method, which enables LLMs to explicitly devise a plan for solving a given problem and generate the intermediate reasoning process before predicting the final answer for the input problem. As opposed to prior few-shot CoT approaches where step-by-step few-shot demonstration examples are included in the prompt, the zero-shot PS prompting method does not require demonstration examples, and its prompt covers the problem itself and a simple trigger sentence. Similar to Zero-shot-CoT, Zero-shot PS prompting consists of two steps. In step 1, the prompt first makes an inference using the proposed prompting template to generate the reasoning process and the answer to a problem. In step 2, it extracts the answer for evaluation by using the answer extraction prompting, such as "Therefore, the answer (arabic numerals) is".

2.1 Step 1: Prompting for Reasoning Generation

To solve the input problem while avoiding errors resulting from incorrect calculation and missing reasoning steps, this step aims to construct templates to meet the following two criteria:

- The templates should elicit LLMs to deter-

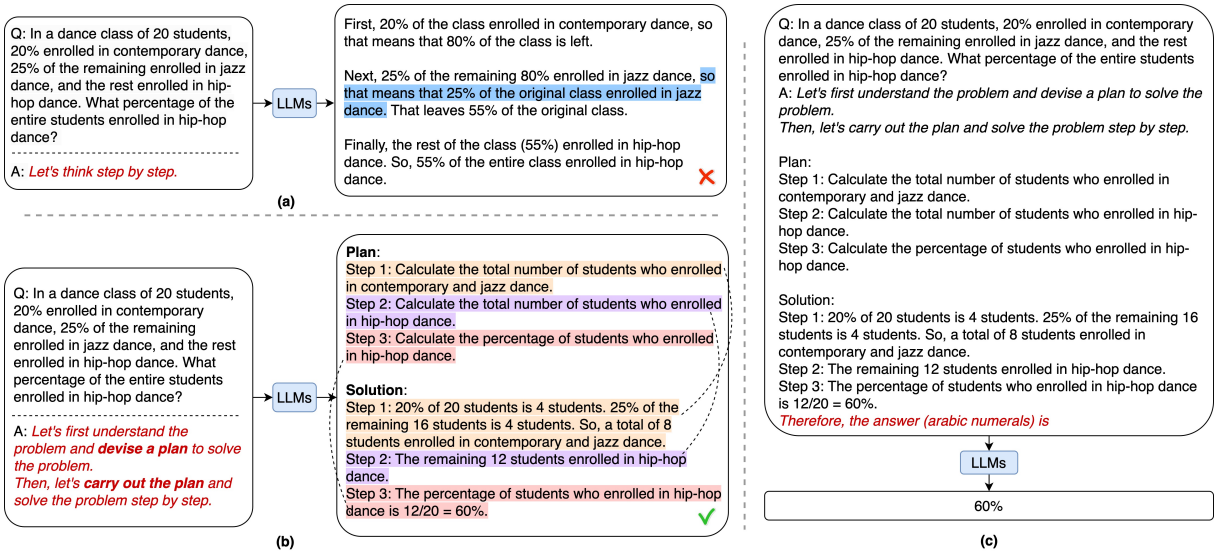

Figure 2: Example inputs and outputs of GPT-3 with (a) Zero-shot-CoT prompting, (b) Plan-and-Solve (PS) prompting, and (c) answer extraction prompting. While Zero-shot-CoT encourages LLMs to generate multi-step reasoning with "*Let's think step by step*", it may still generate wrong reasoning steps when the problem is complex. Unlike Zero-shot-CoT, PS prompting first asks LLMs to devise a plan to solve the problem by generating a step-by-step plan and carrying out the plan to find the answer.

mine subtasks and accomplish the subtasks.

- The templates should guide LLMs to pay more attention to calculations and intermediate results and to ensure that they are correctly performed as much as possible.

To meet the first criterion, we follow Zero-shot-CoT and first convert the input data example into a prompt with a simple template "Q: `[X]`. A: `[T]`". Specifically, the input slot `[X]` contains the input problem statement and a hand-crafted instruction is specified in the input slot `[T]` to trigger LLMs to generate a reasoning process that includes a plan and steps to complete the plan. In Zero-shot-CoT, the instruction in the input slot `[T]` includes the trigger instruction '*Let's think step by step*". Our Zero-shot PS prompting method instead includes the instructions "*devise a plan*" and "*carry out the plan*" as shown in Figure 2(b). Thus, the prompt would be "Q: `[X]`. A: *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*."

We then pass the above prompt to the LLM which subsequently outputs a reasoning process. In accordance with Zero-shot-CoT, our method uses the greedy decoding strategy (1 output chain) for generating output by default.

To meet the second criterion, we extend the plan-based trigger sentence with more detailed instructions. Specifically, "*pay attention to calculation*" is added to the trigger sentence to request the LLMs to perform calculations as accurately as possible. To reduce errors resulting from missing necessary reasoning steps, we include "*extract relevant variables and their corresponding numerals*" to explicitly instruct the LLMs not to ignore relevant information in the input problem statement. We hypothesize that if the LLMs leave out the relevant and important variables, it is more likely to miss out relevant reasoning steps. Correlation analysis of generated content of variable and the missing reasoning step errors, shown in Figure 5, empirically supports this hypothesis (correlation value is less than 0). Additionally, we add "*calculate intermediate results*" to the prompt to enhance LLM's ability to generate relevant and important reasoning steps. The specific example is illustrated in Figure 3(b). At the end of Step 1, LLM generates the reasoning text which includes the answer. For example, the generated reasoning text in Figure 3(b) includes "*Combined weight of Grace and Alex = 125 + 498 = 623 pounds*". The strategy of adding specific descriptions to the trigger sentence represents a new way to improve zero-shot performance on complex reasoning.

