Evolution without Large Models: Training Language Model with Task Principles

Anonymous ACL submission

Abstract

A common training approach for language models involves using a large-scale language model to expand a human-provided dataset, which is subsequently used for model training. This method significantly reduces training costs by eliminating the need for extensive human data annotation. However, it still faces challenges such as high carbon emissions during data augmentation and the risk of data leakage when we use closed-source LLMs.

To address these issues, we propose a self-evolution method for language models. First, we introduce the Multi-level Principle Generation, which enables a large-scale model to summarize task-completion principles based on a small amount of task data. Then, we propose the Principle-based Instance Generation, in which a smaller-scale language model uses these task principles to generate a large amount of data. This data is then used for model training. Experimental results show that our proposed method significantly improves model performance compared to directly using a smaller-scale language model to generate data. Additionally, since we only use the large-scale language model to generate the task-completion principles, the carbon emissions associated with training the model are greatly reduced. Our code is available at https://anonymous.4open.science/r/PSI-0ED6/.

1 Introduction

Instruction tuning (Ouyang et al., 2022; Chung et al., 2024) is a crucial step in training large language models (LLMs). Through supervised learning on a large dataset of instruction-response pairs, LLMs acquire the ability to follow instructions and utilize the knowledge accumulated during pre-training. This capability allows LLMs to be applied to a wider range of tasks. Additionally, this training approach enables LLMs to be applied on more task-specific applications and more tasks in vertical domains.

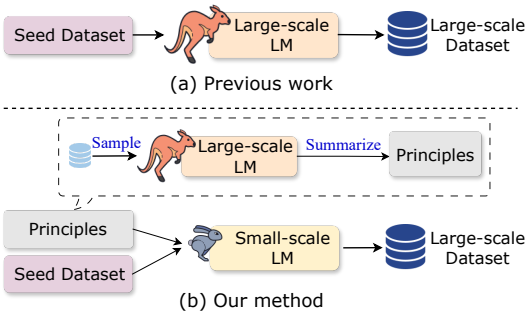

Figure 1: Comparison between existing self-instruct methods (a) and our proposed PSI (b). Instead of using large-scale LM to generate an instruction-tuning dataset, PSI employs small-scale LM to generate a high-quality dataset using the principles as guidance.

Existing instruction tuning methods typically obtain large-scale datasets from two sources. The first involves employing a large number of human annotators to annotate the training data (Zhou et al., 2023; Sanh et al., 2022). While this method can produce high-quality datasets, it is expensive and time-consuming. And the second method uses a larger-scale LM to generate the instances, known as the self-instruct framework (Wang et al., 2023a; Taori et al., 2023). This approach usually requires human annotation of a small seed dataset, after which a large-scale LM mimics the examples in the seed dataset and incorporates the domain knowledge to generate a larger dataset. This method has been widely applied to tasks such as question answering and reasoning in specific domains, *e.g.,* tool learning (Gao et al., 2024).

Since the self-instruct method requires a powerful large-scale LM to perform data annotation in order to obtain high-quality datasets, it involves providing these large-scale LMs with some task examples in the input and having them generate a large-scale dataset. Due to the large number of parameters in these LLMs, the

instances generation process inevitably produces a significant amount of carbon emissions. However, using smaller LLMs to generate instances fails to meet the high-quality dataset requirements for instruction fine-tuning (Wang et al., 2023b).

On the other hand, most self-instruct methods (Xu et al., 2024; Peng et al., 2023; Liu et al., 2023) rely on closed-source LLMs, such as ChatGPT and Claude. During the instances generation process, we need to provide these closed-source models with example instances as well as additional knowledge required for instances generation (Li et al., 2024b). For example, when annotating domain-specific tool-learning instances, we need to include internal tool API documentation in the prompts to generate data using these tools (Qin et al., 2024; Yin et al., 2023). However, this approach risks leaking internal tool documentation. In some applications, due to privacy and security concerns, we cannot have permission to transfer these tool documents to closed-source model service providers.

Therefore, in this paper, we propose the **P**rinciple-based **S**elf-**I**nstruct (PSI) method. This approach involves having the large LM generate a few task-completion principles, which are then used by a smaller LM to generate a high-quality, large-scale dataset by following these principles. This method avoids having the large model directly generate the entire dataset and eliminates the need to transfer domain-specific knowledge to closed-source model service providers. First, we introduce the Multi-level Principle Generation method, where the large-scale LM reflects and summarizes multi-level principles for completing tasks based on a seed dataset annotated by humans. The Multi-level Principle Generation method starts by randomly sampling several subsets from the seed dataset. To generate more targeted task principles, we first let a smaller LM directly generate instances based on the seed dataset. Then, we use a large-scale LM to generate task principles that better describe the points where the smaller LM is prone to errors. Subsequently, through further summarization and reflection, we obtain a refined, high-level task principle pool. Next, we use the Principle-based Instance Generation method to follow these task-completion principles and generate a large-scale instruction fine-tuning dataset by a smaller LM. This instances generation method also mitigates the high carbon emissions when directly using large-scale LMs to generate instances. Finally, this dataset is used for instruction fine-tuning training.

Experiments on several benchmark datasets reveal that the LLM trained on the dataset constructed by PSI achieves comparable performance to those trained on datasets directly annotated by large-scale LMs. This phenomenon demonstrates the effectiveness of our principle-based instances generation method. Compared to directly using smaller-scale LMs to construct datasets, our principle-based method achieves consistently better performance. Additionally, we also explore the performance of variant types of datasets, the impact of different experiences on dataset quality, and a comparison of carbon emissions among various methods.

Our contributions are as follows:

- We propose a principle-based method PSI for constructing instruction tuning datasets, which reduces high carbon emissions and solves privacy leakage issues when using closed-source LLMs.
- We introduce the Multi-level Principle Generation method to build a low-redundancy, high-quality pool of task principles.
- We propose the Principle-based Instance Generation method to enable smaller-scale LLMs to follow task principles and generate high-quality instruction tuning datasets.
- We conducted extensive experiments on several benchmark datasets, demonstrating that our PSI achieves comparable performance to directly using large-scale LMs for annotation while significantly reducing carbon emissions.

