# INDUCT-LEARN: Short Phrase Prompting with Instruction Induction

**Po-Chun Chen**[1] **Sheng-Lun Wei**[1] **Hen-Hsen Huang**[2] **Hsin-Hsi Chen**[1]
[1]National Taiwan University, Taiwan
[2]Academia Sinica, Taiwan
{pcchen,weisl}@nlg.csie.ntu.edu.tw
hhhuang@iis.sinica.edu.tw hhchen@ntu.edu.tw

## Abstract

Large Language Models (LLMs) have demonstrated capability in "instruction induction," generating instructions from demonstrations (input-output pairs). However, existing methods often rely on large datasets or numerous examples, which is impractical and costly in real-world scenarios. In this work, we propose a low-cost, task-level framework called INDUCT-LEARN. It induces pseudo instructions from a few demonstrations and a short phrase, adding a CoT process into existing demonstrations. When encountering new problems, the learned pseudo instructions and demonstrations with the pseudo CoT process can be combined into a prompt to guide the LLM's problem-solving process. We validate our approach on the BBH-Induct and Evals-Induct datasets, and the results show that the INDUCT-LEARN framework outperforms state-of-the-art methods. We also exhibit cross-model adaptability and achieve superior performance at a lower cost compared to existing methods.

## 1 Introduction

Although Large Language Models (LLMs) excel at adapting to unseen tasks through in-context learning (Brown et al., 2020; Wei et al., 2022a; Chen et al., 2023; Wei et al., 2023), their inferential process is often implicit, making it challenging to explicitly understand the underlying mechanisms. Honovich et al. (2023) reveal LLMs' potential for "instruction induction," where they infer rules from input-output pairs to create natural language instructions, offering hope for mitigating the challenge of implicit reasoning.

Following these insights, Zhou et al. (2022), Sun et al. (2023), Zhang et al. (2023), Chen et al. (2024), Hsieh et al. (2024) innovated in utilizing input-output pairs for generating diverse prompts, aimed at selecting optimal prompts for downstream models. These works generally assume the availability of substantial external resources, like extensive training datasets, for acquiring numerous demonstrationsa process that can incur significant costs for producing prompts. Moreover, most of these

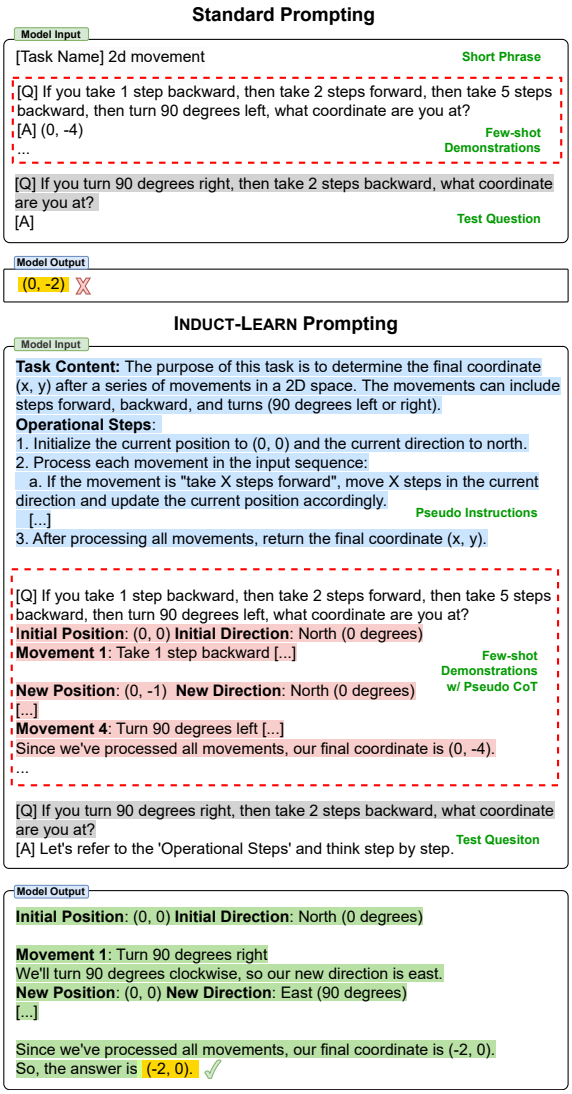



Figure 1: Comparison of model output from standard prompting and INDUCT-LEARN prompting. INDUCT-LEARN framework generates pseudo instructions and pseudo CoT demonstrations to help the model answer questions more effectively.



*Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 5204–5231
November 12-16, 2024 

methods require an initial prompt (instruction) to perform prompt optimization.

However, this differs from the situation of general users. Typical users usually only possess limited data (a few examples) and are unwilling to invest significant costs in prompt optimization using these methods. Moreover, these approaches optimize prompts for specific target models, and the optimized prompts may not be effective for other models. Furthermore, general users often lack sufficient prompting skills to communicate with LLMs (Desmond and Brachman, 2024; Mishra and Nouri, 2022; Zamfirescu-Pereira et al., 2023), making it challenging to write good initial instructions.

We propose a low-cost, task-level INDUCT-LEARN framework that can effectively improve performance using only a few demonstrations and a very simple short phrase to limit the scope. This framework first induces task-level pseudo instructions based on the provided task-specific demonstrations and corresponding short phrase. These pseudo instructions encapsulate the *Task Content*, *Input and Output Format*, and *Operational Steps*. Then, similar to human inductive learning, the model is asked to practice and infer specific examples using its self-induced pseudo instructions, rewriting the demonstrations into pseudo CoT examples. When encountering new questions, the learned pseudo instruction and CoT demonstrations can be combined into a prompt to guide the LLM's thinking and solve the problem.

Our validation on custom datasets BBH-Induct and Evals-Induct shows that our proposed low-cost INDUCT-LEARN framework achieves performance comparable to carefully crafted human instructions and significantly outperforms state-of-the-art frameworks. Notably, powerful LLMs using the INDUCT-LEARN framework can even surpass the performance of human-written instructions. Furthermore, our experiments demonstrate that the INDUCT-LEARN framework exhibits cross-model adaptability, meaning that the prompt generated by one model can be effectively used by other models to improve their performance. In terms of cost, we can outperform state-of-the-art frameworks at a lower cost.

## 2 INDUCT-LEARN Framework

This section details the design of the INDUCT-LEARN framework for generating pseudo instructions and chain-of-thought (CoT) processes.

### 2.1 INDUCT Stage

The objective of this stage is to utilize the inductive capabilities of LLMs to generate comprehensive instructions from demonstrations. Given a set of demonstrations containing $N$ samples:

$$D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\} \quad (1)$$

where $x_i$ denotes an input, and $y_i$ denotes the corresponding output.

Then, we provide additional information with a short phrase $I$. Let the LLM read the samples, induce from them, and generate the instruction prompt. The process can be expressed as follows:

$$P_{\text{INDUCT}} = \mathcal{LM}(I, D \mid P_\phi) \quad (2)$$

where $P_\phi$ is the meta prompt we design to guide the LLM to simulate an expert's analysis, inducing instructions from the given demonstrations $D$. $P_{\text{INDUCT}}$ is the outcome of this step, containing four important components: *Task Content*, which is a description of this task; *Input Format* and *Output Format*, which describe the format of the input-output pairs $(x_i, y_i)$, to guide the LLM in precisely understanding the input and generating the accurate output; and *Operational Steps*, which provide step-by-step guidance to answer the given input $x_i$.

### 2.2 LEARN Stage

This stage utilizes the instruction $P_{\text{INDUCT}}$ generated in the previous stage to produce high-quality demonstrations with a detailed Chain-of-Thought (CoT) process. Specifically, there are several steps in the LEARN stage. First, we take the input $x_i$ from each demonstration $(x_i, y_i) \in D$ along with $P_{\text{INDUCT}}$. We can formulate the process as:

$$(\hat{y}_i, c_i) = \mathcal{LM}(x_i \mid P_{\text{INDUCT}}) \quad \forall\, i \in [1, N] \quad (3)$$

where $c_i$ is the CoT process generated by the LLM and $\hat{y}_i$ is the corresponding generated answer.

Subsequently, we compare $\hat{y}_i$ with the ground-truth $y_i$ and filter out the incorrect ones. The filtering process can be expressed as:

$$D_{\text{LEARN}} = \{(x_i, \hat{y}_i, c_i) \mid \hat{y}_i = y_i, \quad \forall\, i \in [1, N]\} \quad (4)$$

where $D_{\text{LEARN}}$ is the demonstration set that has been verified as correct, containing the input $x_i$, the generated answer $\hat{y}_i$, and the CoT process $c_i$.

Then, we concatenate the first $k$ samples from $D_{\text{LEARN}}$ and generate $P_{\text{LEARN}}$. The order of concatenation will be $x_j, c_j, \hat{y}_j$. This process can be formulated as follows:

$$P_{\text{LEARN}} = \text{concat}(x_j, c_j, \hat{y}_j) \quad \forall\, j \in [1, k] \quad (5)$$

where $P_{\text{LEARN}}$ is the prompt generated by concatenating the input $x_j$, the Chain-of-Thought process $c_j$, and the generated answer $\hat{y}_j$ for the first $k$ samples from $D_{\text{LEARN}}$.

Finally, we can easily create the final prompt by concatenating($\oplus$) $P_{\text{INDUCT}}$ and $P_{\text{LEARN}}$, which can be expressed as:

$$P_{\text{INDUCT\_LEARN}} = P_{\text{INDUCT}} \oplus P_{\text{LEARN}} \quad (6)$$

### 2.3 Inference Stage

The goal of this stage is to leverage $P_{\text{INDUCT\_LEARN}}$ to enhance model performance on new inputs for the same task. Given an input $x'$ where $x' \notin D$, we can use our $P_{\text{INDUCT\_LEARN}}$ to generate the result:

$$y' = \mathcal{LM}(x' \mid P_{\text{INDUCT\_LEARN}}) \quad (7)$$

## 3 Experiments

### 3.1 Models

In our experiments, we conduct evaluations on eight instruction-tuned models across three series, including both open-source models and commercial APIs. For open-source models, we adopt Mistral-7B (Jiang et al., 2023), Mixtral-8x7B (Jiang et al., 2024), Mixtral-8x22B, Llama 3 8B, and Llama 3 70B (Meta, 2024). On the commercial side, we select Gemini 1.0 Pro (Gemini Team, 2023), Gemini 1.5 Flash, and Gemini 1.5 Pro (Gemini Team, 2024).

### 3.2 Datasets

To evaluate the capability of LLMs in inferring instructions from demonstrations without explicit hints, we adopt BBH-Induct and Evals-Induct. Both datasets are sufficiently challenging and do not perform well when LLMs are used in zero-shot settings without given instructions.

**BBH-Induct** We introduce a modification of the BIG-Bench-Hard (BBH) dataset (Suzgun et al., 2022) called BBH-Induct, specifically designed to evaluate the instruction induction capabilities of LLMs. Unlike the original BBH dataset, which contains explicit hints in each instance, BBH-Induct removes information that is unrealistic for real-world cases where only input-output pairs are available, and the LLM is expected to solve problems without explicit instructions. To address this issue, we eliminate the prefix and suffix templates in the questions. Due to space constraints, the construction process is shown in Appendix A. Each BBH-Induct task contains approximately 96 to 600 examples, amounting to a total of 5,161 instances. For detailed descriptions of the tasks, refer to the BBH task descriptions in (Suzgun et al., 2022).

**Evals-Induct** We adopt another challenging dataset: Evals-Induct, derived from the OpenAI Evals project[1]. This project collects tasks that even GPT-4 struggles with. We select 24 suitable challenge tasks in Evals-Induct. Similar to BBH-Induct, we include only question-answer pairs without additional information as demonstrations, aiming to use Evals-Induct to evaluate the induction capability of LLMs. LLMs are required to perform inductive reasoning across multiple examples and generate suitable instructions that are beneficial for answering the questions. Each Evals-Induct task has approximately 91 to 378 examples, totaling 3,683 instances. For detailed information on the tasks, please refer to Table 10 in Appendix.

### 3.3 Baselines

We evaluate our approach against several baselines:

**Short Phrase** Model generates the result without complete instructions, using only a short phrase as a hint. We use task titles (task names) as the short phrases, which briefly describe the core content of the task without providing detailed instructions.

**Human** Model generates results based on instructions provided by human experts, which are sourced from the original datasets. These instructions serve as a strong performance benchmark.

**Self-Discover** The state-of-the-art method (Zhou et al., 2024) automatically constructs task-level reasoning structures by selecting, adapting, and implementing from a set of 39 common reasoning modules.