2.2 Step 2: Prompting for Answer Extraction

Similar to Zero-shot-CoT, we devise another prompt in Step 2 to get the LLM to extract the final numerical answer from the reasoning text gener-

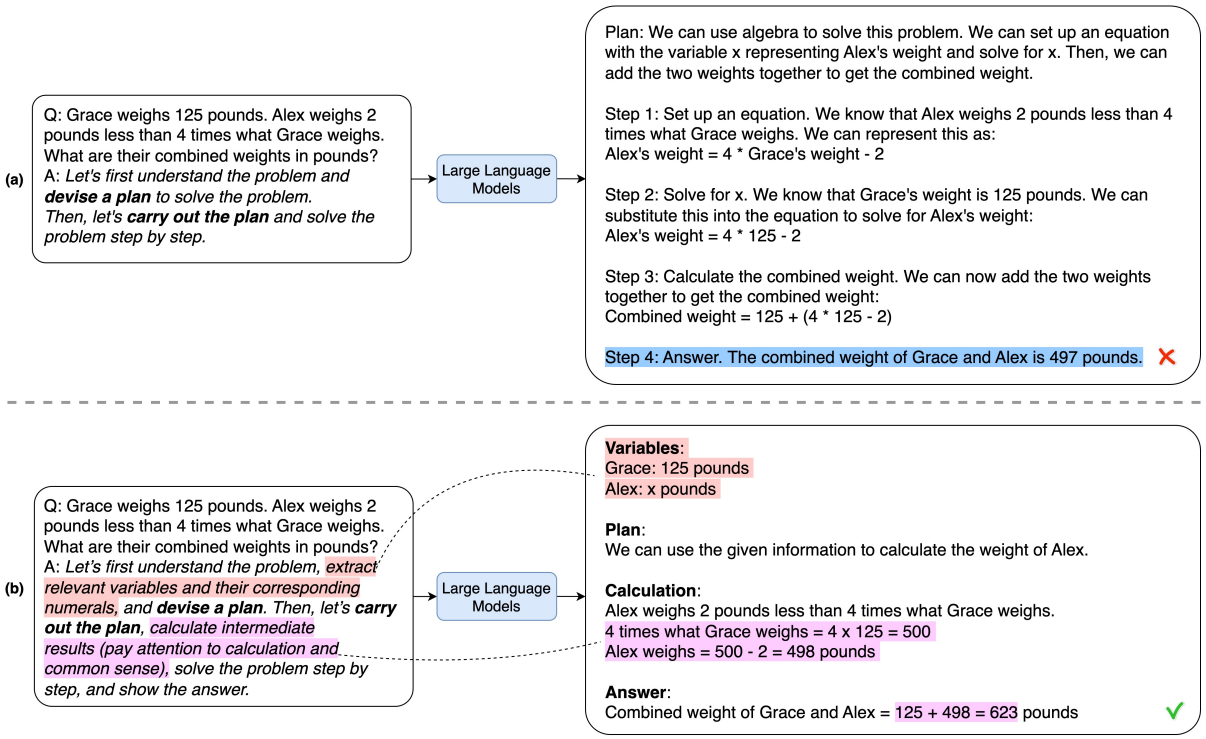

Figure 3: Example inputs and outputs of GPT-3 with (a) Plan-and-Solve (PS) Prompting and (b) Plan-and-Solve prompting with more detailed instructions (PS+ prompting). PS+ prompting greatly improves the quality of the generated reasoning process.

ated in Step 1. This prompt includes the answer extraction instruction appended to the first prompt followed by the LLM generated reasoning text. This way, LLM is expected to return the final answer in the desired form.

Based on the example in Figure 3(b), the prompt used in Step 2 will include "*Q: Grace weighs 125 pounds ⋯ Variables: Grace: 125 pounds ⋯ Answer: Combined weight of Grace and Alex = 125 + 498 = 623 pounds. Therefore, the answer (arabic numerals) is*". For this example, the final answer returned by LLM is "*623*".

3 Experimental Setup

3.1 Benchmarks

The proposed method is evaluated on the ten benchmark datasets from three categories of reasoning problems: **Arithmetic Reasoning:** (1) the GSM8K (Cobbe et al., 2021) dataset of high quality linguistically diverse grade school math word problems created by human problem writers, (2) the SVAMP (Patel et al., 2021) benchmark of one-unknown arithmetic word problems for up-to-4 grade level students by making simple changes to a set of problems from another existing dataset, (3) the MultiArith (Roy and Roth, 2016) dataset of math word problems requiring multiple reasoning steps and operations, (4) the AddSub (Hosseini et al., 2014) dataset of addition and subtraction arithmetic word problems, (5) the AQUA (Ling et al., 2017) dataset of algebraic word problems with natural language rationales, and (6) the SingleEq (Koncel-Kedziorski et al., 2015) dataset of single-equation grade-school algebra word problems with multiple math operations over non-negative rational numbers and one variable; **Commonsense Reasoning**: (7) the CSQA (Talmor et al., 2019) benchmark dataset of multiple-choice questions that require different types of commonsense knowledge to obtain the correct answers; and (8) the StrategyQA (Geva et al., 2021) benchmark dataset with questions requiring multi-step reasoning but the reasoning steps are not given. Hence, they are to be inferred; **Symbolic Reasoning**: (9) the Last Letter Concatenation (Wei et al., 2022b) dataset of questions requiring the last letters of words in a name to be concatenated (e.g., "*James Brown*" → "*sn*"), and (10) the Coin Flip (Wei et al., 2022b) dataset of questions on whether a coin is still heads up after it is flipped or not flipped based on steps given in the questions. Table 1 shows dataset statistics.

Table 1: Details of datasets being evaluated. Math: arithmetic reasoning. CS: commonsense reasoning. Sym.: symbolic reasoning.

Dataset	Domain	# Samples	Ave. words	Answer
MultiArith	Math	600	31.8	Number
AddSub	Math	395	31.5	Number
GSM8K	Math	1319	46.9	Number
AQUA	Math	254	51.9	Option
SingleEq	Math	508	27.4	Number
SVAMP	Math	1000	31.8	Number
CSQA	CS	1221	27.8	Option
StrategyQA	CS	2290	9.6	Yes / No
Last Letters	Sym.	500	15.0	String
Coin Flip	Sym.	500	37.0	Yes / No

3.2 Zero-shot and Few-shot Baselines

We compare our proposed zero-shot PS and PS+ prompting methods with three types of prompting baselines: (1) **Zero-shot baselines.** We include zero-shot-CoT (Kojima et al., 2022) and zero-shot-PoT (Chen et al., 2022). The former appends "Let's think step by step" to the prompt without any demonstration examples. The latter uses LLM (mainly OpenAI Codex[1]) to generate a Python program and then derive an answer by executing the generated program on a Python interpreter; (2) **Few-shot with manual demonstrations.** Manual-CoT (Wei et al., 2022b) creates eight hand-crafted examples as demonstrations. (3) **Few-shot with automatic demonstrations.** Auto-CoT (Zhang et al., 2022) automatically selected examples by clustering with diversity and generates reasoning chains using zero-shot-CoT to construct demonstrations.