2 Related work

Self-instruction (Wang et al., 2023a) has emerged as an effective method for utilizing LLMs to synthesize instruction fine-tune datasets. It starts from a set of manually written seed datasets and utilizes LLM to synthesize instances. `Alpaca` (Taori et al., 2023) propose to improve the performance of llama-7b through the distillation of a larger LLM, *e.g.,* `text-davinci-003`. Then, `Alpaca-GPT4` (Peng et al., 2023) achieve better performance by using the more powerful GPT-4 as the instance generation model. Taking advantage of the rewriting capabilities of LLMs, WizardLM (Xu et al., 2024) iteratively employs ChatGPT to rewrite initial instructions into increasingly complex instructions. For some specific domains such as math (Luo et al., 2023; Li et al., 2024a), tool-learning (Qin et al., 2024; Gao et al., 2024) and dialogue (Xu et al., 2023; Ding et al., 2023),

generating synthetic dataset via self-instruct has also achieved significant success.

However, most of the self-instruct frameworks heavily rely on powerful closed-source LLMs, leading to environmental impact and privacy risks. Therefore, in this paper, we focus on enhancing the instances synthesis capabilities of open-source smaller LLMs.

3 Task Definition

The self-instruct approach (Wang et al., 2023a) typically involves the following steps: First, we employ human annotators to construct a seed dataset $\mathcal{D}_{seed}$. Then $\mathcal{D}_{seed}$ is used as in-context examples for a large LM $\mathcal{M}_L$, with prompts designed to guide $\mathcal{M}_L$ to generate a larger dataset $\mathcal{D}_t$ consistent with $\mathcal{D}_{seed}$. Finally, the large dataset $\mathcal{D}_t$ is used to fine-tune a model $\mathcal{M}_t$.

In PSI, we also use a small seed dataset $\mathcal{D}_{seed}$ as in-context examples for a large language model $\mathcal{M}_L$. However, instead of directly generating a large-scale dataset by prompting the large LM $\mathcal{M}_L$, we design prompts to guide $\mathcal{M}_L$ to generate a set of task-completion principles $\mathcal{P}$. Then, we use the seed dataset $\mathcal{D}_{seed}$ and the task-related principles $\mathcal{P}$ as prompts to a smaller language model $\mathcal{M}_g$, which generates a dataset $\mathcal{D}_t$ similar to $\mathcal{D}_{seed}$ under the guidance of $\mathcal{P}$. Finally, the dataset $\mathcal{D}_t$ is used to fine-tune a model $\mathcal{M}_t$.

4 PSI Method

In our PSI method, there are two main steps. First, we use the Multi-level Principle Generation method, where the large-scale language model $\mathcal{M}_L$ reflects and summarizes a set of task-related principles $\mathcal{P}$ based on the given seed dataset $\mathcal{D}_{seed}$. Then, under the guidance of $\mathcal{P}$, we employ the Principle-based Instance Generation method to use a smaller language model $\mathcal{M}_g$ to generate a large-scale dataset $\mathcal{D}_t$ similar to $\mathcal{D}_{seed}$. The parameters of the language model $\mathcal{M}_g$ is significantly smaller than that of the large-scale language model $\mathcal{M}_L$ used for generating the principles $\mathcal{P}$. Thus, our method can significantly reduce carbon emissions.

4.1 Multi-level Principle Generation

To further reduce reliance on manual data annotation, we first use the language model $\mathcal{M}_g$ to perform a simple expansion of the manually annotated seed dataset $\mathcal{D}_{seed}$, resulting in an expanded dataset $\mathcal{D}_{initial}$:

$$\mathcal{D}_{initial} = \mathcal{M}_g(\mathcal{D}_{seed}, \mathcal{K}, \mathcal{I}_s), \quad (1)$$

where $\mathcal{K}$ represents the external knowledge required for instances generation, and $\mathcal{I}_s$ is the prompt used to guide the language model $\mathcal{M}_g$ in instances augmentation. It is important to indicate that although $\mathcal{D}_{initial}$ includes some external knowledge, the dataset remains relatively small ($|\mathcal{D}_{seed}| < |\mathcal{D}_{initial}| \ll |\mathcal{D}_t|$). This allows us to filter instances or anonymize the knowledge within $\mathcal{D}_{initial}$, thus mitigating the risk of privacy leakage.

Subsequently, to obtain more diverse task-completion principles, we perform multiple random samplings on the dataset $\mathcal{D}_{initial}$. Then, we obtain several subsets $\{d_1, d_2, \ldots, d_T\}$, where each subset d_i contains multiple task instances, and T is the number of subsets. We then use the large-scale language model $\mathcal{M}_L$ to reflect on each subset d_i and summarize **low-level principle** $\mathcal{P}_i^L$:

$$\mathcal{P}_i^L = \mathcal{M}_L(d_i, \mathcal{I}_p), \quad (2)$$

where $\mathcal{P}_i^L$ includes several natural language descriptions of task principles, and $\mathcal{I}_p$ is the prompt guiding the language model $\mathcal{M}_L$ to generate the low-level task principles:

> <INSTRUCTIONS DATA>
> Conduct a thorough analysis of the given instructions output pairs. Provide clear principles that can be derived from this analysis to improve future and outputs. We are not focused on this one data point, but rather on the general principle.

Thus, we obtain the set of low-level task principles for each subset, $\mathcal{P}^L = \{\mathcal{P}_1^L, \mathcal{P}_2^L, \ldots, \mathcal{P}_T^L\}$.

Intuitively, some low-level task principles may be redundant or overly specific. To address this, we first vectorize all low-level task principles $\mathcal{P}^L$ using a semantic embedding model. Then, we apply a clustering algorithm to these semantic representations, resulting in N clusters of task principles, $\mathcal{P}^C = \{\mathcal{P}_1^C, \mathcal{P}_2^C, \ldots, \mathcal{P}_N^C\}$. Specifically, we use a soft clustering algorithm that doesn't require a pre-defined number of clusters.

To reduce computational load and further reflect on the task principles, we use the language model $\mathcal{M}_L$ to summarize each task principle cluster to construct the **high-level principle**:

$$\mathcal{P}_i^H = \mathcal{M}_L(\mathcal{P}_i^C, \mathcal{I}_h), \quad (3)$$

where $\mathcal{P}_i^H$ is a high-level task principle, and $\mathcal{I}_h$ is the prompt guiding the language model $\mathcal{M}_L$ to generate high-level task principles:

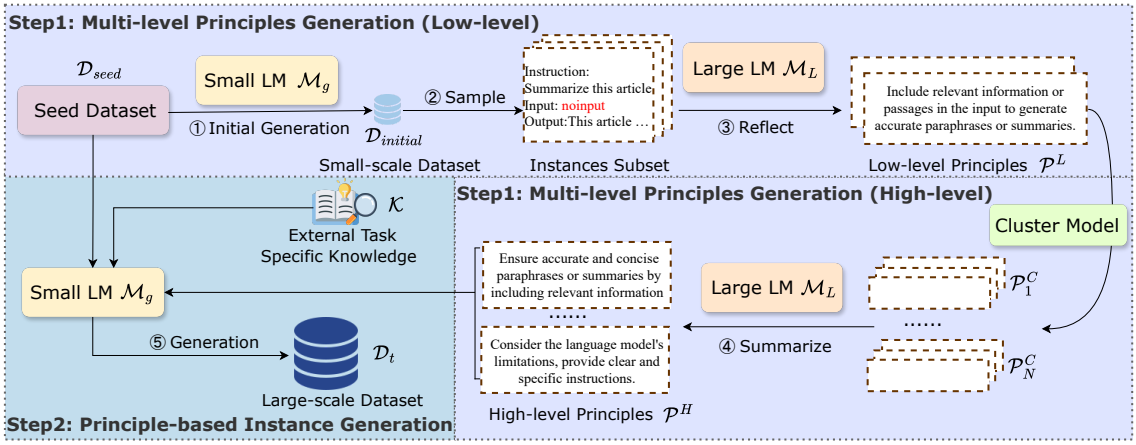

Figure 2: The architecture of our proposed PSI. The Multi-level Principle Generation consists of two parts: First, we employ large LM $\mathcal{M}_L$ to generate low-level principles. Second, these low-level principles will be clustered and summarized by $\mathcal{M}_L$ into high-level principles $\mathcal{P}^H$. The Principle-based Instance Generation uses a smaller LM $\mathcal{M}_g$ to generate instances guided by high-level principles $\mathcal{P}^H$ and external knowledge $\mathcal{K}$.

<LOW LEVEL PRINCIPLES>
Create a high level and insightful principle to improve future responses. Focus on capturing the essence of the feedback while eliminating redundancies.

Then, we obtain the high-level task principle set $\mathcal{P}^H = \{\mathcal{P}_1^H, \mathcal{P}_2^H, \ldots, \mathcal{P}_N^H\}$, containing N high-level principles necessary for completing the tasks.

4.2 Principle-based Instance Generation

Generating a large-scale instruction fine-tuning dataset is the most resource-intensive part of the self-instruction process. Therefore, after obtaining these task principles, to reduce reliance on the large-scale model $\mathcal{M}_L$, we use a smaller parameter model $\mathcal{M}_g$ to generate a large amount of instruction fine-tuning data:

$$d = \mathcal{M}_g(\mathcal{P}^H, \mathcal{K}, \mathcal{I}_g), \tag{4}$$

where d is a generated instruction tuning instance, and $\mathcal{I}_g$ is the prompt guiding model $\mathcal{M}_g$ to generate instances based on the task principles $\mathcal{P}^H$:

You are asked to come up with a set of 20 diverse task instructions.
The following insights and guidelines may improve responses:
<PRINCIPLES>

By repeatedly executing Equation 4, we can obtain a large-scale instruction fine-tuning dataset $\mathcal{D}_t$, where $|\mathcal{D}_t| \gg |\mathcal{D}_{seed}|$. Since $\mathcal{M}_g$ has significantly fewer parameters than model $\mathcal{M}_L$, the carbon emissions during the generation of a large dataset are greatly reduced. Additionally, model $\mathcal{M}_L$ only receives the small-scale dataset $\mathcal{D}_{initial}$ as in-context learning examples throughout the process and does not use our external knowledge $\mathcal{K}$. This also prevents the leakage of the private knowledge base $\mathcal{K}$ to external model service providers. Finally, we use the dataset $\mathcal{D}_t$ to fine-tune a language model $\mathcal{M}_t$.

5 Experimental Setup

5.1 Datasets

To verify the effectiveness of our proposed PSI, we use the generated instruction tuning dataset to fine-tune a model $\mathcal{M}_t$, and then evaluate the model performance of $\mathcal{M}_t$ using the following benchmark datasets. We classify the datasets into four categories according to their task definition:

1. Truthfulness and Knowledge. TruthfulQA is a popular benchmark used to test the model's hallucination (Lin et al., 2022). We utilized it to gauge the truthfulness of models, reporting accuracy under a 0-shot setting. And we use **MMLU** (Hendrycks et al., 2021) to assess the factual knowledge of models and report the mean accuracy under a 5-shot setting.

2. Commonsense Reasoning. WinoGrande is to choose the right option for a given sentence formulated as a fill-in-a-blank (Sakaguchi et al., 2021). **GPQA** (Rein et al., 2023) is a multiple-choice written by domain experts. Both of them require the model to have commonsense reasoning abilities. We report the accuracy of these two datasets under 5-shot and 3-shot settings.

3. Coding. We test the programming capabilities

using HumanEval (Chen et al., 2021) and perform the decoding using two different temperatures: 0 and 0.8. We report the better Pass@10 from these two decoding results.

4. Math. We use the GSM8K dataset (Cobbe et al., 2021) to evaluate the ability of solving multi-step mathematical reasoning problems and report the exact match under the 5-shot setting.

5.2 Baselines

We compare our method with several state-of-the-art self-instruction methods:

Alpaca (Taori et al., 2023) is 52K instruction-tuning dataset generated by text-davinci-003 using the self-Instruct technique.

Alpaca-GPT4 (Peng et al., 2023) follows the same methodology as Alpaca, but incorporates GPT-4 as its teacher model.

WizardLM (Xu et al., 2024) uses Evol-Instruct to rewrite Alpaca dataset step by step into more complex instructions.

Alpagasus (Chen et al., 2024) uses the robust ChatGPT to score and select 9K instances from the original Alpaca dataset.

5.3 Implementation Details

In our experiments, we employ two LLMs as the instances generation model $\mathcal{M}_g$ to verify the generalization of our proposed method: Zephyr-7B-Beta (Tunstall et al., 2023) and Llama-3-8B-Instruct (Meta, 2024). And we also use the generated dataset to fine-tune two language models $\mathcal{M}_t$: Qwen-1.8B (Bai et al., 2023) and Gemma-2B (Team et al., 2023). We construct several instruction tuning datasets, each containing 20K examples, using our proposed PSI method and several other baseline methods (*e.g.,* Alpaca). We set the number of subsets T to 10, and the size of each subset, $|d_i|$, to 10. All our experiments are conducted on 4 $\times$ NVIDIA A800 (80GB) GPUs, with the same setting and hyperparameters of Alpaca. We employ gpt-3.5-turbo from OpenAI as the $\mathcal{M}_L$ to reflect and summarize. For our analytical experiment, we employ Zephyr-7B as $\mathcal{M}_g$ and Gemma-2B as $\mathcal{M}_t$.