### 3.4 Other Details

**INDUCT-LEARN** In our experiments, we set $N$ to 9 in the INDUCT stage and $k$ to 3 in the LEARN stage. This implies that our INDUCT-LEARN method generates instructions by inducing from

[1]https://github.com/openai/evals

| Model | BBH-Induct | | | | | Evals-Induct | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | SP | Human | Self-Discover | INDUCT (Ours) | Δ | SP | Human | Self-Discover | INDUCT (Ours) | Δ |
| Llama 3 8B | 30.28 | **45.51** | 38.96 | 43.49 | -2.02 | 9.19 | **21.04** | 15.24 | 18.21 | -2.83 |
| Llama 3 70B | 59.45 | **70.30** | 59.37 | 69.93 | -0.37 | 18.83 | 36.52 | 24.22 | **47.91** | 11.39 |
| Mistral 7B | 37.58 | **41.30** | 40.26 | 37.59 | -3.71 | 12.78 | **20.87** | 13.96 | 16.32 | -4.55 |
| Mixtral 8x7B | 38.97 | **48.32** | 40.51 | 44.71 | -3.61 | 12.64 | **25.68** | 21.20 | 25.22 | -0.46 |
| Mixtral 8x22B | 45.57 | 58.61 | 45.91 | **67.01** | 8.4 | 16.96 | 36.83 | 31.92 | **40.50** | 3.67 |
| Gemini 1.0 Pro | 46.21 | **57.64** | 50.38 | 53.55 | -4.09 | 16.13 | **36.64** | 19.58 | 24.23 | -12.41 |
| Gemini 1.5 Flash | 42.62 | 56.69 | 54.27 | **65.09** | 8.4 | 18.17 | 42.87 | 25.95 | **44.44** | 1.57 |
| Gemini 1.5 Pro | 52.44 | 69.45 | 62.81 | **78.96** | 9.51 | 22.42 | 49.38 | 37.50 | **51.13** | 1.75 |

Table 1: Results of instruction generation experiment across models and tasks in zero-shot settings. We evaluate exact match accuracy (%) and calculate the mean of all the tasks of both BBH-Induct and Evals-Induct. The best results of each setting are marked in **bold**, and underlines denote the best performance excluding human instructions. Δ denotes the performance gap between INDUCT(Ours) and human instructions (Ours - Human). SP denotes Short Phrase.

only 9 demonstrations and utilizes up to 3 CoT results in the inference scenario.

**Experimental Setups** To optimize experimental efficiency and stability, we utilize the API services of all eight LLMs, including Google, Groq[2], and Together AI[3]. Details such as model version and safety settings are listed in Appendix B. Due to resource constraints, experiments with Gemini 1.5 Pro and Mixtral 8x22B are conducted on a random sample of 25 instances per task per dataset. All other models are evaluated on the complete datasets.

## 4 Results and Analysis

### 4.1 Quality of Instruction Generation

We conduct experiments in the zero-shot setting to evaluate the quality of different instructions. Given the instructions generated by different methods and the question, we calculate the accuracy of the results that LLMs generate. According to Table 1, $P_{\text{INDUCT}}$ significantly outperforms the SOTA baseline Self-Discover (Zhou et al., 2024) in 15 out of 16 settings, with the only exception being the Mistral 7B model on BBH-Induct dataset. Compared to instructions given by human experts, $P_{\text{INDUCT}}$ achieves a comparable performance, with a minor loss in the BBH-Induct dataset (3 vs. 5) and a tie in the Evals-Induct dataset.

Notably, we observe that our $P_{\text{INDUCT}}$ benefits more as the model's scale increases. For the Llama 3 series, as the model scale increases from 8B to 70B, $P_{\text{INDUCT}}$ improves by 26.44% (from 43.49% to 69.93%), while human instruction improves by 24.79 (from 45.51% to 70.30%). As the model scale grows, $P_{\text{INDUCT}}$ gradually closes the gap with human instruction. Hence, the gap between $P_{\text{INDUCT}}$ and human instruction narrows from -2.02% (45.51% minus 43.49%) to -0.37% (70.3% minus 69.93%). There is a similar trend in the Mistral and Gemini series. For the Mixtral 8x22B, Gemini 1.5 Flash, and Gemini 1.5 Pro models, $P_{\text{INDUCT}}$ even surpassed human instructions. This finding aligns with the observations of Honovich et al. (2023) that larger and more capable models exhibit stronger "instruction induction" abilities. We can conclude that our $P_{\text{INDUCT}}$ is significantly better than human expert instructions in modern powerful LLMs.

### 4.2 Effect of LEARN Stage

In order to measure the effectiveness of our generated pseudo CoT process, we conduct a 3-shot experiment across different models and instructions. The first row of each model in Table 2 represents the 3-shot performance without CoT, and the second row represents the 3-shot performance with our pseudo CoT $P_{\text{LEARN}}$. We can see that in most settings, adding our $P_{\text{LEARN}}$ benefits the performance. Specifically, 23 out of 24 settings in BBH-Induct and 18 out of 24 settings in Evals-Induct show improvement.

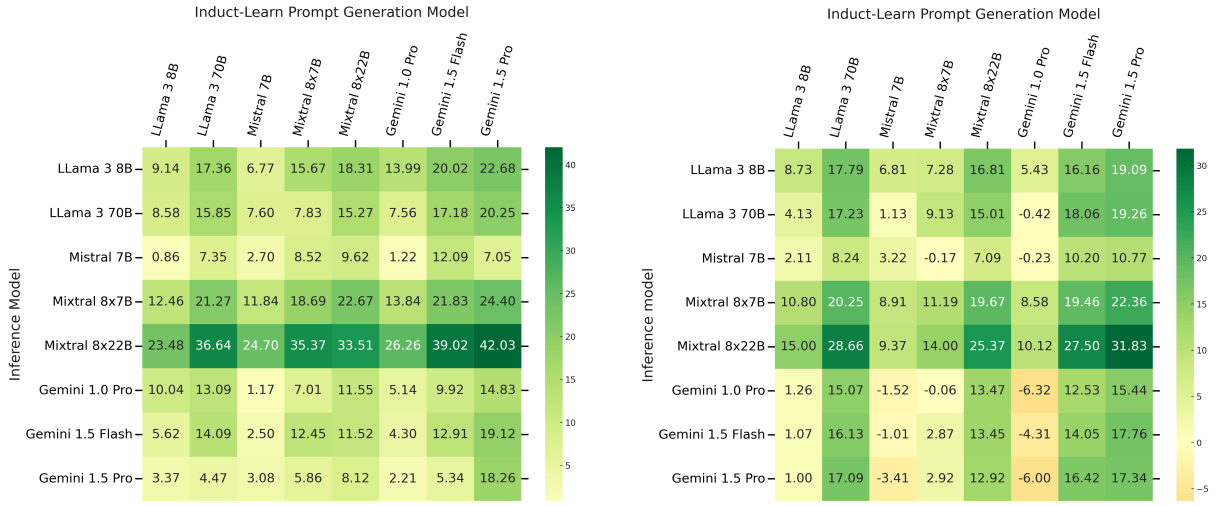
Moreover, compared to SOTA baseline, our $P_{\text{INDUCT-LEARN}}$ outperforms Self-Discover in most of the settings after adding the pseudo CoT process. It only loses in the Llama 3 8B and Mistral 7B models of BBH-Induct and the Gemini 1.0 Pro model of Evals-Induct. This aligns with our previous findings, showing that our method benefits more as the model becomes stronger. Furthermore, Our $P_{\text{INDUCT-LEARN}}$ method also achieved performance comparable to human instructions and even

[2] https://groq.com/

[3] https://www.together.ai/

| Model | BBH-Induct | | | Evals-Induct | | |
|---|---|---|---|---|---|---|
| | Human | Self-Discover | INDUCT (Ours) | Human | Self-Discover | INDUCT (Ours) |
| Llama 3 8B | 37.29 | 30.02 | 35.49 | 16.30 | 15.97 | 19.24 |
| +$P_{\text{LEARN}}$ | 48.59 (+11.30) | **49.79** (+19.77) | 44.62 (+9.13) | 20.23 (+3.93) | 18.52 (+2.55) | **25.23** (+5.99) |
| Llama 3 70B | 72.97 | 64.55 | 68.37 | 42.85 | 37.13 | 46.95 |
| +$P_{\text{LEARN}}$ | **73.68** (+0.71) | 69.00 (+4.45) | 73.35 (+4.98) | 41.69 (-1.16) | 31.49 (-5.64) | **47.73** (+0.78) |
| Mistral 7B | 41.32 | 40.50 | 38.04 | 19.32 | 16.41 | 18.99 |
| +$P_{\text{LEARN}}$ | **47.16** (+5.84) | 44.89 (+4.39) | 44.53 (+6.49) | **22.09** (+2.77) | 16.14 (-0.27) | 20.57 (+1.58) |
| Mixtral 8x7B | 40.99 | 33.11 | 39.78 | 25.17 | 19.93 | 22.25 |
| +$P_{\text{LEARN}}$ | 52.64 (+11.65) | 47.64 (+14.53) | **53.86** (+14.08) | **30.67** (+5.50) | 22.79 (+2.86) | 26.42 (+4.17) |
| Mixtral 8x22B | 34.26 | 41.28 | 50.26 | 26.54 | 22.46 | 34.67 |
| +$P_{\text{LEARN}}$ | 60.75 (+26.49) | 58.09 (+16.81) | **70.55** (+20.29) | 36.17 (+9.63) | 36.04 (+13.58) | **46.00** (+11.33) |
| Gemini 1.0 Pro | 58.67 | 53.83 | 54.41 | 39.73 | 40.14 | 33.37 |
| +$P_{\text{LEARN}}$ | **62.95** (+4.28) | 54.82 (+0.99) | 58.44 (+4.03) | **40.75** (+1.02) | 27.09 (-13.05) | 26.93 (-6.44) |
| Gemini 1.5 Flash | 60.87 | 61.73 | 66.89 | 43.62 | 40.76 | 46.80 |
| +$P_{\text{LEARN}}$ | **73.59** (+12.72) | 61.26 (-0.47) | 73.30 (+6.41) | 45.42 (+1.80) | 34.39 (-6.37) | **49.69** (+2.89) |
| Gemini 1.5 Pro | 76.87 | 62.38 | 76.52 | 46.96 | 40.17 | 49.46 |
| +$P_{\text{LEARN}}$ | 78.43 (+1.56) | 72.52 (+10.14) | **82.72** (+6.20) | 50.58 (+3.62) | 48.63 (+8.46) | **55.92** (+6.46) |

Table 2: The results of using $P_{\text{LEARN}}$ to enhance model performance are presented. Accuracy is expressed as a percentage. Differences compared to 3-shot results are shown in parentheses: improvements are highlighted in blue, indicating performance enhancement due to our method, while declines are marked in red, indicating performance degradation in these cases.




(a) Cross-model Improvement (BBH-Induct)

(b) Cross-model Improvement (Evals-Induct)

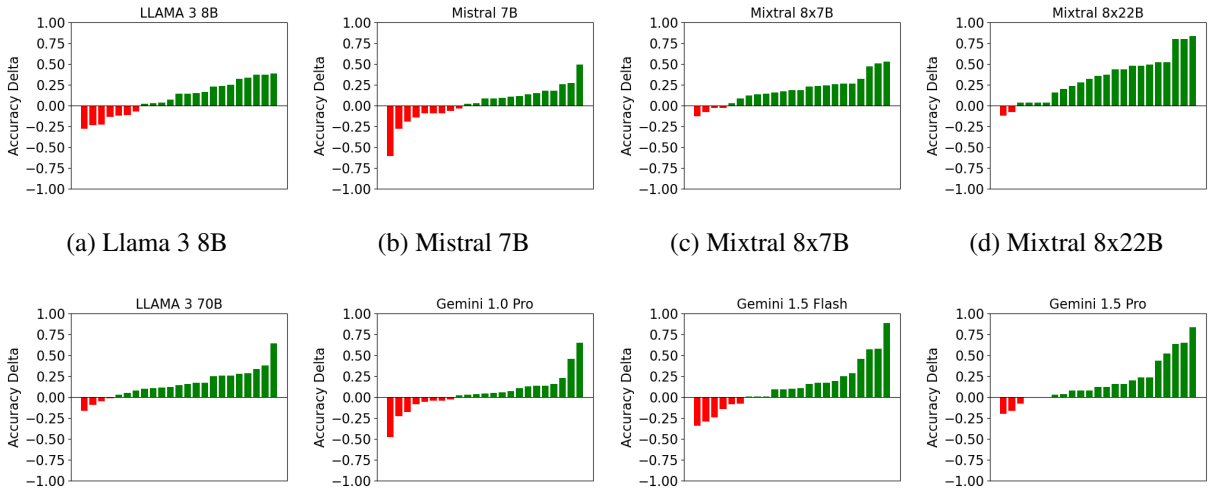
Figure 2: Comparison of INDUCT-LEARN (Ours) and Short Phrase + 3-shot under different cross-model settings, where the prompt induction model and task execution/inference model are different. Accuracy is expressed as a percentage. Darker green indicates greater performance improvement with INDUCT-LEARN (Ours), while darker red indicates greater performance decline. The comparison is conducted on the (a) BBH-Induct and (b) Evals-Induct datasets.

outperformed humans in powerful models, such as the Mixtral 8x22B and Gemini 1.5 Pro models.