3.3 Implementations

Following Auto-CoT (Zhang et al., 2022), we use the public GPT-3 (Brown et al., 2020) (175B) as the backbone language model, which is one of the most widely-used LLMs with public APIs[2]. Since `text-davinci-003` is an upgraded version of `text-davinci-002`, which can produce higher-quality writing, accommodate more complex instructions, and perform better at longer-form content generation, We report the results using `text-davinci-003` engine for GPT-3 in the main paper. We set the temperature to 0 (argmax sampling) throughout our experiments for the greedy decoding strategy. We also include two few-shot baselines, Manual-CoT and Auto-CoT, we use 8 demonstration examples for MultiArith, GSM8K, AddSub, SingleEq, and SVAMP, 4 examples for AQuA and Last Letters, 7 examples for CSQA, and 6 examples for StrategyQA as suggested in the original papers, Wei et al. (2022b) and Zhang et al. (2022). Evaluation metrics wise, we follow Manual-CoT (Wei et al., 2022b) and report the accuracy of all methods across datasets.

[1]https://openai.com/blog/openai-codex/

[2]https://beta.openai.com/docs/models/gpt-3

4 Experimental Results

4.1 Main Results

Arithmetic Reasoning. Table 2 reports the accuracy comparison of our method and existing zero-shot and few-shot methods on the arithmetic reasoning datasets. In the zero-shot setting, our PS+ prompting (i.e., PS prompting with more detailed instructions) consistently outperforms Zero-shot-CoT across all arithmetic reasoning datasets by a large margin. Specifically, PS+ prompting improves the accuracy over Zero-shot CoT by at least 5% for all datasets except GSM8K which sees a 2.9% improvement. The exception could be due to GSM8K being a more challenging dataset from the linguistics complexity aspect. PS prompting also outperforms Zero-shot-CoT across all datasets, and enjoys 2.5% higher average accuracy than that of Zero-shot CoT.

Compared with another competitive Zero-shot baseline, PoT, the performance of PS(+) and PS promptings are still impressive. PS+ prompting outperforms PoT on five out of six arithmetic datasets. PS prompting also outperforms PoT on three arithmetic datasets. The results suggest that adding more detailed instructions to the prompt can effectively elicit higher-quality reasoning steps from LLMs.

Compared with the few-shot methods, Manual CoT and Auto-CoT, PS+ prompting yields an average accuracy (76.7%) slightly lower than Manual-CoT (77.6%) but higher than Auto-CoT (75.9%). While this is an unfair comparison, this result indicates that zero-shot prompting can outperform few-shot CoT prompting, which hopefully will spark further development of new ways with a less manual effort to effectively elicit reasoning in LLMs.

Commmonsense Reasoning. Table 3 shows the results on commonsense reasoning datasets: CommonsenseQA and StrategyQA. We only include our better zero-shot PS+ prompting strategy in this comparison. Zero-shot PoT is excluded as it does not work on this problem. While PS+ prompting underperforms Few-Shot-CoT(Manual) on this

Table 2: Accuracy comparison on six math reasoning datasets. The best and second best results are boldfaced and underlined respectively.

Setting	Method (text-davinci-003)	MultiArith	GSM8K	AddSub	AQuA	SingleEq	SVAMP	Average
Zero-Shot	CoT	83.8	56.4	85.3	38.9	88.1	69.9	70.4
	PoT	**92.2**	57.0	85.1	<u>43.9</u>	<u>91.7</u>	70.8	<u>73.5</u>
	PS (ours)	87.2	<u>58.2</u>	<u>88.1</u>	42.5	89.2	<u>72.0</u>	72.9
	PS+ (ours)	<u>91.8</u>	**59.3**	**92.2**	**46.0**	**94.7**	**75.7**	**76.7**
Few-Shot	Manual-CoT	93.6	58.4	91.6	48.4	93.5	80.3	77.6
	Auto-CoT	95.5	57.1	90.8	41.7	92.1	78.1	75.9

Table 3: Accuracy on commonsense reasoning datasets.

Method	CSQA	StrategyQA
Few-Shot-CoT (Manual)	78.3	71.2
Zero-shot-CoT	65.2	63.8
Zero-shot-PS+ (ours)	**71.9**	**65.4**

Table 4: Accuracy on symbolic reasoning datasets.

Method	Last Letter	Coin Flip
Few-Shot-CoT (Manual)	70.6	100.0
Zero-shot-CoT	64.8	96.8
Zero-shot-PS+ (ours)	**75.2**	**99.6**

problem, it consistently outperforms Zero-shot-CoT on CommonsenseQA (71.9% vs. 65.2%) and StrategyQA (65.4% vs. 63.8%) datasets.

Symbolic Reasoning. Table 4 shows the accuracy of PS+ prompting against Zero-shot-CoT and Few-shot-CoT on symbolic reasoning datasets: Last Letters and Coin Flip. Zero-shot PoT is again excluded as it is not designed for the problem. On Last Letters, our Zero-shot PS+ prompting (75.2%) outperforms Manual-CoT (70.6%) and Zero-shot-CoT (65.2%). On Coin Flip, Zero-shot PS+ prompting (99.6%) is slightly worse than Manual-CoT (100.0%) but outperforms Zero-shot-CoT by a good margin (96.8%). More examples from the experiment results can be found in Appendix A.2.

4.2 Analysis

Results of Prompting with Self-Consistency. Self-consistency (Wang et al., 2022b) (SC) is proposed to reduce randomness in LLM's output by generating N reasoning results and determining the final answer by majority voting. With SC, the methods' results are usually expected to be consistent and better. Hence, we evaluate Zero-shot PS+ prompting with SC on GSM8K and SVAMP datasets. We set the temperature to 0.7 and N to 10 for experiments with SC. Figure 4 shows that PS+ prompting with SC (73.7% and 84.4%) substantially outperforms that without SC (58.7% and

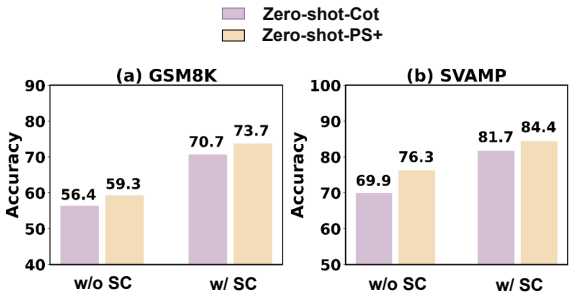

Figure 4: Results of methods with and without self-consistency (SC) on GSM8K and SVAMP.