6 Experimental Results

6.1 Overall Performance

Table 1 shows the evaluation results of our proposed PSI and other self-instruct baselines. Compared to directly using large LMs for instances generation (*e.g.,* `Alpaca w/ GPT4`), our PSI w/ Zephyr-7B achieves comparable performance on two backbone models (including Gemma-2B and Qwen-1.8B). On the GPQA and Winogrande datasets, it even outperformed models that used GPT-4 for instance generation.

From Table 1, it can be seen that although the performance of PSI is slightly lower than that of models trained with the dataset generated by GPT-4, it outperforms models that directly use smaller LMs to generate instances (*e.g.,* Zephyr) on most of the datasets. We can find that compared with the `Alpaca w/ Zephyr` which directly employs the Zephyr as instances generation model $\mathcal{M}_g$, our proposed PSI achieves consistent improvement when using different backbone LLMs (*e.g.,* Gemma-2b and Qwen-1.8B). This indicates that the task principles effectively help these smaller LMs generate a higher-quality instruction-tuning dataset. To demonstrate the generalization ability, we also conduct experiments using Llama-3 as the instance generation LLM. The PSI w/ Llama3 outperforms the `Alpaca w/ Llama3` in the majority of datasets.

We also find that the improvements of PSI are not consistent across different types of datasets. In most knowledge-centric datasets (*e.g.,* MMLU), the performance improvement of our model compared with directly using smaller models for instance generation is not as high as it is for other types of datasets. It is intuitive that task principles usually only provide higher-level guidance on methodology or instance formats for completing tasks, whereas knowledge-based datasets require extensive knowledge content as support, which is not included in the task principles. For logical reasoning tasks (*e.g.,* GPQA), the problem-solving experience included in the task principles can effectively help the model generate high-quality data. Therefore, our PSI achieves better improvements than the baselines without principles on logical reasoning tasks.

6.2 Analysis of Carbon Emission

Since one of the motivations of our proposed PSI is to reduce carbon emission when generating large-scale instruction tuning dataset, we compare the carbon emission between PSI w/ Zephyr-7B and `Alpaca w/ GPT4` as instances generation LLM. The detailed calculation method of carbon emission is illustrated in Appendix A.3. Additionally, we directly report the number of LLM tokens

Method	Instance Gen LLM $\mathcal{M}_g$	MMLU	TruthfulQA	GPQA	Winogrande	GSM8K	HumanEval	Overall
Base Model ($\mathcal{M}_t$): Gemma-2B	-	40.7	33.2	23.4	65.1	18.5	34.2	35.9
Alpaca	text-davinci-003	38.7	38.2	24.6	66.9	13.3	35.4	36.2
AlpaGasus	gpt-3.5-turbo	41.6	36.3	27.0	66.5	15.5	37.8	37.5
WizardLM	gpt-3.5-turbo	41.0	43.2	**27.5**	64.9	19.4	**44.5**	**40.1**
Alpaca	GPT-4	**42.3**	**43.6**	25.7	65.8	20.2	37.2	39.1
Alpaca	Zephyr-7B	40.4	42.0	25.0	65.8	17.3	31.7	37.0
PSI	Zephyr-7B	$41.7_{\uparrow 1.3}$	$43.5_{\uparrow 1.5}$	$26.3_{\uparrow 1.3}$	$66.1_{\uparrow 0.3}$	$19.0_{\uparrow 1.7}$	$36.6_{\uparrow 4.9}$	$38.9_{\uparrow 1.9}$
Alpaca	Llama3-8B	40.9	42.9	25.7	65.7	19.4	35.4	38.3
PSI	Llama3-8B	$39.9_{\downarrow 1.0}$	$41.8_{\downarrow 1.1}$	$26.6_{\uparrow 0.9}$	**66.9** $_{\uparrow 1.2}$	**22.3**$_{\uparrow 2.9}$	$37.8_{\uparrow 2.4}$	$39.2_{\uparrow 0.9}$
Base Model ($\mathcal{M}_t$): Qwen-1.8B	-	45.3	39.4	27.2	61.6	34.5	32.9	40.2
Alpaca	text-davinci-003	45.3	37.5	28.4	61.3	17.7	34.8	37.5
AlpaGasus	gpt-3.5-turbo	46.5	38.5	27.5	61.6	26.9	**36.0**	39.5
WizardLM	gpt-3.5-turbo	46.3	39.9	**29.2**	62.5	**36.6**	25.0	39.9
Alpaca	GPT-4	46.2	**42.3**	28.4	60.9	34.4	34.8	**41.2**
Alpaca	Zephyr-7B	45.6	41.6	25.5	62.2	29.4	27.4	38.6
PSI	Zephyr-7B	**46.9**$_{\uparrow 1.3}$	$41.3_{\downarrow 0.3}$	$28.8_{\uparrow 3.3}$	**63.3**$_{\uparrow 1.1}$	$31.0_{\uparrow 1.6}$	$29.3_{\uparrow 1.9}$	$40.1_{\uparrow 1.5}$
Alpaca	Llama3-8B	45.7	36.1	28.1	61.1	31.9	26.8	38.3
PSI	Llama3-8B	$45.8_{\uparrow 0.1}$	$40.4_{\uparrow 4.3}$	$29.0_{\uparrow 0.9}$	$62.1_{\uparrow 1.0}$	$33.1_{\uparrow 1.2}$	$35.4_{\uparrow 8.6}$	$41.0_{\uparrow 2.7}$

Table 1: The performance of two base models: Gemma-2B and Qwen-1.8B, which are trained on the instruction tuning datasets generated by our proposed PSI and baselines respectively.

Method	Token Consume		Carbon Emission
	$\mathcal{M}_L$ (GPT)	$\mathcal{M}_g$ (Zephyr)	(kgCO_{2e})
AlpaGasus	5,345,600	-	40.72
WizardLM	7,834,077	-	34.74
Alpaca w/ GPT4	3,033,669	-	1.74
PSI	18,264	3,934,321	0.49▼

Table 2: Comparison between PSI and baseline methods in terms of token consumption and carbon emission.

Method	MMLU	GSM8K	HumanEval	GPQA
PSI	41.7	19.0	36.6	26.3
- *w/o initial*	41.6	17.7	34.8	26.1
- *w/o sample*	41.5	18.6	30.5	24.8
- *w/o cluster*	40.7	18.7	31.1	25.0

Table 3: Ablation study on four types of datasets.