### 4.3 Cross-Model Performance

We present the cross-model adaptability and effectiveness of $P_{\text{INDUCT\_LEARN}}$ in Figure 2. Specifically, the figure shows the performance difference (i.e., improvement) between using our $P_{\text{INDUCT\_LEARN}}$ and the original Short Phrase + 3-shot setting. Cross-model refers to using $P_{\text{INDUCT\_LEARN}}$ generated by one model to perform tasks with another model. For example, we use the Gemini 1.5 Pro model to generate $P_{\text{INDUCT\_LEARN}}^{Gemini-1.5-Pro}$, and then use $P_{\text{INDUCT\_LEARN}}^{Gemini-1.5-Pro}$ as the prompt for the Mixtral 8x22B model to perform tasks. Compared to Mixtral 8x22B directly using Short Phrase, the performance improves by 42.03.

**BBH-Induct Dataset** In Figure 2(c), we observe several key phenomena: **1)** $P_{\text{INDUCT\_LEARN}}$ gener-




(a) Llama 3 8B (b) Mistral 7B (c) Mixtral 8x7B (d) Mixtral 8x22B

(e) Llama 3 70B (f) Gemini 1.0 Pro (g) Gemini 1.5 Flash (h) Gemini 1.5 Pro

Figure 3: The accuracy difference distribution across 23 BBH-Induct tasks for 8 models using our Induct-Learn Prompt compared to the Short Phrase under 3-shot settings. Tasks are sorted by accuracy difference from low to high, indicating that tasks to the right benefit more from our method. Improvements are marked in green, while performance declines are highlighted in red.

ated by all models can effectively improve the performance of other models when used as prompts. For example, even using Llama 3 8B to generate $P_{\text{INDUCT\_LEARN}}^{Llama-3-8B}$ as a prompt for Gemini 1.5 Pro (the most powerful model), the performance can be improved by 3.37 compared to directly using Gemini 1.5 Pro to perform tasks (Short Phrase + 3-shot). **2)** As the model becomes stronger, the generated $P_{\text{INDUCT\_LEARN}}$ improves, effectively enhancing the performance of any model.

**Evals-Induct Dataset** Figure 2(d) shows that the Evals-Induct dataset exhibits a similar trend. However, the Evals-Induct dataset specifically collects tasks that are challenging even for GPT-4. Therefore, weaker models (e.g., Llama 3 8B, Mistral 7B, Gemini 1.0 Pro) may not fully understand the questions, and the generated $P_{\text{INDUCT\_LEARN}}$ may sometimes mislead certain models, causing performance degradation. Nevertheless, prompts generated by more powerful models (e.g., Llama 3 70B, Mixtral 8x22B, Gemini 1.5 Flash/Pro) still effectively enhance the performance of other models.

**Cost-effective Strategy** These findings indicate that under limited cost conditions, we can adopt different deployment strategies based on the estimated future frequency of task execution: **1)** For tasks that are estimated to be performed very few times (or even just once), we can choose to use relatively weaker (low-cost) models to generate $P_{\text{INDUCT\_LEARN}}$, and then the original models can use these prompts to perform tasks. This can improve performance with almost no additional cost. **2)** For tasks that are expected to be performed very frequently, since INDUCT_LEARN generates task-level prompts and only needs to be generated once, we can use powerful models to generate INDUCT-LEARN prompts and then have relatively weaker models use these prompts to perform tasks in the inference stage.

### 4.4 Breakdown BBH-Induct Tasks

To gain deeper insights, we segmented 23 BBH-Induct tasks for detailed analysis. Figure 3 presents a head-to-head comparison between using our $P_{\text{INDUCT\_LEARN}}$ framework and not using it (Short Phrase + 3-shot). The results for three series of models are as follows:

**Llama 3 Series** Improved from 16-0-7 (win-tie-lose) with Llama 3 8B to 19-1-3 with Llama 3 70B.

**Mistral 3 Series** Improved from 14-9 (win-lose) with Mistral 7B to 21-0-2 with Mixtral 8x22B.

**Gemini Series** Improved from 15-8 (win-lose) with Gemini 1.0 Pro to 17-0-3 with Gemini 1.5 Pro.

Our experimental results show that as model capabilities enhance, the improvements brought by the INDUCT_LEARN framework become more significant.

| Model | BBH-Induct | Evals-Induct |
|---|---|---|
| Llama 3 8B | 46.50 | 29.89 |
| Llama 3 70B | 73.25 | 62.07 |
| Mistral 7B | 39.51 | 34.10 |
| Mixtral 8x7B | 47.33 | 38.70 |
| Mixtral 8x22B | 65.43 | 51.34 |
| Gemini 1.0 Pro | 51.44 | 32.18 |
| Gemini 1.5 Flash | 70.78 | 55.56 |
| Gemini 1.5 Pro | 74.07 | 60.54 |

Table 3: The Macro Pass Rate of various models on the BBH-Induct and Evals-Induct dataset.

| Model | $N$=3 | $N$=6 | $N$=9 | $N$=12 |
|---|---|---|---|---|
| Gemini 1.5 Flash | 65.71 | 68.23 | 70.41 | **73.10** |
| Mixtral 8x22B | 64.41 | 65.51 | 67.01 | **71.59** |

Table 4: Results for varying numbers of demonstrations $N$ in the INDUCT stage using the BBH-Induct dataset.

| Model | $k$=1 | $k$=2 | $k$=3 | $k$=4 |
|---|---|---|---|---|
| Gemini 1.5 Flash | 72.58 | 74.20 | 74.29 | **74.57** |
| Mixtral 8x22B | 66.36 | 68.20 | 70.55 | **73.68** |

Table 5: Results for varying numbers of demonstrations $k$ in the LEARN stage using the BBH-Induct dataset.

## 5 Discussion and Analysis

### 5.1 Larger LLMs Benefit More from INDUCT-LEARN

Our experiments indicate that larger LLMs benefit more significantly from the proposed INDUCT-LEARN framework compared to smaller models.

**INDUCT stage** This observation aligns with findings by Honovich et al. (2023), who noted that larger models exhibit stronger instruction induction abilities. Therefore, in the INDUCT stage, larger models possess enhanced comprehension capabilities, enabling them to generate higher-quality task instructions $P_{\text{INDUCT}}$ from a limited number of examples, resulting in better performance.

**LEARN stage** In the LEARN stage, as shown in Equation 4, we filter out the incorrect results to obtain $D_{\text{LEARN}}$, the set of demonstrations where the model's predicted output $\hat{y}_j$ matches the ground truth $y_j$. We define the *Pass Rate* (PR) as follows:

$$PR = \frac{\text{count}(D_{\text{LEARN}})}{N} \tag{8}$$

where $N$ is the total number of demonstrations.

The *Macro Pass Rate* for a given model across all tasks is then defined as:

$$Macro\ PR = \frac{1}{M}\sum_{i=1}^{M} PR_i \tag{9}$$

where $M$ is the number of tasks and $PR_i$ is the pass rate of the $i$-th task.

Table 3 shows the macro pass rates across various models and datasets. We can observe that the larger the model in each series, the higher the macro pass rate. Specifically, the strongest models (i.e., Llama 3 70B, Mixtral 8x22B, and Gemini 1.5 Pro) achieve the highest macro pass rate in their respective series. On the other hand, a low macro pass rate may occur in situations where the number of generated CoT demonstrations is less than $k$. This situation weakens the effect of the LEARN stage, causing smaller models to benefit less from our framework. The detailed statistics of this situation are provided in Appendix D.5.

### 5.2 Different Number of Demonstrations

To comprehensively evaluate the impact of the number of demonstrations $N$ during the INDUCT stage and the number of demonstrations $k$ used in the LEARN stage, we conducted additional experiments by sampling instances from each task in the BBH-Induct dataset. For both open-source and closed-source models, we initially considered selecting the models with the highest parameter counts. However, due to the high cost of Gemini 1.5 Pro, we ultimately selected the Mixtral 8x22B and Gemini 1.5 Flash models for our experiments. Specifically, for the Mixtral 8x22B model, we sampled 25 instances per task, consistent with our previous experiments. For the Gemini 1.5 Flash model, we sampled 50 instances per task.

**INDUCT Stage** Table 4 shows that as $N$ increases, performance continuously improves. This indicates that more demonstrations help induce (generate) better instructions, thereby enhancing the model's performance. In practice, considering our paper mainly focuses on giving a limited amount of training data (X, Y pairs without CoT), and best utilize this data to improve LLM's question-answering capabilities, we chose to set N to 9 for subsequent experiments.

**LEARN Stage** Table 5 demonstrates that when $N$ is fixed at 9, an increase in $k$ consistently leads to improved performance across all models. This suggests that a greater number of model-generated demonstrations, augmented with Chain-of-Thought (CoT), can significantly enhance the model's reasoning abilities.

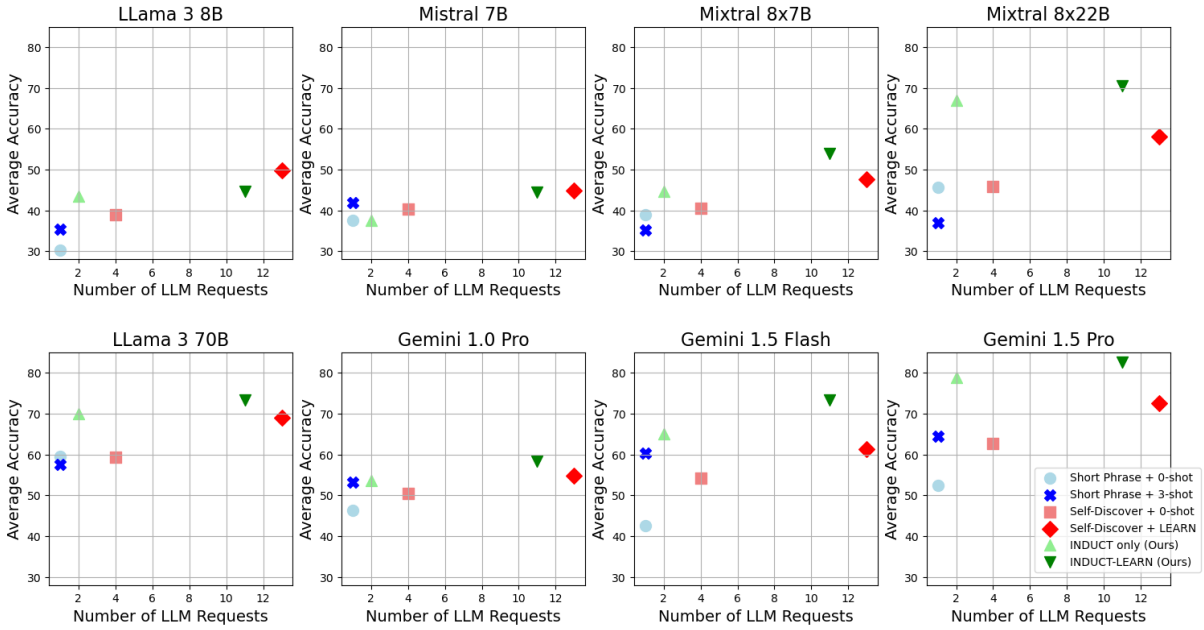



Figure 4: Comparison of average accuracy and number of LLM requests across 8 models under an extreme condition where a single task with only one instance is executed. The configurations compared are Short Phrase + 0-shot, Short Phrase + 3-shot, Self-Discover + 0-shot, Self-Discover + LEARN, INDUCT(Ours), and INDUCT-LEARN on the BBH-Induct dataset.

### 5.3 Meta Prompt Ablation

To prevent LLMs from being sensitive to the given prompt, we conducted an ablation study to assess the consistency of our proposed INDUCT-LEARN framework. We used GPT-4o to rewrite the original meta prompt, and these rewritten prompts are shown in Figures 10 and 13 in Appendix D.6. We selected the Gemini 1.5 Flash model as the representative and sampled 50 data points from each task in the BBH-Induct dataset for analysis. These experiments were conducted using the INDUCT + 0-shot setup. Table 6 indicates that different meta-prompts do not significantly impact the performance of our INDUCT Stage.

| Meta Prompt | BBH-Induct |
|---|---|
| Ours | 70.41 |
| Rewritten 1 | 68.75 |
| Rewritten 2 | 69.39 |
| Rewritten 3 | 72.52 |
| Average of Rewrittens | 70.22 |

Table 6: The results of the ablation study on the meta prompt. We evaluate the exact match accuracy (%) and calculate the mean across all tasks.

## 6 Cost Analysis

In scenarios where the same task is executed multiple times, prompt generation costs can be largely ignored. To comprehensively assess potential costs, we analyze the worst-case scenario where only a single task is executed once, making the cost of prompt generation a significant factor. Given that different tasks and models have varying token amounts, and even different models may use different tokenization methods, we chose to standardize variables by analyzing costs based on the number of requests. For detailed request calculations, please refer to Table 14 in Appendix.