75.7%) on GSM8K and SVAMP, respectively. The former also consistently outperforms Zero-shot-CoT with SC (70.7% and 81.7%) on GSM8K and SVAMP, respectively, although Zero-shot CoT also enjoys improvement with the self consistency approach.

Effect of Prompts. Table 5 demonstrates a comparison of the performance of 6 different input prompts. Prompts 1 and 2 are used in Zero-shot CoT and Zero-shot PoT respectively. The rest are variations of prompts used in Step 1 of the Zero-shot PS+ prompting strategies with greedy decoding. We observe that Prompt 3 with variables and numeral extraction performs worse than Prompt 1 of Zero-shot-CoT. The reason is that Prompt 3 doesn't include instructions for devising and completing a plan. However, the other prompts of Zero-shot-PS+ perform well as we add more instructions about intermediate results calculation, plan design, and implementation. The above results conclude that LLMs are capable of generating high-quality reasoning text when the prompts include more detailed instructions to guide the LLMs. More prompts for different reasoning problems can be found in Appendix A.1.

Error Analysis. To qualitatively evaluate the impact of the Zero-shot-PS+ prompting on calculation errors and reasoning steps missing errors, we examine the distribution of errors on the GSM8K dataset. We first randomly sample 100 problems

Table 5: Performance comparison of trigger sentences measured on GSM8K and SVAMP datasets with `text-davinci-003` except for No. 2 (`code-davinci-002`). (*1) means the trigger sentence used in Zero-shot-CoT (Kojima et al., 2022). (*2) means the trigger sentence used in Zero-shot-PoT (Chen et al., 2022).

No.	Trigger Sentence		GSM8K	SVAMP
1	Let's think step by step.	(*1)	56.4	69.9
2	import math import numpy as np # Question: example['question'] # Answer this question by implementing a solver() function. def solver(): # Let's write a Python program step by step, and then return the answer # Firstly, we need define the following variable:	(*2)	57.0	70.8
3	Extract variables and assign their corresponding numerals to these variables first and then solve the problem step by step.		50.5	69.5
4	Firstly, extract variables and their corresponding numerals. Then, calculate intermediate variables. Finally, solve the problem step by step.		54.8	70.8
5	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.		58.2	72.0
6	Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.		**59.3**	**75.7**

Table 6: Distribution of error types of 100 examples from GSM8K where Zero-shot-CoT, zero-shot PS (Zero-shot-PS) prompting, and zero-shot PS+ prompting get incorrect final answers.

Method	**Calculation**	**Missing**	**Semantic**
Zero-shot-CoT	7%	12%	27%
Zero-shot-PS	7%	10%	26%
Zero-shot-PS+	5%	7%	27%

from GSM8K, generate the reasoning text, and extract answers using Zero-Shot-CoT, Zero-shot-PS, and Zero-shot-PS+ prompting strategies. Zero-Shot-CoT generated incorrect final answers for 46 of the problems, 43 for Zero-shot-PS, and 39 for Zero-shot-PS+. Subsequently, we analyze and determine the error types of all these problems as shown in Table 6.

The analysis results show that PS+ prompting achieves the least calculation (5%) and missing-step (7%) errors, and semantic understanding errors comparable to Zero-shot-CoT. Zero-shot-PS has slightly more errors but is still better than Zero-shot-CoT. Their plan-and-solve prompts thus effectively guide the LLMs to generate clear and complete reasoning steps. Moreover, the additional detailed instructions in PS+ prompting (i.e., "*extract relevant variables and their corresponding numerals*" and "*calculate intermediate variables*") enable the LLMs to generate high-quality reasoning steps leading to fewer calculation errors.

Correlation Analysis of Generated Reasoning and Error Types. To obtain deeper insight into the impact of PS+ prompting on error types, we examine the correlation between the sub-parts of the generated reasoning and error types. Specifically, we analyze the existence of variable definition, reasoning plan, and solution in the generated reasoning text and correlate them with the three error types. The set of problems used for this analysis study is the same as that used in the earlier error type analysis. Figure 5 shows the correlation matrix among the existence of variable definitions, plans, solutions and three different types of errors. It is observed that both variable definition and plan existences have a negative correlation with calculation errors and missing-reasoning-step errors. The Zero-shot-PS+ prompt can further improve the performance of LLMs on mathematical reasoning problems by reducing calculation errors and missing-reasoning-step errors.

Exploring the Presence of Plans in PS Predictions. To ascertain the presence of a plan in each prediction made by PS, we conducted a random sampling of 100 data examples and examined their corresponding predictions. Our analysis reveals that 90 of the 100 predictions indeed incorporated a plan. This observation indicates the emergence

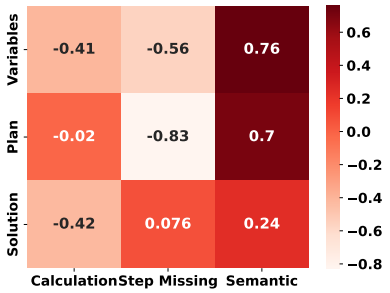

Figure 5: Correlation analysis of generated reasoning and error types of randomly sampled 100 data examples from GSM8K for Zero-shot-PS+.

of strong planning abilities in recent LLMs such as GPT-3.5 and GPT-4.

5 Related Work

5.1 Reasoning in NLP

It is well known that complex reasoning problems are challenging for NLP models, and such problems include mathematical reasoning (Cobbe et al., 2021; Patel et al., 2021; Ling et al., 2017; Koncel-Kedziorski et al., 2016) (requiring the ability to understand mathematical concepts, calculation, and multi-step reasoning), commonsense reasoning (Talmor et al., 2019; Geva et al., 2021) (requiring the ability to make judgments based on commonsense knowledge), and logical reasoning (Wei et al., 2022b) (requiring the ability to manipulate symbols by applying formal logical rules). Before the advent of Large Language models (LLMs), Talmor et al. (2019) trained the NLP model using explanations generated by the fine-tuned GPT model and found that the trained model yields better performance on commonsense QA problems. Hendrycks et al. (2021) attempted to fine-tune pretrained language models with labeled rationale, but found out that these fine-tuned models could not easily generate high-quality reasoning steps. Recent work by Wei et al. (2022a) showed that LLMs demonstrates strong reasoning ability when scaled up to tens of billions of parameters, such as GPT-3 (Brown et al., 2020) and PaLM (Chowdhery et al., 2022). These LLMs with a few demonstration exemplars can yield impressive performance across different NLP tasks. However, these models still perform poorly in problems that require multi-step reasoning. This may be due to the fact that the few exemplars provided are insufficient to unlock the LLMs' capabilities.