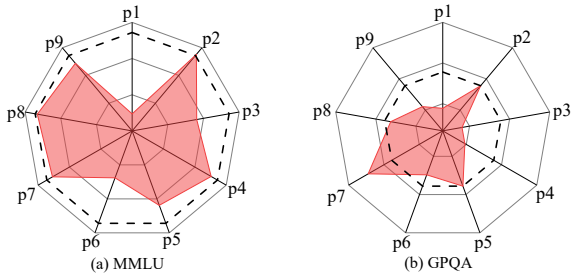

Figure 3: Effectiveness evaluation of each principle. Each vertex of the radar chart represents the performance of the model trained after removing the principle from the set. The dashed circle indicates the performance of the model using the whole set.

used by different instances generation LLMs. Table 2 shows a comparison between these two methods. From Table 2, it is evident that our PSI method significantly reduces carbon emissions when generating large-scale instances (with p-value < 0.05), which demonstrates that PSI is greener than directly using large LM to generate instances.

6.3 Ablation Study

To verify the effectiveness of each module in our PSI, we employ three variant models that remove each module and conduct experiments on each type of dataset. *w/o initial* indicates no initial generation, using $\mathcal{D}_{seed}$ to replace with $\mathcal{D}_{initial}$. *w/o sample* means there are no samples, and $\mathcal{M}_L$ evaluates as much instances as possible within its context length. *w/o cluster* signifies no clustering, utilizing a set of low-level principles to guide generation. Table 3 shows the performance of each ablation model on four datasets, and we can find that removing any module results in a certain degree of performance decline, which validates the necessity of each module.

6.4 Effect of Different Principles

In the PSI, we employ the $\mathcal{M}_L$ to generate several task principles to guide the $\mathcal{M}_g$ to generate a large-scale dataset. Therefore, an intuitive question arises: *Is each generated principle useful for improving the quality of the dataset?* In this section, we individually remove each principle, then use the remaining set of principles to generate data, and finally fine-tune the model. Figure 3 shows the performance changes of the model on two datasets after removing each principle. From Figure 3, we can find that the performance

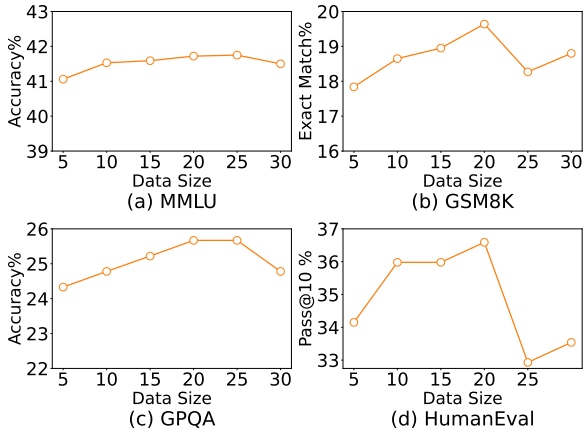

Figure 4: Model performance of using different scales of dataset.

declines compared to using the full set of principles after removing any principle, which indicates the necessity of these principles. Additionally, we also found that some principles are crucial for instances quality. For example, we show the principle $\mathcal{P}_1^H$ from the MMLU and GPQA dataset:

> Ensure clear and concise communication by aligning the output with the prompt, maintaining proper formatting and grammar, addressing the task accurately and comprehensively, and organizing the output logically for better readability.

This principle describes a critical method of generating data instances, which enables the model to generate comprehensive and detailed outputs aligned with the instruction. Thus, this experiment demonstrates that our proposed PSI can generate valuable task principles, thereby helping the model generate high-quality datasets.

6.5 Analysis of Dataset Size

In the experiments shown in Table 1, all methods generate 20K samples for model training. In this section, we explore the impact of generating datasets of different sizes. Figure 4 shows the model performance changes when using datasets of varying sizes, with the number of task principles kept constant. From Figure 4, we can find that as the dataset size increases from 5K to 20K, the performance of our model gradually improves. However, when the dataset size exceeds 20K, the performance no longer increases. The reason for this phenomenon is evident that with a fixed number of task principles, a dataset with enough samples is sufficient for the model to fully grasp these principles, while an excessively large dataset leads to the model overfitting these principles.

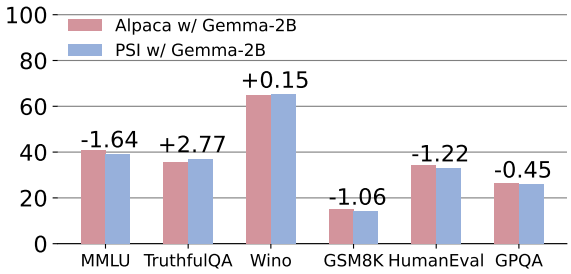

Figure 5: Comparison between PSI and Alpaca which both use a small-scale LM as the instance generation model.

6.6 Analysis of Different Backbone LLM

In our experiments, we utilized the 7B model for generating instances as the $\mathcal{M}_g$, while a smaller 2B model was employed for fine-tuning. An intuitive experiment is to see whether a smaller model with 2B parameters could effectively generate instances. In this section, we replace the $\mathcal{M}_g$ in PSI with Gemma-2B and use the generated dataset to finetune another Gemma-2B model. For comparison, we employ the same instance generation model in `Alpaca`. Figure 5 illustrates the comparison results. It shows that the performance of the model trained with instances generated by the smaller Gemma-2B is comparable with the model trained with `Alpaca`, which does not use any principles. This outcome indicates that the task principles did not help the model generate higher-quality data. An intuitive reason is that the smaller model cannot understand the complex instructions (*e.g.,* following task principles), leading to lower-quality datasets. Conversely, the results in Table 1 demonstrate that the instances generation LLM with 7B parameters can understand complex instructions and achieves better performance than `Alpaca` without using principles.

6.7 Analysis of Instances Length

Some research works (Zhao et al., 2024) have found that the length of the instruction tuning instances affects the performance. Thus, in this section, we compare the sample length distribution of our PSI and `Alpaca w/ Zephyr`. As shown in Figure 6, we can find that the samples generated by our PSI are longer than those generated by `Alpaca`. It indicates the effectiveness of our proposed Principle-based Instance Generation, which employs the task principles to guide the LLM to generate a high-quality dataset.

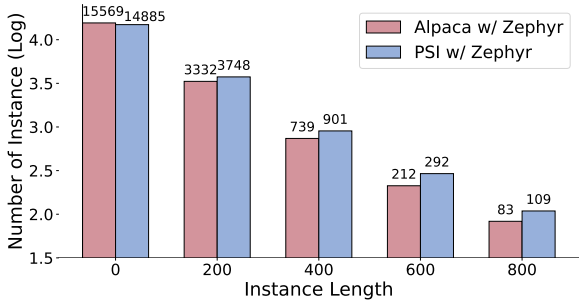

Figure 6: The instance length comparison between Alpaca and PSI.