As shown in Figure 5, our Induct-Learn method demonstrates significant efficiency and accuracy advantages. For the 0-shot and $P_{\text{LEARN}}$ settings, INDUCT-LEARN requires fewer requests than Self-Discover but provides higher average accuracy in almost all cases. In almost all settings (except Mistral 7B), our $P_{\text{INDUCT}}$ only requires one additional request to significantly improve performance over a single request for Short Phrase + 0-shot/3-shot. Notably, our method's advantages increase with stronger models. For example, with powerful models like Llama 3 70B, Mixtral 8x22B, Gemini 1.5 Flash, and Gemini 1.5 Pro, our $P_{\text{INDUCT}}$ (requiring only 2 requests) can outperform Self-Discover with $P_{\text{LEARN}}$ (requiring 13 requests). These results indicate that our Induct-Learn method improves accuracy without significantly increasing computational costs.

## 7 Related Work

**Inductive Abilities of LLMs** Inductive reasoning, a key cognitive ability, has been extensively studied in the context of LLMs (Liu et al., 2024; Yu et al., 2023). Yang et al. (2022) introduced a task for inferring natural language rules from facts, demonstrating LLMs' inductive capabilities. Zhu et al. (2023) leveraged these capabilities to generate rules from examples, which were then used as input prompts. Qiu et al. (2024) and Han et al. (2023) confirmed that LLMs, particularly GPT-4, can derive rules from input-output pairs, performing on par with humans on classic property induction tasks. Honovich et al. (2023) showcased LLMs generating instructions from input-output examples, termed "instruction induction." Our experimental results support the observation that the inductive ability of LLMs increases with their scale, with larger models generating better instructions that lead to greater performance improvements in downstream tasks.

**Instruction Generation** Building on Honovich et al. (2023), several studies have focused on generating instructions using LLMs. Some researchers proposed methods to train LLMs to generate instructions from input examples and labels, such as reverse learning (Ye et al., 2022), generating and ranking candidate instructions (Zhou et al., 2022; Zhang et al., 2023), and iteratively refining instructions by combining input-output examples (Sun et al., 2023; Chen et al., 2024). Additionally, some studies have proposed methods to iteratively optimize prompts, such as using genetic evolution (Xu et al., 2022; Fernando et al., 2023; Yang and Li, 2023; Cui et al., 2024; Guo et al., 2024) or gradient-based approaches (Tang et al., 2024; Pryzant et al., 2023). However, these methods often require substantial computational resources. In contrast, INDUCT-LEARN focuses on generating a single instruction and its CoT, improving efficiency and reducing resource usage. Our results indicate that our method's prompts possess cross-model capabilities rather than being tailored for specific models.

**LLM Reasoning** CoT prompts enable large language models to generate intermediate reasoning steps, guiding and explaining solutions (Wei et al., 2022b; Kojima et al., 2022; Nye et al., 2021). Zhang et al. (2022) leveraged datasets to select diverse samples and generate CoT examples. Wang et al. (2022) extracted multiple reasoning chains during decoding and chose the most consistent one. (Wang et al., 2023) introduced a planning step before answering questions, followed by solving the problem according to the plan. In contrast, our approach operates at the task level. The INDUCT stage generates $P_{\text{INDUCT}}$ from a small number of demonstrations for a given task, followed by the LEARN stage, which converts the demonstrations into CoT examples. As a result, our method does not require a training set, unlike the aforementioned approaches. Moreover, when encountering new questions, only a single request is needed.

## 8 Conclusion

In this work, we proposed The INDUCT-LEARN framework that leverages the instructional induction capabilities of LLMs to enhance model performance in low-resource settings. By inducing pseudo instructions from limited input-output pairs and a short guiding phrase, the framework guides the model in rewriting demonstrations into pseudo chain-of-thought examples. When encountering new problems, the INDUCT-LEARN prompt, which includes the learned pseudo instructions and pseudo CoT demonstrations, guides the LLMs reasoning and problem-solving approach. Empirical evaluations on two datasets demonstrate the effectiveness of our framework, outperforming state-of-the-art methods. The cross-model adaptability exhibited by the INDUCT-LEARN framework further highlights its practical utility, as prompts generated by one model can effectively enhance the performance of others. Our work underscores the potential of harnessing the instructional induction capabilities of LLMs to develop cost-effective solutions for improving model performance in real-world, low-resource scenarios. We open up new avenues for exploring the integration of inductive learning strategies with the capabilities of LLMs, paving the way for more efficient and accessible approaches to enhancing model performance.

## 9 Limitations

**Dependence on Instruction-Following Models** LLMs employing the INDUCT-LEARN framework require instruction tuning. This allows them to follow the directives of the Meta Prompt during the INDUCT stage, inducting $P_{\text{Induct}}$ from input-output pairs. Models lacking instruction tuning are unlikely to complete this task successfully.

**Necessity of Large-Scale Models** Our experimental results indicate that the ability for instruction induction improves with the strength of the model. To achieve results surpassing human instruction, more powerful LLMs are required to generate $P_{\text{INDUCT-LEARN}}$.

## Acknowledgements

This work was supported by National Science and Technology Council, Taiwan, under grants NSTC 112-2634-F-002-005 -, and Ministry of Education (MOE) in Taiwan, under grants NTU-113L900901.

## References

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, T. J. Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *ArXiv*, abs/2005.14165.

Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. 2024. InstructZero: Efficient instruction optimization for black-box large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6503–6518. PMLR.

Wei-Lin Chen, Cheng-Kuang Wu, Yun-Nung Chen, and Hsin-Hsi Chen. 2023. Self-ICL: Zero-shot in-context learning with self-generated demonstrations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15651–15662, Singapore. Association for Computational Linguistics.

Wendi Cui, Jiaxin Zhang, Zhuohang Li, Hao Sun, Damien Lopez, Kamalika Das, Bradley Malin, and Kumar Sricharan. 2024. Phaseevo: Towards unified in-context prompt optimization for large language models. *ArXiv*, abs/2402.11347.

Michael Desmond and Michelle Brachman. 2024. Exploring prompt engineering practices in the enterprise. *ArXiv*, abs/2403.08950.

Chrisantha Fernando, Dylan S. Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution. *ArXiv*, abs/2309.16797.

Google Gemini Team. 2023. Gemini: A family of highly capable multimodal models. *ArXiv*, abs/2312.11805.

Google Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. Accessed: 2024-05-20.

Qingyan Guo, Rui Wang, Junliang Guo, Bei Li, Kaitao Song, Xu Tan, Guoqing Liu, Jiang Bian, and Yujiu Yang. 2024. Connecting large language models with evolutionary algorithms yields powerful prompt optimizers. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Simon J. Han, Keith Ransom, Andrew Perfors, and Charles Kemp. 2023. Inductive reasoning in humans and large language models. *Cognitive Systems Research*, 83.

Or Honovich, Uri Shaham, Samuel R. Bowman, and Omer Levy. 2023. Instruction induction: From few examples to natural language task descriptions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1935–1952, Toronto, Canada. Association for Computational Linguistics.

Cho-Jui Hsieh, Si Si, Felix Yu, and Inderjit Dhillon. 2024. Automatic engineering of long prompts. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10672–10685, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L'elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts. *ArXiv*, abs/2401.04088.

Albert Qiaochu Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L'elio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *ArXiv*, abs/2310.06825.

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

Emmy Liu, Graham Neubig, and Jacob Andreas. 2024. An incomplete loop: Deductive, inductive, and abductive learning in large language models. *ArXiv*, abs/2404.03028.

Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date.

Swaroop Mishra and Elnaz Nouri. 2022. Help me think: A simple prompting strategy for non-experts to create customized content with models. In *Annual Meeting of the Association for Computational Linguistics*.

Maxwell Nye, Anders Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show your work: Scratchpads for intermediate computation with language models. *ArXiv*, abs/2112.00114.

Reid Pryzant, Dan Iter, Jerry Li, Yin Tat Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Conference on Empirical Methods in Natural Language Processing*.

Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, and Xiang Ren. 2024. Phenomenal yet puzzling: Testing inductive reasoning capabilities of language models with hypothesis refinement. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Hong Sun, Xue Li, Yi Xu, Youkow Homma, Qinhao Cao, Min man Wu, Jian Jiao, and Denis Xavier Charles. 2023. Autohint: Automatic prompt optimization with hint generation. *ArXiv*, abs/2307.07415.

Mirac Suzgun, Nathan Scales, Nathanael Scharli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed Huai hsin Chi, Denny Zhou, and Jason Wei. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Annual Meeting of the Association for Computational Linguistics*.

Xinyu Tang, Xiaolei Wang, Wayne Xin Zhao, Siyuan Lu, Yaliang Li, and Ji-Rong Wen. 2024. Unleashing the potential of large language models as prompt optimizers: An analogical analysis with gradient-based model optimizers. *ArXiv*, abs/2402.17564.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Annual Meeting of the Association for Computational Linguistics*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. 2022. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed Huai hsin Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *ArXiv*, abs/2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903.

Jerry W. Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, and Tengyu Ma. 2023. Larger language models do in-context learning differently. *ArXiv*, abs/2303.03846.

Hanwei Xu, Yujun Chen, Yulun Du, Nan Shao, Yanggang Wang, Haiyu Li, and Zhilin Yang. 2022. Gps: Genetic prompt search for efficient few-shot learning. In *Conference on Empirical Methods in Natural Language Processing*.

Heng Yang and Ke Li. 2023. Instoptima: Evolutionary multi-objective instruction optimization via large language model-based instruction operators. In *Conference on Empirical Methods in Natural Language Processing*.

Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, E. Cambria, Xiaodong Liu, Jianfeng Gao, and Furu Wei. 2022. Language models as inductive reasoners. *ArXiv*, abs/2212.10923.

Seonghyeon Ye, Doyoung Kim, Joel Jang, Joongbo Shin, and Minjoon Seo. 2022. Guess the instruction! flipped learning makes language models stronger zero-shot learners. *ArXiv*, abs/2210.02969.

Fei Yu, Hongbo Zhang, and Benyou Wang. 2023. Natural language reasoning, a survey. *ACM Computing Surveys*.

J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why johnny cant prompt: How non-ai experts try (and fail) to design llm prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI '23, New York, NY, USA. Association for Computing Machinery.

Zhihan Zhang, Shuohang Wang, Wenhao Yu, Yichong Xu, Dan Iter, Qingkai Zeng, Yang Liu, Chenguang Zhu, and Meng Jiang. 2023. Auto-instruct: Automatic instruction generation and ranking for black-box language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9850–9867, Singapore. Association for Computational Linguistics.

Zhuosheng Zhang, Aston Zhang, Mu Li, and Alexander J. Smola. 2022. Automatic chain of thought prompting in large language models. *ArXiv*, abs/2210.03493.

Pei Zhou, Jay Pujara, Xiang Ren, Xinyun Chen, Heng-Tze Cheng, Quoc Le, Ed Chi, Denny Zhou, Swaroop Mishra, and Huaixiu Steven Zheng. 2024. Self-discover: Large language models self-compose reasoning structures. *ArXiv*, abs/2402.03620.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *ArXiv*, abs/2211.01910.

Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. 2023. Large language models can learn rules. *ArXiv*, abs/2310.07064.

## A Construction of BBH-Induct Dataset

Figure 5 shows an example in the *movie_recommendation* task, where each question in BBH starts with *Find a movie similar to*, explicitly guiding the model. In our BBH-Induct dataset, we remove this instruction to better reflect real-world scenarios.

[Task Instruction]
Recommend movies similar to the given list of movies.

[Question]
~~Find a movie similar to~~ Minority Report, Total Recall, Inside Out, Forrest Gump:
Options:
(A) Phenomena
(B) Lilting
(C) Catwoman
(D) Edge of Tomorrow

[Answer]
(D)

Figure 5: An example of removing explicit information from BBH to create BBH-Induct.

## B Detailed Experimental Configurations

**Model Version** The models used in this paper include several versions from Llama, Mistral, Mixtral, and Gemini, each associated with specific API services. Llama 3 8B (`llama3-8b-8192`) and Llama 3 70B (`llama3-70b-8192`) were both accessed through Groq. Mistral 7B (`Mistral-7B-Instruct-v0.3`) and Mixtral 8x22B (`Mixtral-8x22B-Instruct-v0.1`) were provided by Together AI, while Mixtral 8x7B (`mixtral-8x7b-32768`) was also accessed via Groq. In addition, the Gemini series, which includes Gemini 1.0 Pro (`gemini-1.0-pro-001`), Gemini 1.5 Flash (`gemini-1.5-flash-001`), and Gemini 1.5 Pro (`gemini-1.5-pro-001`), were utilized through Googles API service.

**Safety Settings** We disable safety settings for Gemini to prevent Googles API from refusing to respond. The following harm categories were set to `BLOCK_NONE`: `HARASSMENT`, `HATE_SPEECH`, `SEXUALLY_EXPLICIT`, and `DANGEROUS_CONTENT`.