5.2 Prompting Methods

To exploit the reasoning ability in LLMs, Wei et al. (2022b) propose Chain-of-Thought prompting, appending multiple reasoning steps before the answer to the input question. With this simple few-shot prompting strategy, LLMs are able to perform much better in complex reasoning problems. Subsequently, many works (Wang et al., 2022a; Suzgun et al., 2022; Shaikh et al., 2022; Saparov and He, 2022) propose to further improve CoT prompting in different aspects, including prompt format (Chen et al., 2022), prompt selection (Lu et al., 2022), prompt ensemble (Wang et al., 2022b; Li et al., 2022; Weng et al., 2022; Fu et al., 2022), problem decomposition (Zhou et al., 2022; Khot et al., 2022; Dua et al., 2022; Press et al., 2022), and planning (Yao et al., 2022; Huang et al., 2022; Wang et al., 2023; Liu et al., 2023; Sun et al., 2023; Yao et al., 2023). Chen et al. (2022) introduced PoT prompting to use LLMs with code pre-training to write a program as a rationale for disentangling computation from reasoning. To do away with manual effort, Kojima et al. (2022) proposed Zero-shot-CoT to elicit reasoning step generation without exemplars. To leverage the benefit of demonstration examples and minimize manual effort, Zhang et al. (2022) designed Auto-CoT. It first automatically obtains k examples by clustering the given dataset. It then follows Zero-shot-CoT to generate rationales for the selected examples. Finally, demonstration examples are constructed by adding the generated rationales to selected examples as CoT prompts. Our work is different from the above works by focusing on eliciting multi-step reasoning by LLMs in a zero-shot approach. We ask LLMs to write a plan to decompose a complex reasoning task into multiple reasoning steps. Furthermore, we introduce detailed instructions to the prompt to avoid obvious errors in the reasoning steps. We refer readers to the survey (Huang and Chang, 2022) for more related works.

6 Conclusion

In this paper, we find that Zero-shot-CoT still suffers from three pitfalls: calculation errors, missing-reasoning-step errors, and semantic understanding errors. To address these issues, we introduce plan-and-solve prompting strategies (PS and PS+ prompting). They are new zero-shot prompting methods that guide LLMs to devise a plan that divides the entire task into smaller subtasks and then

carries out the subtasks according to the plan. Evaluation on ten datasets across three types of reasoning problems shows PS+ prompting outperforms the previous zero-shot baselines and performs on par with few-shot CoT prompting on multiple arithmetic reasoning datasets. Overall, our results suggest that (a) Zero-shot PS+ prompting can generate a high-quality reasoning process than Zero-shot-CoT prompting since the PS prompts can provide more detailed instructions guiding the LLMs to perform correct reasoning; (b) Zero-shot PS+ prompting has the potential to outperform manual Few-shot CoT prompting, which hopefully will spark further development of new CoT prompting approaches to elicit reasoning in LLMs. Moreover, PS(+) prompting is a general idea that can be used for non-reasoning tasks, and refining the plan is also an interesting idea. We leave them for future work.

7 Limitations

There are two limitations to this work. First, it takes effort to design the prompt to guide the LLMs to generate correct reasoning steps. The GPT-3 models are sensitive to the expressions in prompts. Thus we need to carefully design the prompts. Second, the proposed plan-and-solve prompting can help address the calculation errors and missing-reasoning-step errors, but the semantic misunderstanding errors still remain. We will explore how to address semantic misunderstanding errors by prompting instead of upgrading LLMs in the future.

8 Ethics

We experiment on six math reasoning datasets, including AQuA (Ling et al., 2017), GSM8K (Cobbe et al., 2021), MultiArith, AddSub, SingleEq, and SVAMP (Patel et al., 2021), two commonsense reasoning tasks (CommonsenseQA (Talmor et al., 2019) and StrategyQA (Geva et al., 2021)), and two symbolic tasks (Last Letter and Coin Flip (Wei et al., 2022b)), where GSM8K and SVAMP use the MIT License code, AQUA and StrategyQA use the Apache-2.0 code, the remaining datasets are unspecified.

The proposed prompts do not collect and use personal information about other individuals. The prompts we used are listed in Appendix. The prompts in this work do not contain any words that discriminate against any individual or group. In this work, prompts would not negatively impact other people's safety.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W Cohen. 2022. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. *arXiv preprint arXiv:2211.12588*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. PaLM: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*, pages 4171–4186.

Dheeru Dua, Shivanshu Gupta, Sameer Singh, and Matt Gardner. 2022. Successive prompting for decomposing complex questions. *arXiv preprint arXiv:2212.04092*.

Yao Fu, Hao Peng, Ashish Sabharwal, Peter Clark, and Tushar Khot. 2022. Complexity-based prompting for multi-step reasoning. *arXiv preprint arXiv:2210.00720*.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *TACL*, 9:346–361.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. *arXiv preprint arXiv:2110.04366*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*.

Mohammad Javad Hosseini, Hannaneh Hajishirzi, Oren Etzioni, and Nate Kushman. 2014. Learning to solve arithmetic word problems with verb categorization. In *EMNLP*, pages 523–533.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

Jie Huang and Kevin Chen-Chuan Chang. 2022. Towards reasoning in large language models: A survey. *arXiv preprint arXiv:2212.10403*.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *arXiv preprint arXiv:2205.11916*.

Rik Koncel-Kedziorski, Hannaneh Hajishirzi, Ashish Sabharwal, Oren Etzioni, and Siena Dumas Ang. 2015. Parsing algebraic word problems into equations. *Transactions of the Association for Computational Linguistics*, 3:585–597.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of NAACL*, pages 1152–1157.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2022. On the advance of making language models better reasoners. *arXiv preprint arXiv:2206.02336*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 158–167.

Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2022. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. *arXiv preprint arXiv:2209.14610*.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of NAACL*, pages 2080–2094.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Subhro Roy and Dan Roth. 2016. Solving general arithmetic word problems. *arXiv preprint arXiv:1608.01413*.

Abulhair Saparov and He He. 2022. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*.