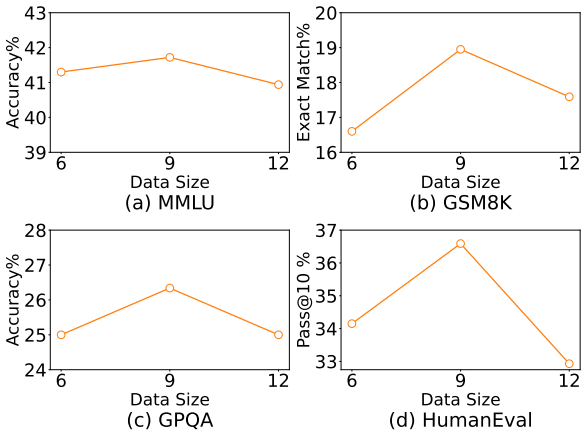

Figure 7: The model performance of using the different number of principles.

6.8 Analysis of Principle Number

In Equation 2, we randomly sample a subset d_i from the expanded dataset $\mathcal{D}_{initial}$ and then leverage the LLM $\mathcal{M}_L$ to summarize a low-level task principle $\mathcal{P}_i^L$. And we randomly sample multiple subsets d_i to construct principles. In this section, we analyze the effect of the principle number. From Figure 7, we can find that the model achieves the best performance when we sample 10 times. Additionally, neither increasing nor decreasing the number of samples can achieve better performance. The reason for this phenomenon is that too few times of samples may not encompass the experience needed to complete the task, while too many samples can result in redundant principles confusing the model $\mathcal{M}_g$.

6.9 Human Evaluation

Since the instruction following task aims at generating an open-ended response as the answer, we conduct a human evaluation to qualitatively evaluate the performance of our PSI. We randomly sample 40 instructions from each dataset: Dolly (Conover et al., 2023), Koala (Geng et al., 2023), and Vicuna (Chiang et al., 2023), and use the fine-tuned model to generate the response for each instruction. We employ 3 highly educated human annotators to evaluate the generated responses[1].

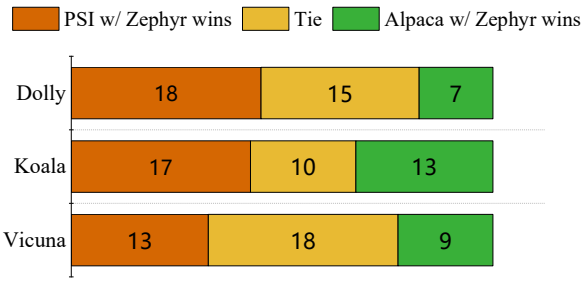

Figure 8: Human evaluation of the model trained by different self-instruct methods.

The Cohen's kappa for the human annotators is 0.56, indicating moderate inter-annotator agreement. Figure 8 compares the performance of models trained on datasets generated by PSI and `Alpaca`. As shown in Figure 8, on three datasets, the models trained on datasets generated by PSI achieved better performance. This further demonstrates the effectiveness of PSI.

7 Conclusion

In this paper, we propose **P**rinciple-based **S**elf-**I**nstruct (PSI) which is an environmentally friendly self-instruction framework that also avoids inputting task-related knowledge into proprietary LLMs, thereby protecting data privacy. Specifically, we first proposed the Multi-level Principle Generation method, which uses a large-scale LM to generate the task principles based on a given seed dataset. Then, by leveraging these task principles, a smaller-scale LM can be used to generate high-quality instruction-tuning datasets. We conducted extensive experiments on four types of instruction-following datasets, using various combinations of backbone LLMs. These experiments consistently demonstrated the effectiveness of our proposed PSI. Additionally, compared to directly using large-scale LMs (*e.g.,* GPT-4) for instances generation, our method significantly reduces carbon emissions while maintaining comparable performance.

[1]The detailed evaluation criterion for human annotators is shown in Appendix A.1.

Limitations

In this paper, we have conducted extensive experiments on four types of datasets, including knowledge, reasoning, coding, and math. Due to the limited space, some task-specific datasets (*e.g.,* tool learning and protein structure analysis) are not included in our experiments. However, these datasets have some task-specific features that our general framework does not include. We will expand our PSI to many other fields in our future work.

Ethical Considerations

In this paper, we propose the PSI for instruction-tuning data generation based on LLMs. Although the LLMs have finished the alignment training, this method cannot entirely prevent generating unsafe data. However, since the PSI generates data based on a small set of high-level task principles, we can constrain these task principles to minimize the generation of unsafe data. Additionally, to ensure complete safety, a manual review of the data is still required. Nevertheless, compared to manually constructing instruction-tuning datasets, only reviewing data significantly reduces the workload of annotators.

References

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. 2023. Qwen technical report. *arXiv*, abs/2309.16609.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. Alpagasus: Training a better alpaca with fewer data. In *ICLR*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv*, abs/2107.03374.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2024. Scaling instruction-finetuned language models. *JMLR*, pages 1–53.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. 2021. Training verifiers to solve math word problems. *arXiv*, abs/2110.14168.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Alex de Vries. 2023. The growing energy footprint of artificial intelligence. *Joule*, 7(10):2191–2194.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *EMNLP*, pages 3029–3051.

Jesse Dodge, Taylor Prewitt, Remi Tachet des Combes, Erika Odmark, Roy Schwartz, Emma Strubell, Alexandra Sasha Luccioni, Noah A Smith, Nicole DeCario, and Will Buchanan. 2022. Measuring the carbon intensity of ai in cloud instances. In *FAccT*, pages 1877–1894.

Shen Gao, Zhengliang Shi, Minghang Zhu, Bowen Fang, Xin Xin, Pengjie Ren, Zhumin Chen, Jun Ma, and Zhaochun Ren. 2024. Confucius: Iterative tool learning from introspection feedback by easy-to-difficult curriculum. In *AAAI*, pages 18030–18038.

Xinyang Geng, Arnav Gudibande, Hao Liu, Eric Wallace, Pieter Abbeel, Sergey Levine, and Dawn Song. 2023. Koala: A dialogue model for academic research. Blog post.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *ICLR*.

Chen Li, Weiqi Wang, Jingcheng Hu, Yixuan Wei, Nanning Zheng, Han Hu, Zheng Zhang, and Houwen Peng. 2024a. Common 7b language models already possess strong math capabilities. *arXiv*, abs/2403.04706.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. 2024b. Self-alignment with instruction backtranslation. In *ICLR*.