**Cost Details** Table 7 shows the estimated total cost for all experiments, which amounted to approximately $1,265, with API services provided by Groq, Together AI, and Google.

## C Additional Dataset Information

We provide the task details for the two datasets used in our experiments: Evals-Induct and BBH-Induct.

| Model | Total Cost | API Services |
|---|---|---|
| Llama 3 8B | $50 | Groq |
| Llama 3 70B | $475 | Groq |
| Mistral 7B | $120 | Together AI |
| Mixtral 8x7B | $150 | Groq |
| Mixtral 8x22B | $85 | Together AI |
| Gemini 1.0 Pro | $135 | Google |
| Gemini 1.5 Flash | $100 | Google |
| Gemini 1.5 Pro | $150 | Google |
| **Total Cost** | $1,265 | - |

Table 7: Estimated costs for each model across all experiments in this work.

The subtasks for BBH-Induct are listed in Table 9, while Evals-Inducts subtasks and descriptions are shown in Tables 10 and 11.

## D Detailed Experiment Results

### D.1 Further Analysis of INDUCT and Short Phrase

This section provides a supplemental comparison between $P_{\text{INDUCT}}$ and the Short Phrase (SP) with 0-shot, 0-shot-CoT, and 3-shot settings. As shown in Table 12, the effectiveness of our framework increases significantly with larger and more powerful models, demonstrating substantial performance improvements over smaller models.

### D.2 Detailed Cross-Model Performance Results

Figure 6 and Figure 7 present the exact match accuracy (%) of $P_{\text{INDUCT}}$ and $P_{\text{INDUCT-LEARN}}$ under different cross-model settings on the BBH-Induct and Eval-Induct datasets, where the prompt induction model and task execution/inference model are different. Figure 8 compares the performance difference in percentage between INDUCT and the baseline of using zero-shot without instructions under the cross-model settings.

### D.3 Breakdown of BBH-Induct tasks and Win Rate Analysis

The Table 13 shows the accuracy $\Delta$ for the INDUCT-LEARN (Ours) and Short Phrase + 3-shot across different tasks. The win rate indicates the proportion of instances where the INDUCT-LEARN outperformed the Short Phrase + 3-shot setting. On average, our model achieved a win rate of 6/8. The performance of all models improved overall on BBH-Induct dataset.

### D.4 Breakdown of Eval-Induct Tasks

Figure 9 presents the head-to-head comparison of 8 models on the 24 tasks from Eval-Induct. The table illustrates the accuracy difference distribution across these tasks for the models using our Induct-Learn Prompt compared to the Short Phrase under 3-shot settings.

### D.5 Model Fill Rate

To estimate the frequency of the insufficient $k$ situation in the LEARN stage, we define the fill rate (FR) as the proportion of cases where the model successfully generates $k$ valid CoT demonstrations. The fill rate is calculated as follows:

$$FR = \frac{\sum_{j=1}^{M} [|D_{\text{LEARN},j}| \geq k]}{M}$$

where $M$ is the total number of tasks, $D_{\text{LEARN}}$ is the set of demonstrations generated during the LEARN stage, and $k$ represents the number of required demonstrations. The formula sums 1 for each task where $|D_{\text{LEARN},j}| \geq k$, and 0 for subtasks where this condition is not met.

Table 8 presents the fill rates for various models on the BBH-Induct and Evals-Induct datasets. A higher fill rate indicates that the model can produce sufficient examples for effective few-shot learning in the LEARN stage. As observed, larger models, such as Llama 3 70B, Mixtral 8x22B, and Gemini 1.5 Pro, demonstrate significantly higher fill rates compared to smaller models. This indicates that larger models are better at generating the necessary number of high-quality CoT demonstrations, leading to more robust few-shot learning.

| Model | BBH-Induct | Evals-Induct |
|---|---|---|
| Llama 3 8B | 77.78 | 37.93 |
| Llama 3 70B | 88.89 | 79.31 |
| Mistral 7B | 66.67 | 48.28 |
| Mixtral 8x7B | 70.37 | 58.62 |
| Mixtral 8x22B | 92.59 | 72.41 |
| Gemini 1.0 Pro | 70.37 | 44.83 |
| Gemini 1.5 Flash | 96.30 | 82.76 |
| Gemini 1.5 Pro | 96.30 | 86.21 |

Table 8: Fill Rate (%) of models in generating sufficient CoT demonstrations. We set the experimental parameters as $N = 9$ and $k = 3$. Therefore, the fill rate represents the percentage of instances where at least 3 CoT demonstrations are generated.

### D.6 Details of Meta Prompt Variation

We instruct GPT-4o to generate variations of the original meta prompt, maintaining the template structure while rewriting the content. Each variation directs LLMs to act as different domain experts, and the phrasing and terminology were varied across versions. The original meta prompt and its variations are shown in Figures 10 to 13.

### D.7 Induct-Learn Framework Examples

Figure 14 through Figure 19 demonstrate how the Induct-Learn framework functions on the two datasets, providing illustrative examples of its operation. The figures showcase the INDUCT, LEARN, and inference stages of the framework.

## E Use of AI Assistants

In this paper, GPT-4o was employed for text refinement and assisting in code development. All outputs from GPT-4o were for reference, and the final work was completed by the authors.

| **Task** | **Subtasks** | Instance Count |
|---|---|---|
| Boolean Expressions | boolean expressions | 200 |
| Causal Judgment | causal judgement | 137 |
| Date Understanding | date understanding | 200 |
| Disambiguation QA | disambiguation qa | 200 |
| Dyck Languages | dyck languages | 200 |
| Formal Fallacies Syllogisms Negation | formal fallacies | 200 |
| Geometric Shapes | geometric shapes | 200 |
| Hyperbaton (Adjective Ordering) | hyperbaton | 200 |
| Logical Deduction | logical deduction five objects | 200 |
| | logical deduction seven objects | 200 |
| | logical deduction three objects | 200 |
| Movie Recommendation | movie recommendation | 200 |
| Multi-Step Arithmetic | multistep arithmetic two | 200 |
| Navigate | navigate | 200 |
| Object Counting | object counting | 200 |
| Penguins in a Table | penguins in a table | 96 |
| Reasoning about Colored Objects | reasoning about colored objects | 200 |
| Ruin Names | ruin names | 200 |
| Salient Translation Error Detection | salient translation error detection | 200 |
| Snarks | snarks | 128 |
| Sports Understanding | sports understanding | 200 |
| Temporal Sequences | temporal sequences | 200 |
| Tracking Shuffled Objects | tracking shuffled objects five objects | 200 |
| | tracking shuffled objects seven objects | 200 |
| | tracking shuffled objects three objects | 200 |
| Web of Lies | web of lies | 200 |
| Word Sorting | word sorting | 200 |

Table 9: The 27 subtask names in BBH-Induct and their corresponding 23 task. In the paper, we calculate the average accuracy score based on these 23 task.

| Task Name | Description |
|---|---|
| 2d_movement | Evaluate if GPT can keep track of its position and orientation while random walking on a 2D grid in first-person POV. |
| anagrams | Given an anagram (word whose letters has been randomly shuffled), find the original word. The dataset only takes 12+ character long words. The prompt could include a list of 100+ candidates (the anagram being one of those) which would make the guess easier for a human being but oddly enough, the accuracy is lower with this input. |
| bitwise | Bitwise operations are common across most programming languages. GPT-4 should be able to successfully evaluate them. |
| count_token_freq_dna | The count_token_freq_dna evaluation tasks the model with counting the occurrence of a specific nucleotide (A, T, G, or C) within provided DNA sequences. These sequences are derived from real human DNA and have lengths ranging from 5 to 20 base pairs. |
| css-selectors | Test the models ability to translate css selectors to verbal description and the other way around |
| determinant | Asks GPT only the answer of the determinant from 1x1 to 4x4 matrices. |
| forth_stack_sim | Tests the models ability to keep track of a stack of numbers given a set of ANS Forth words. The model is asked to respond to a series of numbers and words with the resulting stack representation. The words used in the tests are arithmetic operators: +, -, *, / and stack operators: drop, swap, rot, over, dup, 2over, 2drop, 2swap, 2dup, nip. The prompts and expected results on the stack are all less than 15 numbers and words long. |
| guess_the_singer | This evaluation measures the model's ability to identify a singer or band by analyzing the first 10 words of a song. To ensure the evaluation's fairness and focus, we have excluded songs with multiple singers/bands and included only those published before 2021. To test the model's performance, we provide it with three potential choices and evaluate its accuracy in selecting the correct one. |
| largest_country | Test the model's ability to determine the largest country from the list. |
| lat_long_identify | This eval tests the model's ability to correctly match latitude and longitude coordinates to the country they are located in. The twist is that the model's response must be in the official language of the country. |
| mate-in-one | Finds the only mate in one move on various chess board positions. |
| math_equations | Test model's ability to explain and solve math equations described in words. |
| missing_operators | Solve missing operators (two blanks) |
| ner_finance | Named entity recognition (NER) over financial documents. |
| next-val-series | Test the model's ability to predict the next value in a mathematical series. |
| partially_solved_crossword_clues | This evaluates the model's ability to find the answer to crossword clues, when some of the letters of the answer are already provided. |
| points_on_line | 100 sets of vector coordinates in the form of (x, y, z), (x, y, z), with an ideal centre coordinate. The coordinates have a random start position of (-10, -10, -10) to (10, 10, 10) and a furthest maximum distance from origin per-component of 20. All positions are in steps of 0.01 for ease of readability and human understanding. |
| poker_analysis | Adds a calculation eval in 2-player no-fold heads up scenario pre-flop without viewing the community cards on the raw probability of winning against a random opponent. |
| recurrence-relation | This will evaluate the models' performances when calculating the runtime of recurrence relations |
| resistor_ohm_calculator | Tests the model's ability to calculate the resistance (in ohms) based on the colors of resistor's four bands. This eval doesn't test the model's ability to calculate the minimum/maximum value based on the tolerance. This eval doesn't test the model's ability to calculate the resistance of a resistor with five or six bands. |
| smiles_to_formula | Conversion of SMILES (Simplified Molecular Input Line Entry System) strings – a widely used ASCII string notation for molecular structures – to the corresponding molecular formula (the types and numbers of atoms in the molecule). |
| sort_numeric | Tests performance sorting different comma-separated values under different circumstances (integers/decimals, positives/negatives, as well as currency-formatted values). |
| three-pt-mapping | Given a result from a three-point cross (using 3 genes to determine the order and distance between the genes.), return the allele that is in the middle of the 3 genes. It is a multi-step genetic problem. |
| word_association | The Word Association Evaluation Set is designed to gauge the model's proficiency in discerning a covertly embedded word within a sequence of interconnected words. |

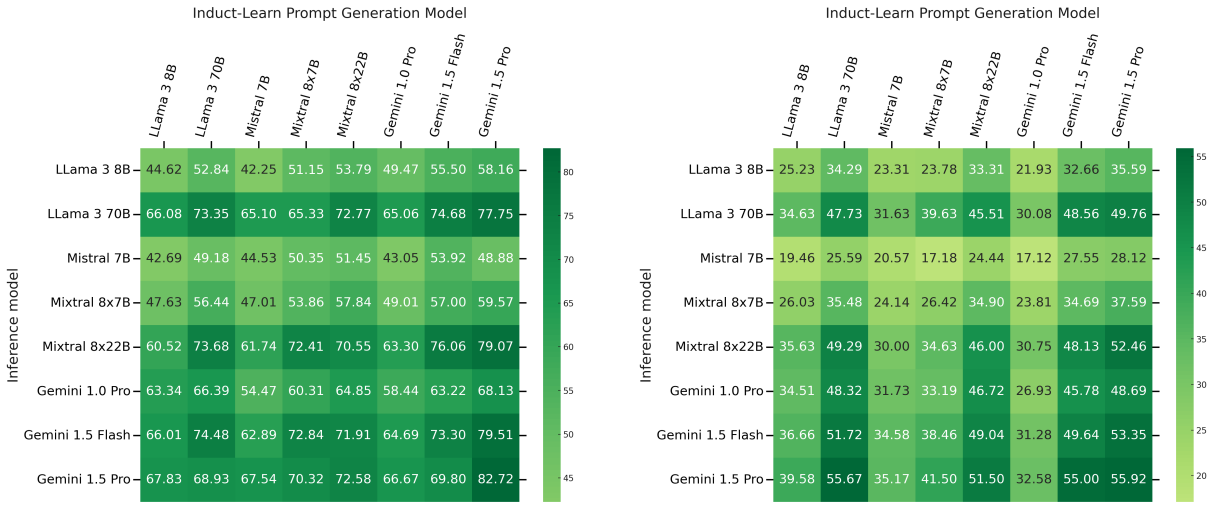
Table 10: Tasks Descriptions of Evals-Induct

| Task | Sub Task | Instance Count |
|---|---|---|
| 2d_movement | - | 91 |
| anagrams | - | 200 |
| bitwise | - | 200 |
| count_token_freq_dna | - | 92 |
| css-selectors | css-selectors_explain | 105 |
| | css-selectors_verbal | 48 |
| determinant | - | 200 |
| forth_stack_sim | - | 167 |
| guess_the_singer | - | 200 |
| largest_country | - | 115 |
| lat_long_identify | - | 141 |
| mate-in-one | - | 91 |
| math_equations | - | 91 |
| missing_operators | - | 91 |
| ner_finance | - | 200 |
| next-val-series | - | 91 |
| partially_solved_crossword_clues | - | 91 |
| points_on_line | - | 91 |
| poker_analysis | - | 200 |
| recurrence-relation | - | 200 |
| resistor_ohm_calculator | - | 200 |
| smiles_to_formula | - | 200 |
| sort_numeric | - | 91 |
| three-pt-mapping | - | 109 |
| word_association | word_association_related_words_2 | 111 |
| | word_association_related_words_3 | 99 |
| | word_association_related_words_4 | 89 |
| | word_association_related_words_5 | 79 |