Omar Shaikh, Hongxin Zhang, William Held, Michael Bernstein, and Diyi Yang. 2022. On second thought, let's not think step by step! bias and toxicity in zero-shot reasoning. *arXiv preprint arXiv:2212.08061*.

Simeng Sun, Yang Liu, Shuohang Wang, Chenguang Zhu, and Mohit Iyyer. 2023. Pearl: Prompting large language models to plan and execute actions over long documents.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. 2022. Black-box tuning for language-model-as-a-service. *arXiv preprint arXiv:2201.03514*.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of NAACL-HLT*, pages 4149–4158.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. 2022. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*.

Boshi Wang, Sewon Min, Xiang Deng, Jiaming Shen, You Wu, Luke Zettlemoyer, and Huan Sun. 2022a. Towards understanding chain-of-thought prompting: An empirical study of what matters. *arXiv preprint arXiv:2212.10001*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, and Denny Zhou. 2022b. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.

Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. 2023. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *arXiv preprint arXiv:2302.01560*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS 2022)*.

Yixuan Weng, Minjun Zhu, Shizhu He, Kang Liu, and Jun Zhao. 2022. Large language models are reasoners with self-verification. *arXiv preprint arXiv:2212.09561*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alex Smola. 2022. Automatic chain of thought prompting in large language models. *arXiv preprint arXiv:2210.03493*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Chi. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625*.

A Appendix

This section includes two parts: (1) Results of all prompts we have tried; (2) Example texts generated by Zero-shot-PS+. Unless otherwise mentioned, we use GPT3 (text-davinci-003) model.

A.1 Results of All Trigger Sentences

Tables 7 to 16 list the results of all prompts we have tried for each dataset.

A.2 Example Outputs by Zero-shot-PS+

Tables 17 to 25 list example outputs generated by Zero-shot-PS+ for each dataset.

Table 7: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on AQuA.

No.	Trigger Setence	Accuracy
1	Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan to solve the problem step by step.	42.5
2	Let's first understand the problem, extract all relevant variables and their corresponding numerals carefully, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and common sense), solve the problem step by step carefully, and show the answer.	42.9
3	Let's first understand the problem, extract relevant correct variables and their correct corresponding numerals, and devise complete plans. Then, let's carry out the plan, calculate intermediate variables including extracted variables (pay attention to correct numerical calculation and common sense), solve the problem by single equations, and show the answer.	43.7
4	Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.	46.0

Table 8: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on GSM8K.

No.	Trigger Setence	Accuracy
1	Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan to solve the problem step by step.	58.2
2	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.	58.7
3	Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.	59.3

Table 9: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on MultiArith.

No.	Trigger Setence	Accuracy
1	Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan to solve the problem step by step.	87.2
2	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.	88.3
3	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to the correctness of the calculation and common sense), solve the problem step by step, and show the answer.	90.5
4	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.	91.8

Table 10: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on SVAMP.

No.	Trigger Setence	Accuracy
1	Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan to solve the problem step by step.	72.0
2	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.	75.4
3	Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.	75.7

Table 11: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on AddSub.

No.	Trigger Setence	Accuracy
1	Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan to solve the problem step by step.	87.3
2	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.	87.8
3	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.	92.2

Table 12: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on SingleEq.

No.	Trigger Setence	Accuracy
1	Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan to solve the problem step by step.	92.3
2	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.	94.7

Table 13: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on CSQA.

No.	Trigger Setence	Accuracy
1	Let's devise a plan and solve the problem step by step.	67.4
2	Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.	71.9

Table 14: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on StrategyQA.

No.	Trigger Setence	Accuracy
1	Let's devise a plan and solve the problem step by step.	61.5
2	Let's devise a complete plan. Then, let's carry out the plan, solve the problem step by step, and show the answer.	63.0
3	Let's first prepare relevant information and make a plan. Then, let's answer the question step by step (pay attention to commonsense and logical coherence).	65.4

Table 15: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on Last Letters.

No.	Trigger Setence	Accuracy
1	Let's devise a plan and solve the problem step by step.	75.2

Table 16: Performance comparison of prompts used in Step 1 of Zero-shot-PS+ prompting with `text-davinci-003` on Coin Flip.

No.	Trigger Setence	Accuracy
1	Let's devise a complete plan. Then, let's carry out the plan, solve the problem step by step, and show the answer.	70.6
2	Let's first devise a plan, then solve the problem step by step.	72.6
3	Let's first devise a plan, then solve the problem step by step.(Distinguish between tail up and head up)	84.0
4	Let's first understand the problem and devise a complete plan.Then, let's carry out the plan, reason problem step by step. Every step answer the subquestion "does the person flip and what is the coin current state?", and according to the last coin state, give the final answer (pay attention that the coin turns state at every flip). Plan: Step 1:	94.2
5	Let's first understand the problem and devise a complete plan. Then, let's carry out the plan, reason problem step by step. Every step answer the subquestion "does the person flip and what is the coin current state?", and according to the coin last state give the final answer(pay attention to every flip, the coin turns state). Plan: Step 1:	99.0
6	Let's first understand the problem and devise a complete plan. Then, let's carry out the plan and reason problem step by step. Every step answer the subquestion, "does the person flip and what is the coin's current state?". According to the coin's last state, give the final answer (pay attention to every flip and the coin's turning state).	99.6

Table 17: Example outputs by Zero-shot-PS+ for AQuA.

Q: The average wages of a worker during a fortnight comprising of 15 consecutive working days was $90 per day. During the first 7 days, his average wage was $87 per day and the average wage during the last 7 days was $92 per day. What was his wage on the 8th day? Answer Choices: (A) $83 (B) $92 (C) $90 (D) $97 (E) $104

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Variables:
Average wage during the first 7 days = 87
Average wage during the last 7 days = 92
Average wage during the fortnight = 90
Plan:
Calculate the total wages earned during
the first 7 days.
Calculate the total wages earned during
the last 7 days.
Calculate the total wages earned during
the fortnight.
Calculate the wage on the 8th day.
Calculation:
Total wages earned during the first 7 days = 7 x 87 = 609
Total wages earned during the last 7 days = 7 x 92 = 644
Total wages earned during the fortnight = 15 x 90 = 1350
Wage on the 8th day = (1350 - 609 - 644) / 1 = 97
Answer: (D) $97.