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *ACL*, pages 3214–3252.

Yilun Liu, Shimin Tao, Xiaofeng Zhao, Ming Zhu, Wenbing Ma, Junhao Zhu, Chang Su, Yutai Hou, Miao Zhang, Min Zhang, et al. 2023. Automatic instruction optimization for open-source llm instruction tuning. *arXiv preprint arXiv:2311.13246*, abs/2311.13246.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv*, abs/2308.09583.

Meta. 2024. Llama 3 model card. *GitHub repository*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *NIPS*, pages 27730–27744.

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *arXiv*, abs/2104.10350.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *ArXiv*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, dahai li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *ICLR*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. Gpqa: A graduate-level google-proof q&a benchmark. *ArXiv*, abs/2311.12022.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64:99–106.

Victor Sanh, Albert Webson, Colin Raffel, Stephen Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Teven Le Scao, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M Rush. 2022. Multitask prompted training enables zero-shot task generalization. In *ICLR*.

Parth Sarthi, Salman Abdullah, Aditi Tuli, Shubh Khanna, Anna Goldie, and Christopher D Manning. 2024. RAPTOR: Recursive abstractive processing for tree-organized retrieval. In *The Twelfth International Conference on Learning Representations*.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *ACL*, pages 3645–3650.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2023. Gemma: Open models based on gemini research and technology. *arXiv*, abs/2403.08295.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. 2023. Zephyr: Direct distillation of lm alignment. *arXiv*, abs/2310.16944.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023a. Self-instruct: Aligning language models with self-generated instructions. In *ACL*, pages 13484–13508.

Yue Wang, Xinrui Wang, Juntao Li, Jinxiong Chang, Qishen Zhang, Zhongyi Liu, Guannan Zhang, and Min Zhang. 2023b. Harnessing the power of david against goliath: Exploring instruction data generation without using closed-source models. *arXiv*, abs/2308.12711.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *ICLR*.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. In *EMNLP*, pages 6268–6278.

Da Yin, Faeze Brahman, Abhilasha Ravichander, Khyathi Chandu, Kai-Wei Chang, Yejin Choi, and Bill Yuchen Lin. 2023. Lumos: Learning agents with unified data, modular design, and open-source llms. *arXiv*, abs/2311.05657.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024. Long is more for alignment: A simple but tough-to-beat baseline for instruction fine-tuning. In *ICML*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. In *NIPS*, pages 55006–55021.

A Appendix

A.1 Prompts

We provide our prompts used to generate principles. The prompt to generate low-level principles is shown in Table 4. The prompt to generate high-level principles is shown in Table 5. The prompt guiding $\mathcal{M}_g$ to generate instances based on high-level principles is shown in Table 6

A.2 Human evaluation

We engaged three human annotators to label the data, following the same instruction as presented in (Chen et al., 2024) (see Table 7). Each annotator was required to choose the better response from the two options provided for each instruction. The response receiving the majority of votes was selected as the final result for each instruction.

A.3 Carbon emission calculation

Based on previous research (Patterson et al., 2021; Strubell et al., 2019; Dodge et al., 2022), we estimate the carbon emissions by calculating the total power required for generating and then multiplying it by the carbon emission intensity of the power grid used. It can be described using the following formula:

$$\text{Carbon Emissions} = \text{EC (kWh)} \times \text{CI (kgCO}_{2e}\text{/kWh)}$$

where the EC is the electricity consumption, refers to the total amount of electrical energy used in the generating process, and CI represents the carbon intensity. For a fair comparison, we use 0.24 kCO_{2e}/KWh, the carbon intensity in the Microsoft Azure US West region, for all following calculations.

Although reporting these operational emissions is standard practice, it overlooks other sources of emissions, such as those from the manufacturing, transportation, and disposal of hardware and data center infrastructure, lifetime operational emissions from usage, rebound effects, and other environmental impacts like water consumption and mining. Therefore, our estimates should be considered lower bounds.

For `Alpaca-GPT4`, due to the lack of detailed inference information disclosed by OpenAI, we can only make a preliminary estimation. We consider that ChatGPT require an estimated average electraicity consumption of 2.9 Wh per request based on (de Vries, 2023) . The average response generates 8 instances per request, so generating the 20k data we use requires 2.5k requests. The carbon emission is :

$$2.5k \times 2.9\text{Wh} \times 0.24 = 1.74\,\text{kgCO}_{2e}$$

For `AlpaGasus`, it must first generate 52k instances, and then score all the instances using ChatGPT 52k times. Its carbon emission can be calculated as follows:

$$(\frac{52}{8} + 52) \times 2.9 \times 0.24 = 40.72\,\text{kgCO}_{2e}$$

For `WizardLM`, the author obtained the 250k instances with requesting ChatGPT 624k times in (Xu et al., 2024). We only use 20k of them, the carbon emission can be calculated as follows:

$$624 \times \frac{20}{250} \times 2.9 \times 0.24 = 34.74\,\text{kgCO}_{2e}$$

For our PSI, we requested ChatGPT 10 times to reflect and summarize principles and ran a single A800 GPU with a power of 250W for 8 hours to generate instances. The carbon emission is :

$$\frac{10 \times 2.9 + 250 \times 8}{1000} \times 0.24 = 0.49\,\text{kgCO}_{2e}$$

A.4 Learned Principles

In this section, we show the high-level principles learned by PSI in Table 8 and Table 9.

A.5 Implementation Details

Training details During the training stage, we follow the practices in instruction tuning in `Alpaca` (Taori et al., 2023). We employed the AdamW optimizer to finetune the model $\mathcal{M}_t$ for 3 epochs. The initial learning rate is set to 2×10^{-5}, with a warm-up ratio of 0.03. The per GPU batch size is set to 8, resulting in a total batch size of 32, as we use 4 A800 GPUs for training.

Cluster model For clustering, we employ `all-mpnet-base-v2`[2] as the embedding model to encode the low-level principles $\mathcal{P}^L$. We then utilize the clustering algorithm described in (Sarthi et al., 2024), which applies soft clustering without requiring a pre-defined number of clusters to organize these principles.