Table 11: The 28 subtask names in Eval-Induct and their corresponding 24 task. In the paper, we calculate the average accuracy score based on these 24 task.

| Model | BBH-Induct | | | | Eval-Induct | | | |
|---|---|---|---|---|---|---|---|---|
| | INDUCT (Ours) | SP 0-shot | SP 0-shot-CoT | SP 3-shot | INDUCT (Ours) | SP 0-shot | SP 0-shot-CoT | SP 3-shot |
| Llama 3 8B | **43.49** | 30.28 | 42.15 | 35.48 | **18.21** | 9.19 | 11.03 | 16.50 |
| Llama 3 70B | **69.93** | 59.45 | 63.40 | 57.50 | **47.91** | 18.83 | 17.28 | 30.50 |
| Mistral 7B | 37.59 | 37.58 | 39.25 | **41.83** | 16.32 | 12.78 | 10.57 | **17.35** |
| Mixtral 8x7B | 44.71 | 38.97 | **44.92** | 35.17 | **25.22** | 12.64 | 12.60 | 15.23 |
| Mixtral 8x22B | **67.01** | 45.57 | 56.12 | 37.04 | **40.50** | 16.96 | 18.17 | 20.63 |
| Gemini 1.0 Pro | **53.55** | 46.21 | 49.88 | 53.30 | 24.23 | 16.13 | 15.77 | **33.25** |
| Gemini 1.5 Flash | **65.09** | 42.62 | 56.57 | 60.39 | **44.44** | 18.17 | 18.15 | 35.59 |
| Gemini 1.5 Pro | **78.96** | 52.44 | 63.48 | 64.46 | **51.13** | 22.42 | 24.13 | 38.58 |

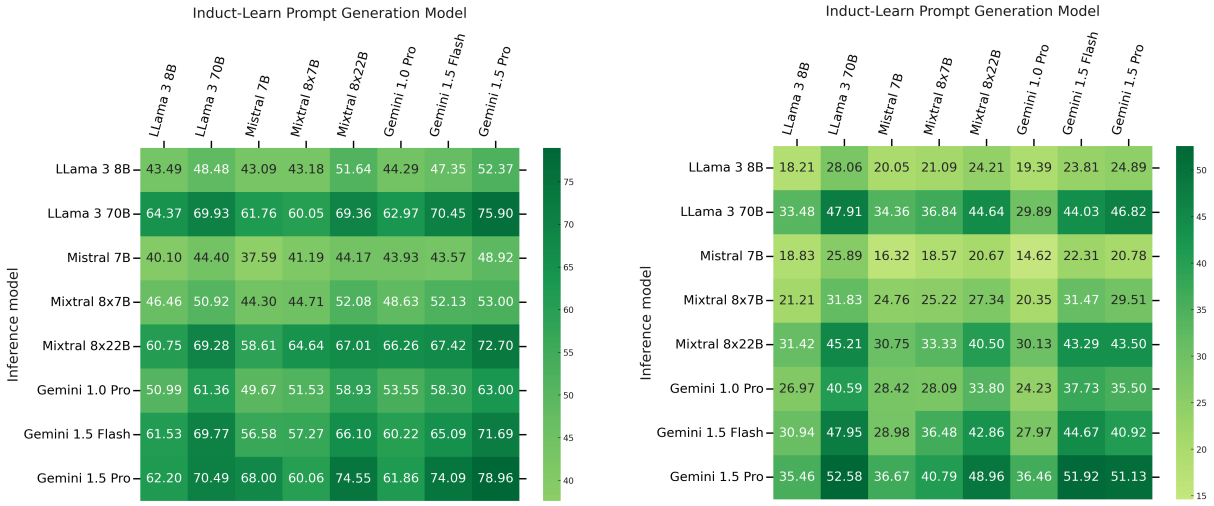
Table 12: Results of instruction generation experiments across models and tasks in zero-shot and few-shot settings. We evaluate exact match accuracy (%) and calculate the mean of all the tasks of both BBH-Induct and Evals-Induct datasets. SP denotes Short Phrase.




(a) Cross-model Performance (BBH-Induct)

(b) Cross-model Performance (Evals-Induct)

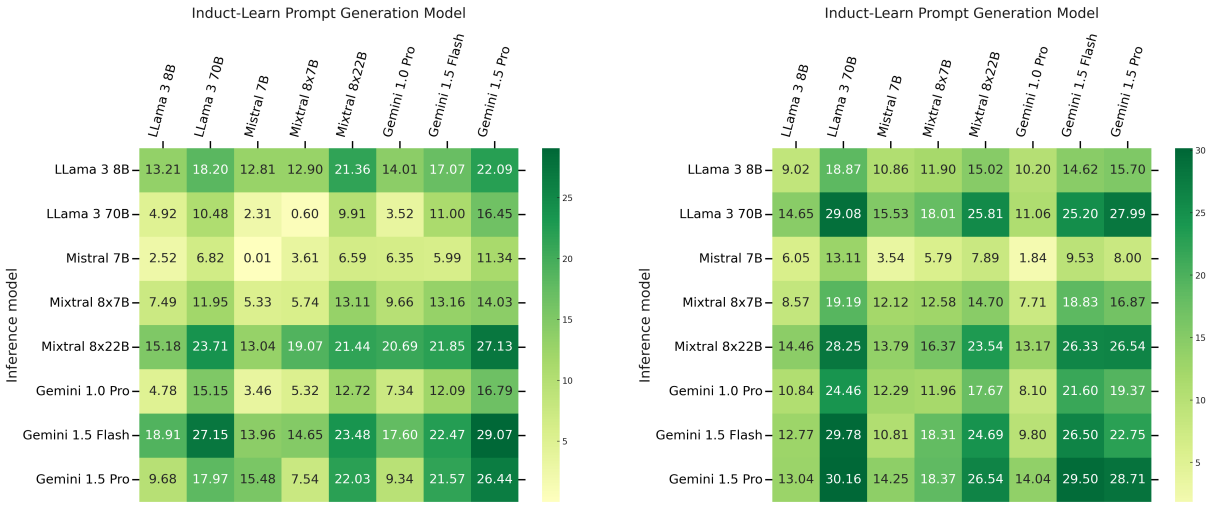
Figure 6: Exact match accuracy (%) of INDUCT-LEARN (Ours) under different cross-model settings, where the prompt induction model and task execution/inference model are different. Darker green indicates better performance of INDUCT-LEARN (Ours), while darker red indicates worse performance.The comparison is conducted on the (a) BBH-Induct and (b) Evals-Induct datasets.




(a) Cross-model Performance (BBH-Induct)

(b) Cross-model Performance (Evals-Induct)

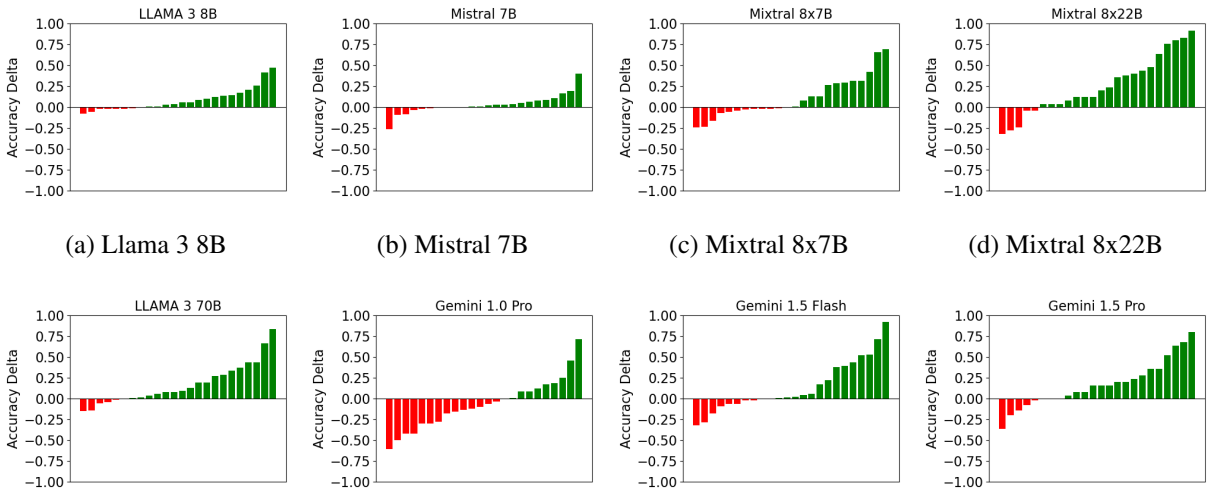
Figure 7: Exact match accuracy (%) of INDUCT (Ours) under different cross-model settings, where the prompt induction model and task execution/inference model are different. Darker green indicates better performance of INDUCT (Ours), while darker red indicates worse performance. The comparison is conducted on the (a) BBH-Induct and (b) Evals-Induct datasets.




(a) Cross-model Improvement (BBH-Induct)

(b) Cross-model Improvement (Evals-Induct)

Figure 8: Comparison of INDUCT (Ours) and Short Phrase + 0-shot under different cross-model settings, where the prompt induction model and task execution/inference model are different. Accuracy is expressed as a percentage. Darker green indicates greater performance improvement with INDUCT (Ours), while darker red indicates greater performance decline. The comparison is conducted on the (a) BBH-Induct and (b) Evals-Induct datasets.




(a) Llama 3 8B

(b) Mistral 7B

(c) Mixtral 8x7B

(d) Mixtral 8x22B

(e) Llama 3 70B

(f) Gemini 1.0 Pro

(g) Gemini 1.5 Flash

(h) Gemini 1.5 Pro

Figure 9: The accuracy difference distribution across 24 Eval-Induct tasks for 8 models using our Induct-Learn Prompt compared to the Short Phrase under 3-shot settings. Tasks are sorted by accuracy difference from low to high, indicating that tasks to the right benefit more from our method. Improvements are marked in green, while performance declines are highlighted in red.