Q: If 27 bottles of soda cost A cents, how much will B bottles cost in dollars? Answer Choices: (A) AB/2700 (B) 27/AB (C) AB/270 (D) 2700/AB (E) 100AB/27

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Given:
27 bottles of soda cost A cents
B bottles of soda
Plan:
1. Convert A cents to dollars
2. Calculate the cost of B bottles of soda
Solution:
1. Convert A cents to dollars: A cents = A/100 dollars
2. Calculate the cost of B bottles of soda: B bottles of soda = (A/100) * (B/27) = AB/2700 dollars
Answer: (A) AB/2700

Table 18: Example outputs by Zero-shot-PS+ for GSM8K.

Q: James decides to run 3 sprints 3 times a week. He runs 60 meters each sprint. How many total meters does he run a week?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Given:
James runs 3 sprints 3 times a week.
Each sprint is 60 meters.
Plan:
We need to calculate the total meters run by James in a week.
Calculation:
Total number of sprints run by James in a week = 3 sprints x 3 times = 9 sprints
Total meters run by James in a week = 9 sprints x 60 meters = 540 meters
Answer:
James runs 540 meters in a week.

Q: In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Given:
Total number of students = 20
Percentage of students enrolled in contemporary dance = 20%
Percentage of students enrolled in jazz dance = 25%
Plan:
1. Calculate the number of students enrolled in contemporary dance.
2. Calculate the number of students remaining after subtracting the number of students enrolled in contemporary dance.
3. Calculate the number of students enrolled in jazz dance.
4. Calculate the number of students enrolled in hip-hop dance.
5. Calculate the percentage of students enrolled in hip-hop dance.
Calculation:
1. Number of students enrolled in contemporary dance = 20% of 20 = 20% × 20 = 4
2. Number of students remaining after subtracting the number of students enrolled in contemporary dance = 20 - 4 = 16
3. Number of students enrolled in jazz dance = 25% of 16 = 25% × 16 = 4
4. Number of students enrolled in hip-hop dance = 16 - 4 = 12
5. Percentage of students enrolled in hip-hop dance = 12/20 × 100% = 60%
Answer:
60% of the entire students enrolled in hip-hop dance.

Table 19: Example outputs by Zero-shot-PS+ for MultiArith.

Q: The school cafeteria ordered 42 red apples and 7 green apples for students lunches. But, if only 9 students wanted fruit, how many extra did the cafeteria end up with?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Variables:
Red apples = 42
Green apples = 7
Students wanting fruit = 9
Plan:
Calculate the total number of apples ordered by the cafeteria
Subtract the total number of apples ordered by the cafeteria from the number of students wanting fruit
Calculation:
Total number of apples ordered by the cafeteria = 42 + 7 = 49
Answer:
Extra apples = 49 - 9 = 40

Q: Carol and her mom were picking carrots from their garden. Carol picked 29 and her mother picked 16. If only 38 of the carrots were good, how many bad carrots did they have?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Variables:
Carol: 29
Mom: 16
Good Carrots: 38
Plan:
We can calculate the number of bad carrots by subtracting the number of good carrots from the total number of carrots picked.
Calculation:
Total Carrots Picked = Carol + Mom = 29 + 16 = 45
Bad Carrots = Total Carrots Picked - Good Carrots = 45 - 38 = 7
Answer:
There were 7 bad carrots.

Table 20: Example outputs by Zero-shot-PS+ for SVAMP.

Q: Mom buys 70 white t - shirts in total. If white t - shirts can be purchased in packages and mom buys 14 packages. How many white t - shirts does each package have?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make and device a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Variables:
Total number of white t-shirts = 70
Number of packages = 14
Plan:
We can use the equation:
Total number of white t-shirts = Number of packages x Number of t-shirts per package
Calculation:
Number of t-shirts per package = 70/14 = 5
Answer:
Each package has 5 white t-shirts.

Q: Jesse's rooms are 19 feet long and 18 feet wide. If she has 20 rooms in all. How much carpet does she need to cover the floors of all rooms?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and make and device a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Relevant Variables:
Number of rooms: 20
Length of each room: 19 feet
Width of each room: 18 feet
Plan:
Step 1: Calculate the area of each room.
Step 2: Multiply the area of each room by the number of rooms.
Step 3: Show the answer.
Calculation:
Step 1: Area of each room = Length x Width = 19 feet x 18 feet = 342 square feet
Step 2: Total area of all rooms = Area of each room x Number of rooms = 342 square feet x 20 rooms = 6840 square feet
Answer: Jesse needs 6840 square feet of carpet to cover the floors of all rooms.

Table 21: Example outputs by Zero-shot-PS+ for AddSub.

Q: There were 28 bales of hay in the barn. Tim stacked bales in the barn today. There are now 54 bales of hay in the barn. How many bales did he store in the barn ?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Given:
Initial number of bales of hay in the barn = 28
Number of bales of hay in the barn after Tim stacked them = 54
Plan:
Calculate the number of bales of hay Tim stored in the barn by subtracting the initial number of bales of hay from the number of bales of hay after Tim stacked them.
Calculation:
Number of bales of hay Tim stored in the barn = 54 - 28 = 26
Answer:
Tim stored 26 bales of hay in the barn.

Q: There were 6 roses in the vase. Mary cut some roses from her flower garden. There are now 16 roses in the vase. How many roses did she cut ?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Given:
Initial number of roses in the vase = 6
Final number of roses in the vase = 16
Plan:
Subtract the initial number of roses from the final number of roses to calculate the number of roses Mary cut.
Calculation:
Number of roses Mary cut = 16 - 6 = 10
Answer:
Mary cut 10 roses from her flower garden.

Table 22: Example outputs by Zero-shot-PS+ for SingleEq.

Q: After eating at the restaurant, Sally, Sam, and Alyssa decided to divide the bill evenly. If each person paid 45 dollars, what was the total of the bill ?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Problem: After eating at the restaurant, Sally, Sam, and Alyssa decided to divide the bill evenly. If each person paid 45 dollars, what was the total of the bill?
Variables:
- Total of the bill (T)
- Number of people (N)
- Amount paid by each person (A)
Numerals:
- N = 3
- A = 45
Plan:
- Calculate the total of the bill (T) by multiplying the number of people (N) by the amount paid by each person (A).
Calculation:
T = N x A
T = 3 x 45
T = 135
Answer: The total of the bill was 135 dollars.

Q: Mary is baking a cake. The recipe wants 8 cups of flour. She already put in 2 cups. How many more cups does she need to add ?

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a plan. Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numeral calculation and commonsense), solve the problem step by step, and show the answer.

Output:
Mary needs to add 8 cups of flour to the cake. She has already put in 2 cups.
Plan:
Subtract the amount of flour Mary has already put in from the total amount of flour needed.
Calculation:
8 cups (total amount of flour needed) - 2 cups (amount of flour Mary has already put in) = 6 cups (amount of flour Mary needs to add)
Answer: Mary needs to add 6 more cups of flour.

Table 23: Example outputs by Zero-shot-PS+ for CommonsenseQA.

Q: What is a place that has a bench nestled in trees? Answer Choices: (A) state park (B) bus stop (C) bus depot (D) statue (E) train station

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
The question is asking for a place that has a bench nestled in trees. The answer choices are (A) state park, (B) bus stop, (C) bus depot, (D) statue, and (E) train station.
The answer is (A) state park. State parks are typically located in natural areas with trees, and they often have benches for visitors to sit and enjoy the scenery.

Q: Where is a good idea but not required to have a fire extinguisher? Answer Choices: (A) school bus (B) boat (C) house (D) hospital (E) school

A: Let's first understand the problem, extract relevant variables and their corresponding numerals, and devise a complete plan.Then, let's carry out the plan, calculate intermediate variables (pay attention to correct numerical calculation and commonsense), solve the problem step by step, and show the answer.

Output:
The question is asking for a good idea to have a fire extinguisher, but it is not required. The answer choices are (A) school bus, (B) boat, (C) house, (D) hospital, and (E) school.
The best answer is (C) house. Having a fire extinguisher in a house is a good idea, but it is not required. Fire extinguishers are not typically required in school buses, boats, hospitals, or schools.

Table 24: Example outputs by Zero-shot-PS+ for StrategyQA.

Q: Does the country that received the most gold medals during the 1976 Olympics still exist?

A: Let's first prepare relevant information and make a plan. Then, let's answer the question step by step (pay attention to commonsense and logical coherence).

Output:
Relevant information:
- The 1976 Olympics were held in Montreal, Canada.
- The country that received the most gold medals during the 1976 Olympics was the Soviet Union.
Plan:
1. Determine if the Soviet Union still exists.
2. If the Soviet Union still exists, answer the question.
3. If the Soviet Union does not exist, determine which countries it has been replaced by. Answer:
1. The Soviet Union no longer exists.
2. No, the country that received the most gold medals during the 1976 Olympics does not still exist.
3. The Soviet Union has been replaced by 15 independent countries, including Russia, Ukraine, Belarus, and Kazakhstan.

Q:Is the Mona Lisa in the same museum as the Venus de Milo?

A: Let's first prepare relevant information and make a plan. Then, let's answer the question step by step (pay attention to commonsense and logical coherence).

Output:
Relevant information:
Mona Lisa: Located in the Louvre Museum in Paris, France
Venus de Milo: Located in the Louvre Museum in Paris, France
Plan:
Step 1: Identify the locations of the Mona Lisa and the Venus de Milo
Step 2: Compare the locations
Answer:
Yes, the Mona Lisa and the Venus de Milo are both located in the Louvre Museum in Paris, France.

Table 25: Example outputs by Zero-shot-PS+ for Last Letters.

Q: Take the last letters of each words in "Lino Mariel Aditya Elisabeth" and concatenate them

A:Let's devise a plan and solve the problem step by step.

Output:
Step 1: Take the last letters of each word in the given string.
Lino: o
Mariel: l
Aditya: a
Elisabeth: h
Step 2: Concatenate the last letters.
Answer: olah

Q: Take the last letters of each words in "Jacky Socorro Mark Wanda" and concatenate them

A:Let's devise a plan and solve the problem step by step.

Output:
Step 1: Take the last letters of each word.
Jacky: y
Socorro: o
Mark: k
Wanda: a
Step 2: Concatenate the letters.
Answer: yoka

Table 26: Example outputs by Zero-shot-PS+ for Coin Flip.

Q: A coin is heads up. Walter does not flip the coin. Mikayla does not flip the coin. Larry does not flip the coin. Ryan does not flip the coin. Is the coin still heads up? Note that "flip" here means "reverse".

A: Let's first understand the problem and devise a complete plan. Then, let's carry out the plan and reason problem step by step. Every step answer the subquestion, "does the person flip and what is the coin's current state?". According to the coin's last state, give the final answer (pay attention to every flip and the coin's turning state).

Output:
Plan:
Step 1: Walter does not flip the coin.
Step 2: Mikayla does not flip the coin.
Step 3: Larry does not flip the coin.
Step 4: Ryan does not flip the coin.
Answer: Yes, the coin is still heads up.
Explanation:
Step 1: Walter does not flip the coin. Therefore, the coin is still heads up.
Step 2: Mikayla does not flip the coin. Therefore, the coin is still heads up.
Step 3: Larry does not flip the coin. Therefore, the coin is still heads up.
Step 4: Ryan does not flip the coin. Therefore, the coin is still heads up.
Conclusion: Yes, the coin is still heads up.

Q: A coin is heads up. Lorena does not flip the coin. Shana does not flip the coin. Priscilla does not flip the coin. Summer does not flip the coin. Is the coin still heads up? Note that "flip" here means "reverse"

A: Let's first understand the problem and devise a complete plan. Then, let's carry out the plan and reason problem step by step. Every step answer the subquestion, "does the person flip and what is the coin's current state?". According to the coin's last state, give the final answer (pay attention to every flip and the coin's turning state).

Output:
Plan:
Step 1: Lorena does not flip the coin.
Step 2: Shana does not flip the coin.
Step 3: Priscilla does not flip the coin.
Step 4: Summer does not flip the coin.
Answer:
Step 1: Lorena does not flip the coin. Therefore, the coin is still heads up.
Step 2: Shana does not flip the coin. Therefore, the coin is still heads up.
Step 3: Priscilla does not flip the coin. Therefore, the coin is still heads up.
Step 4: Summer does not flip the coin. Therefore, the coin is still heads up.
Final Answer: Yes, the coin is still heads up.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
section 7

☑ A2. Did you discuss any potential risks of your work?
section 8

☑ A3. Do the abstract and introduction summarize the paper's main claims?
abstract and section 1

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

section 3 and 4

☑ B1. Did you cite the creators of artifacts you used?
section 3 and 4

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
section 8

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
section 8

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
section 8

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
section 3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
section 3

C ☑ Did you run computational experiments?

section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
section 3

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
section 3

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
section 4

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
section 3

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.