| Task | Llama 3 | | Mis(x)tral | | | Gemini | | | Win Rate |
|---|---|---|---|---|---|---|---|---|---|
| | 8B | 70B | 7B | 8x7B | 8x22B | 1.0 Pro | 1.5 Flash | 1.5 Pro | |
| boolean expressions | -7.00 | 10.00 | 27.00 | 26.50 | 44.00 | 15.50 | 9.50 | 12.00 | **7/8** |
| causal judgement | 2.92 | 2.92 | -3.65 | -12.41 | 4.00 | -2.92 | 0.73 | 8.00 | **5/8** |
| date understanding | 38.50 | 26.00 | 9.00 | 13.50 | 52.00 | 13.50 | 17.00 | 24.00 | **8/8** |
| disambiguation qa | 33.50 | 25.50 | -9.00 | 26.00 | 4.00 | -8.50 | 10.50 | -20.00 | **5/8** |
| dyck languages | 16.50 | 8.00 | 10.50 | 23.50 | 4.00 | 2.50 | 9.50 | 24.00 | **8/8** |
| formal fallacies | -23.50 | 14.50 | -9.50 | -3.00 | -12.00 | -48.00 | -8.50 | 0.00 | 1/8 |
| geometric shapes | 25.50 | 28.00 | 14.00 | 9.00 | 36.00 | 11.00 | 17.00 | 0.00 | **7/8** |
| hyperbaton | 2.50 | 10.50 | -9.50 | 17.50 | 20.00 | 3.00 | -29.50 | 12.00 | **6/8** |
| logical deduction | 15.00 | 17.17 | 11.50 | 26.50 | 49.33 | 5.33 | 10.17 | 2.67 | **8/8** |
| movie recommendation | -11.00 | -5.00 | -60.50 | -8.00 | -8.00 | -23.00 | -14.00 | -16.00 | 0/8 |
| multistep arithmetic two | 14.50 | 17.00 | 9.00 | 24.50 | 80.00 | 65.00 | 89.00 | 84.00 | **8/8** |
| navigate | 14.50 | 26.00 | 3.00 | 3.00 | 44.00 | -4.50 | -7.50 | 64.00 | **6/8** |
| object counting | 37.00 | 34.00 | -6.50 | 53.00 | 80.00 | -4.50 | 19.50 | 44.00 | **6/8** |
| penguins in a table | 32.29 | 11.46 | 9.38 | 22.92 | 4.00 | 13.54 | 25.00 | 16.00 | **8/8** |
| reasoning about colored objects | 37.00 | 16.00 | 26.00 | 19.00 | 24.00 | 13.00 | 1.00 | 4.00 | **8/8** |
| ruin names | -13.50 | -9.50 | -14.00 | 14.50 | 28.00 | 5.50 | 28.50 | 20.00 | **5/8** |
| salient translation error detection | 23.00 | 5.00 | 18.00 | 16.00 | 52.00 | -5.50 | 0.50 | -8.00 | **6/8** |
| snarks | 3.91 | 28.91 | 14.84 | 32.03 | 48.00 | 46.09 | 57.81 | 8.00 | **8/8** |
| sports understanding | -12.00 | -16.00 | -27.50 | 47.00 | 16.00 | -17.50 | -34.00 | 16.00 | 3/8 |
| temporal sequences | -28.00 | -1.00 | -19.50 | -3.00 | 48.00 | 7.50 | -24.00 | 0.00 | 2/8 |
| tracking shuffled objects | 7.17 | 64.67 | 18.00 | 18.83 | 37.33 | 4.67 | 57.33 | 65.33 | **8/8** |
| web of lies | -22.50 | 38.00 | 49.50 | 51.00 | 84.00 | 23.00 | 46.00 | 52.00 | **7/8** |
| word sorting | 24.00 | 12.50 | 2.00 | 12.00 | 32.00 | 3.50 | 15.50 | 8.00 | **8/8** |
| AVERAGE | 9.14 | 15.85 | 2.70 | 18.69 | 33.51 | 5.14 | 12.91 | 18.26 | **6/8** |

Table 13: Accuracy $\Delta$ comparison for INDUCT-LEARN (Ours) and Short Phrase + 3-shot across various tasks in the BBH-Induct dataset. The Win Rate column represents the proportion of times INDUCT-LEARN (Ours) outperforms the Short Phrase. **Bold** indicates a win rate greater than 0.5.

You are an expert in the field of NLP (Natural Language Processing), possessing exceptional data observation and analysis skills. Your expertise includes extracting significant rules from complex datasets and formulating precise task instructions based on these rules.

Currently, you are focusing on analyzing a specific set of examples, deriving insights to formulate a clear and detailed task description. This task description will serve as an instruction set, guiding the execution of the related tasks.

The task description should include the following elements:

Task Content: Clearly define the purpose of the task and the specific activities required to be completed.

Input Format: Provide detailed descriptions of the types of data accepted, their formats, and how to process these data effectively.

Output Format: Clearly outline the expected result types and formats, including any necessary standards or specifications.

Operational Steps: Detail the specific step-by-step procedures required to complete the task.

Please write the [Task Instruction] concisely and clearly to ensure it is easily understandable and followable by users.

Figure 10: Meta Prompt

You are an authority in the realm of Quantum Physics, known for your exemplary skills in data observation and theoretical analysis. Your expertise lies in extracting significant principles from intricate datasets and creating precise task instructions based on these principles.

Presently, you are concentrating on examining a specific set of examples to draw insights for formulating a lucid and comprehensive task description. This description will serve as a set of guidelines directing the accomplishment of related tasks.

The task description should encompass the following components:

Task Content: Explicitly state the objective of the task and detail the specific activities that need to be performed.

Input Format: Describe thoroughly the types of data that are acceptable, their formats, and how to process these data properly.

Output Format: Clearly specify the expected result types and formats, including any required standards or criteria.

Operational Steps: Outline the exact step-by-step actions necessary to complete the task.

Please craft the [Task Instruction] succinctly and clearly to ensure it is straightforward and easily followable by users.

Figure 11: Meta Prompt 1 (Rewritten)

You are an expert in the field of Environmental Science, known for your outstanding observational and analytical abilities. Your skillset includes identifying key patterns in complex environmental data and crafting accurate task instructions based on these observations.

Currently, you are concentrating on examining a particular set of examples to derive insights and create a comprehensive task description. This task description will guide the completion of related assignments.

The task description should include the following components:

Task Content: Clearly delineate the objective of the task and the specific actions required to accomplish it.

Input Format: Offer detailed explanations of the types of data accepted, their formats, and the best methods for processing these data.

Output Format: Define clearly the expected types and formats of the results, including any relevant standards or specifications.

Operational Steps: Describe the precise step-by-step procedures necessary to complete the task.

Please write the [Task Instruction] concisely and clearly to ensure it is straightforward and easy to follow for users.

Figure 12: Meta Prompt 2 (Rewritten)

You are an expert in the field of materials science, possessing exceptional skills in data observation and analysis. Your expertise includes extracting significant rules from intricate datasets and formulating precise task instructions based on these rules.

Presently, you are concentrating on analyzing a specific set of examples, deriving insights to create a clear and comprehensive task description. This task description will act as a guide, directing the execution of related tasks.

The task description should include the following components:

Task Content: Explicitly define the purpose of the task and the specific activities that need to be performed.

Input Format: Provide detailed descriptions of the types of data accepted, their formats, and how to process these data effectively.

Output Format: Clearly outline the expected result types and formats, including any necessary standards or specifications.

Operational Steps: Detail the specific step-by-step procedures that are necessary to complete the task.

Please write the [Task Instruction] concisely and clearly to ensure it is easily understandable and followable by users.

Figure 13: Meta Prompt 3 (Rewritten)

| Method | shot | Request Count | | | |
|---|---|---|---|---|---|
| | | Instruction generation | 3-shot COT generation | Inference | Total |
| Short Phrase | 0 | 0 | 0 | 1 | 1 |
| | 3 | 0 | 0 | 1 | 1 |
| Self-Discover | 0 | 3 | 0 | 1 | 4 |
| | 3 | 3 | 9 | 1 | 13 |
| Ours | 0 | 1 | 0 | 1 | 2 |
| | 3 | 1 | 9 | 1 | 11 |

Table 14: Breakdown of request usage from Figure 4. Note that for *Short Phrase*, we use the human-provided demonstrations in the dataset. For *Self-Discover* and *Ours*, we use the demonstrations generated after the LEARN stage.


Figure 14: INDUCT Stage: A Case Example in BBH-Induct

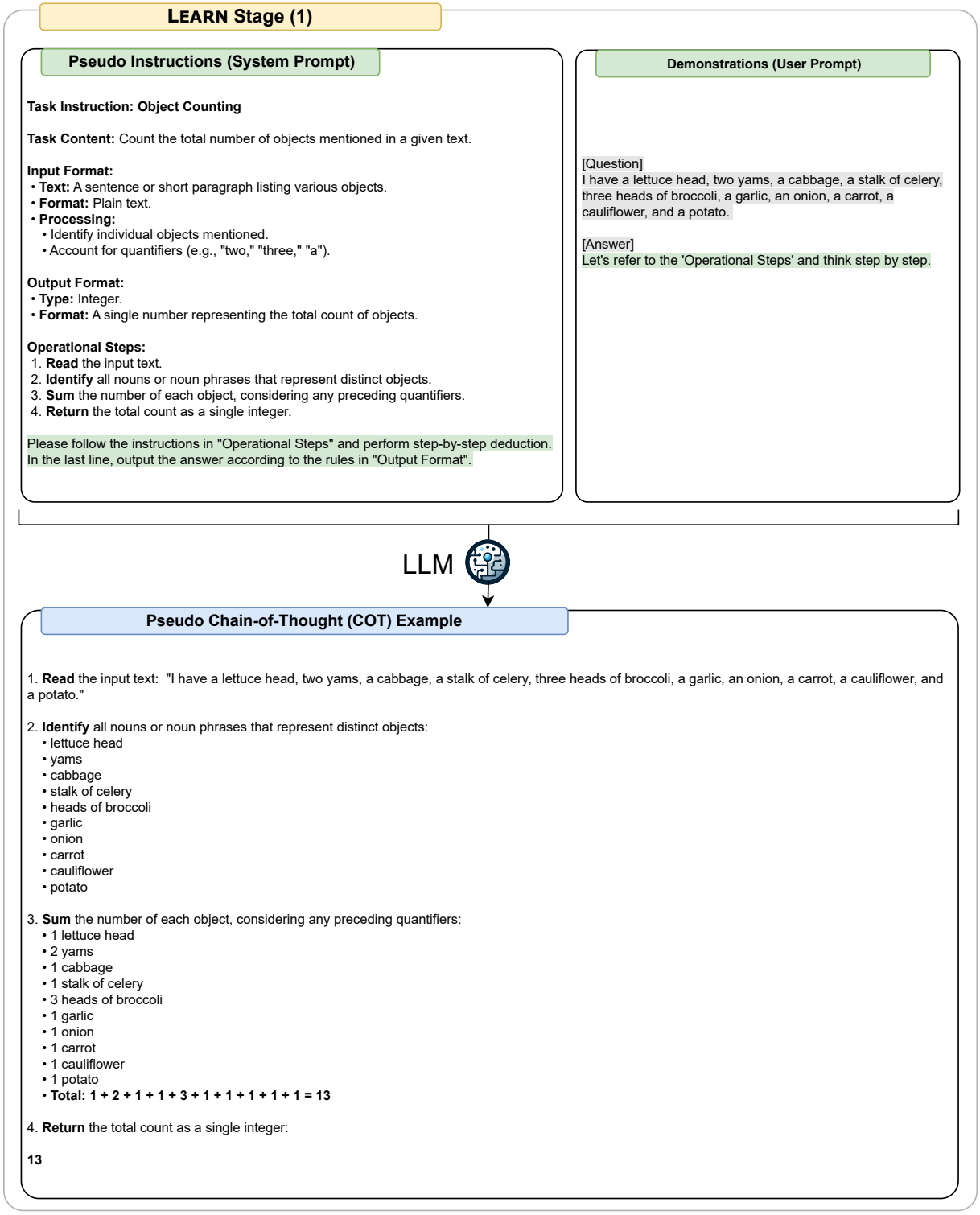



Figure 15: LEARN Stage: A Case Example in BBH-Induct

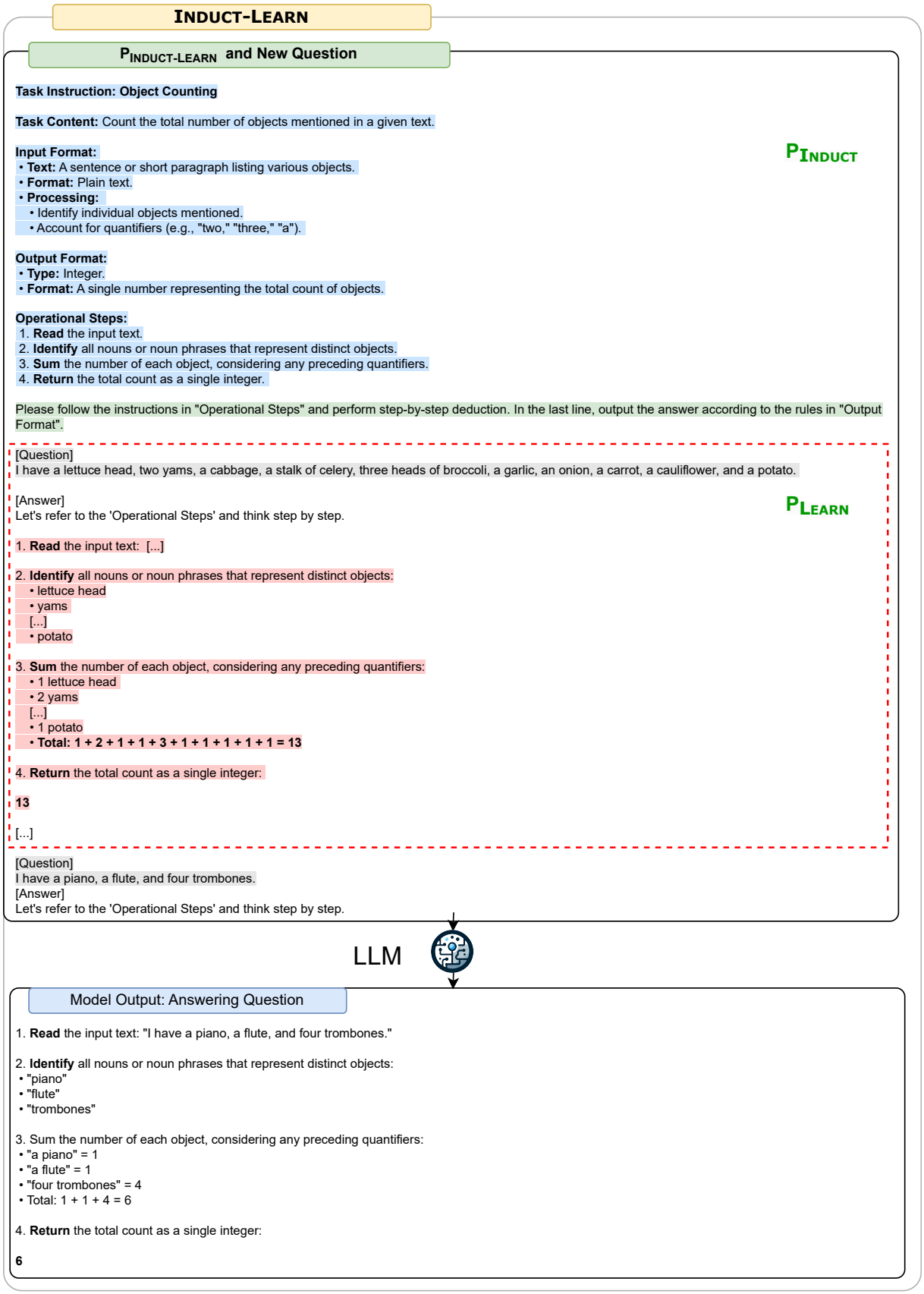



Figure 16: INDUCT-LEARN framework: A Case Example in BBH-Induct

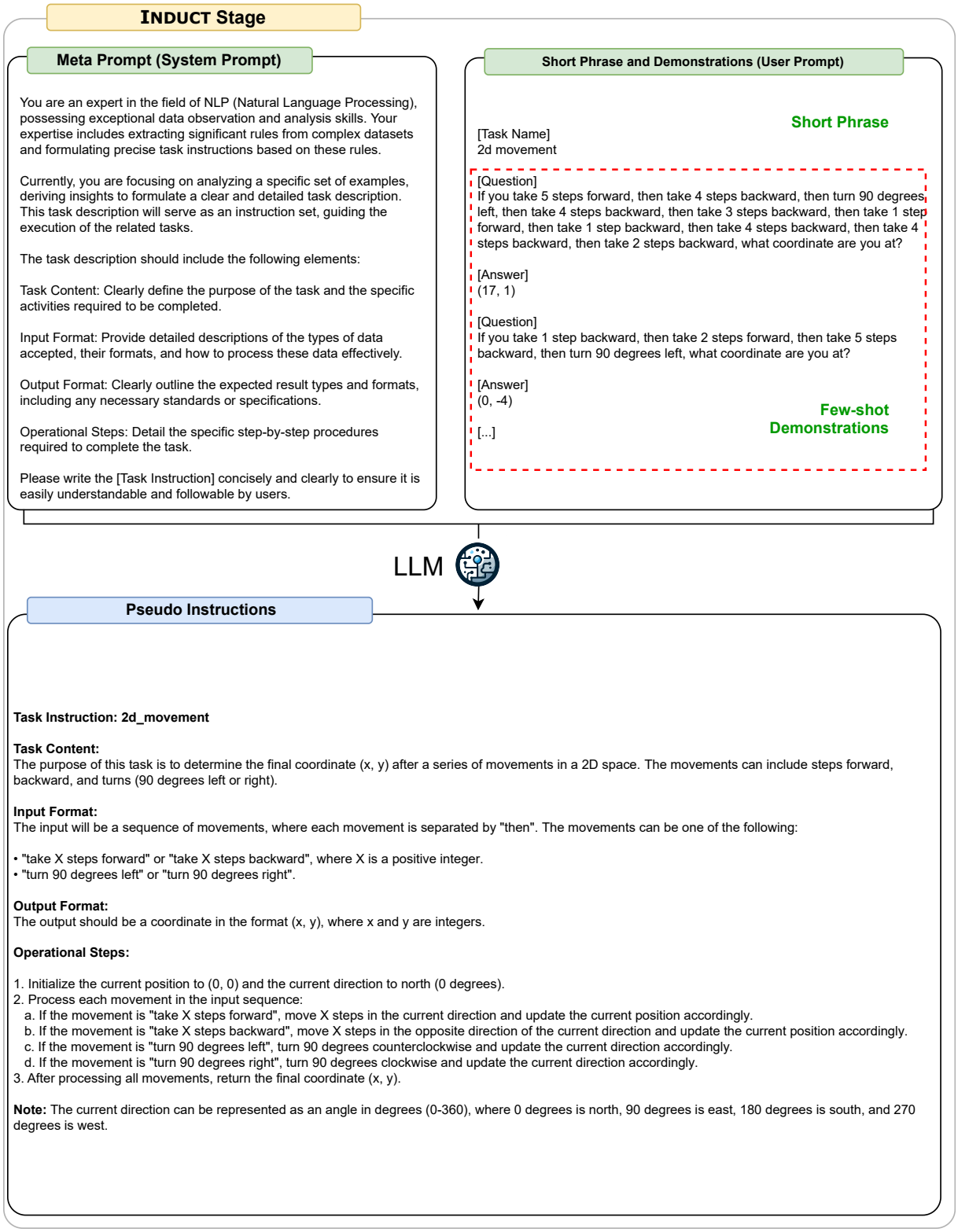



Figure 17: INDUCT Stage: A Case Example in Eval-Induct

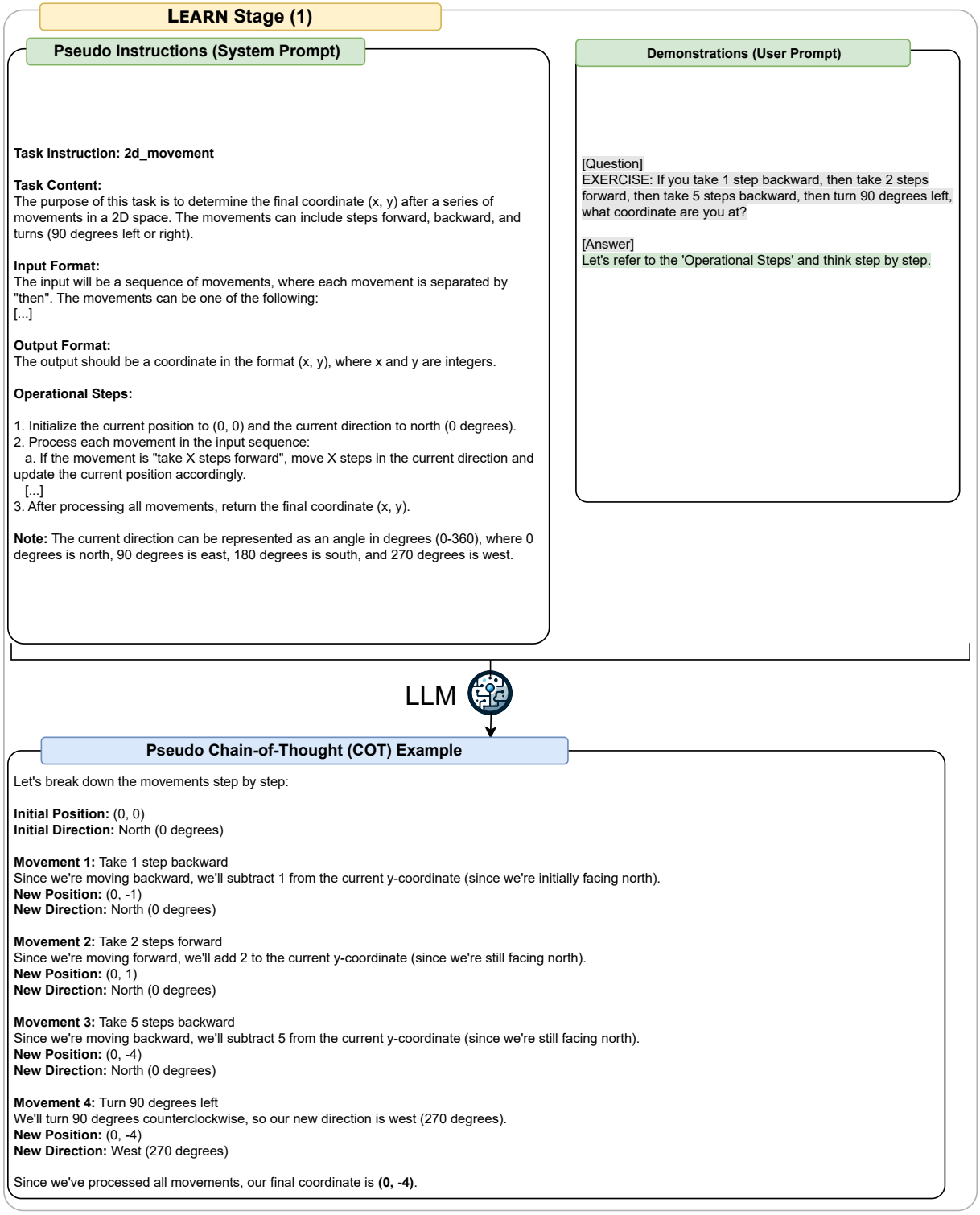



Figure 18: LEARN Stage: A Case Example in Eval-Induct

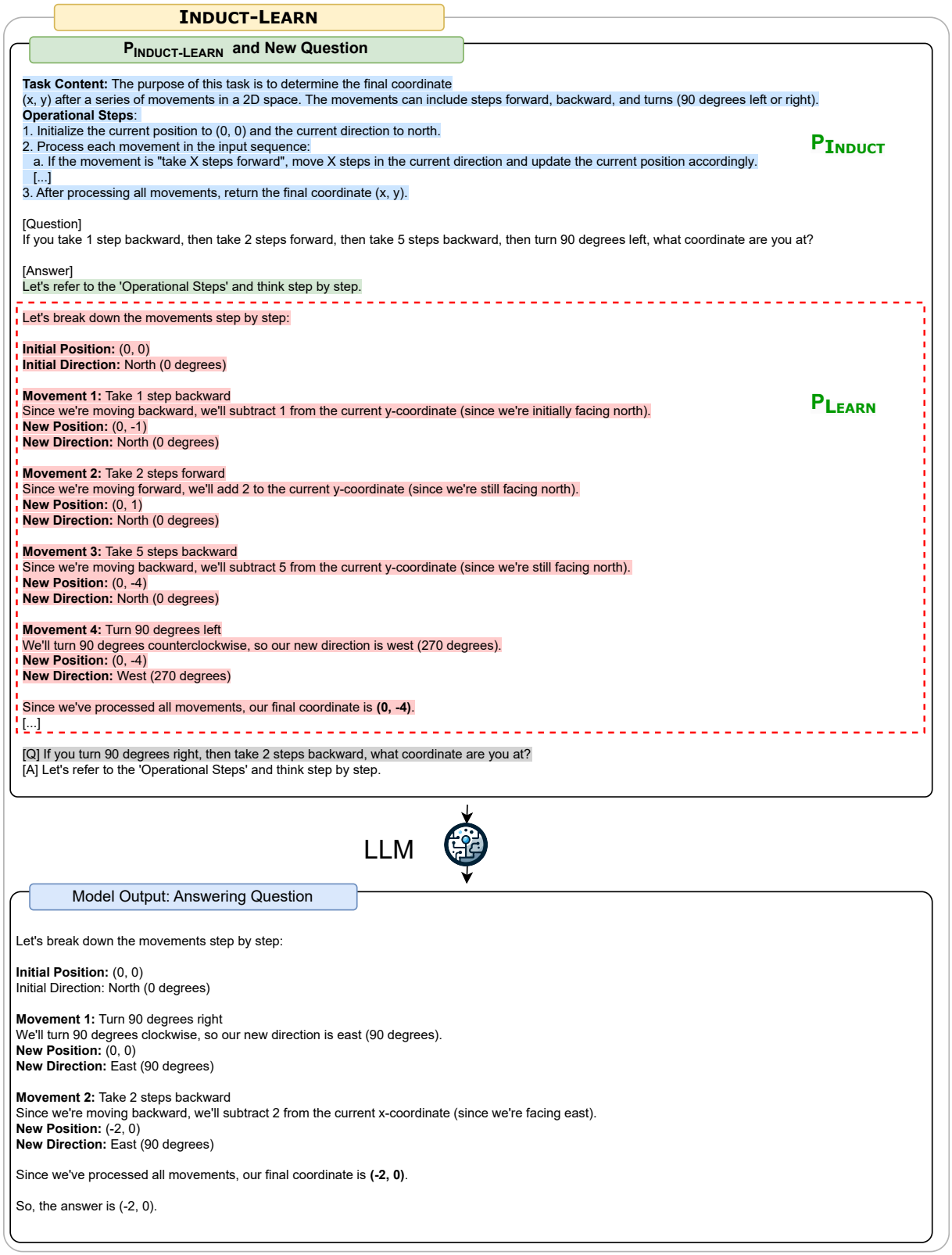



Figure 19: INDUCT-LEARN framework: A Case Example in Eval-Induct