[2] https://huggingface.co/sentence-transformers/all-mpnet-base-v2

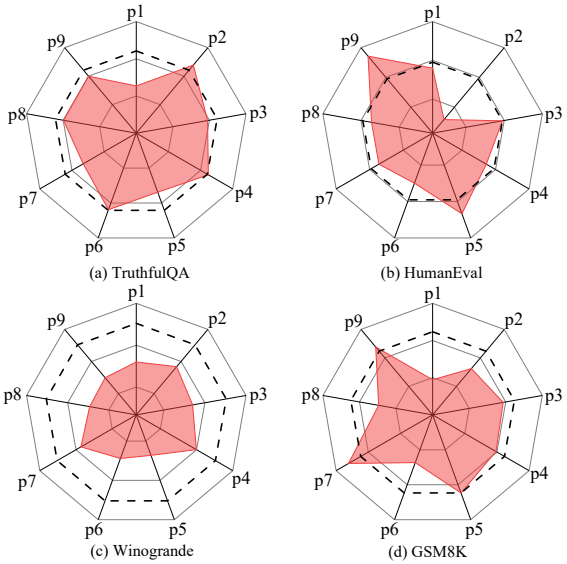

Figure 9: Effectiveness evaluation of each principle. Each vertex of the radar chart represents the performance of the model trained after removing the principle from the set. The dashed circle indicates the performance of the model using the whole set.

A.6 Additional Experimental Results

In section 6.4, we only provide the results on two datasets after removing each principle. To obtain clearer and more comprehensive results, we provide the outcomes for four additional datasets in Figure 9. From the figure, we can see that $\mathcal{P}_1^H$ still plays an important role in the results of TruthfulQA and GSM8K.

Prompt to generate low-level principles

The data instructions: {INSTRUCTIONS}.

You are an AI assistant. Conduct a thorough analysis of the given instructions output pairs. Identify the points that can be improved. Hallucinations, empty input and output, tasks that the language model cannot complete, etc. are all considered bad instructions. Provide clear insights, principles, or guidelines that can be derived from this analysis to improve future instructions and outputs. We are not focused on this one data point, but rather on the general principle.

Your output should follow the following format.

Reasoning: <discuss how the instruction generator could be improved>

Insights: <what principle should be looked at carefully to improve the instructions outputs quality in the future, given in points>

Table 4: The prompt to generate low-level principles.

Prompt to generate high-level principles

Low-level principles: {low_level_principles}

Create a high level and insightful principle to improve future responses based on the principles above. Focus on capturing the essence of the feedback while eliminating redundancies. Leave specific details in place.

Principle:

Table 5: The prompt to generate high-level principles.

Prompt to generate instances

You are asked to come up with a set of 20 diverse task instructions. These task instructions will be given to a GPT model and we will evaluate the GPT model for completing the instructions.
Here are the requirements:
1. Try not to repeat the verb for each instruction to maximize diversity.
2. The language used for the instruction also should be diverse. For example, you should combine questions with imperative instrucitons.
3. The type of instructions should be diverse. The list should include diverse types of tasks like open-ended generation, classification, editing, etc.
2. A GPT language model should be able to complete the instruction. For example, do not ask the assistant to create any visual or audio output. For another example, do not ask the assistant to wake you up at 5pm or set a reminder because it cannot perform any action.
3. The instructions should be in English.
4. The instructions should be 1 to 2 sentences long. Either an imperative sentence or a question is permitted.
5. You should generate an appropriate input to the instruction. The input field should contain a specific example provided for the instruction. It should involve realistic data and should not contain simple placeholders. The input should provide substantial content to make the instruction challenging but should ideally not exceed 100 words.
6. Not all instructions require input. For example, when a instruction asks about some general information, "what is the highest peak in the world", it is not necssary to provide a specific context. In this case, we simply put "<noinput>" in the input field.
7. The output should be an appropriate response to the instruction and the input. Make sure the output is less than 100 words.

The following insights and guidelines may improve responses:
{PRINCIPLES}

List of 20 tasks:

Table 6: The prompt to generate instances.

Evaluation criterion for human annotators

You'll be presented with a series of questions. For each question, two answers will be provided. Your task is to read both answers carefully and decide which one you believe is better. When judging, consider:

Relevance: Does the answer directly address the question?
Completeness: Is the answer comprehensive?
Coherence: Is the answer logically structured and easy to understand?
Accuracy: Is the information provided in the answer correct?

Question: <QUESTION>
Answer A: <ANSWER A> **Answer B:** <ANSWER B>

Comparing these two answers, which answer is better?
1. Answer A is significantly better.
2. Answer B is significantly better.
3. Neither is significantly better.

Table 7: The detailed evaluation criterion for human annotators.

Learned high-level principles for Zephyr-7B-beta by PSI

1. Ensure clear and concise communication by aligning the output with the prompt, maintaining proper formatting and grammar, addressing the task accurately and comprehensively, and organizing the output logically for better readability.
2. Consider the language model's limitations, provide clear and specific instructions, and anticipate potential constraints to ensure accurate and feasible responses.
3. Ensure accurate and concise paraphrases or summaries by including relevant information and guiding the model to provide simpler and more straightforward responses.
4. Ensure clear and relevant inputs to guide the AI in generating accurate and meaningful outputs.
5. Ensure clear and concise instructions that provide complete and specific information, avoiding ambiguity and open-endedness, while encouraging creativity and providing examples or templates when necessary.
6. Provide clear and detailed instructions that address the specific context and requirements of the task, including specific guidelines, prompts, and background information, to guide the generation of relevant and inclusive outputs.
7. Ensure comprehensive and accurate responses by understanding and incorporating user preferences, adhering to given restrictions, and providing detailed feedback, while avoiding fictional or irrelevant information.
8. Ensure meaningful and complete inputs and outputs to enhance the AI's understanding and delivery of relevant information.
9. Provide clear and concise instructions that guide the model to generate accurate, comprehensive, and effective outputs, while avoiding unsupported assumptions or hallucinations.

Table 8: The high-level principles for Zephyr-7B-beta.

Learned high-level principles for Llama3-8B-Instruct by PSI

1. Encourage comprehensive and critical analysis by providing clear instructions that include all relevant details, specify evaluation criteria, and request justification or reasoning, while also ensuring outputs include explanations or examples, final results when applicable, and distinct explanations of differences or similarities.
2. Provide clear and specific instructions: Clear and specific instructions, with relevant context and desired outcomes, help the AI assistant understand the task accurately and generate more accurate and meaningful responses. Avoid ambiguity and subjective interpretations by providing objective guidelines and criteria.
3. Continuous improvement through user feedback and iterative refinement enhances the quality, accuracy, and relevance of AI assistant responses, ensuring a better user experience and reliable assistance.
4. Continuous improvement: By continuously testing, iterating, and fact-checking instructions and outputs, while ensuring alignment between input and output, future responses can be improved to provide accurate, useful, and personalized information. Incorporating user feedback and considering creativity in the generation process will lead to better instruction outputs over time.
5. Promote creativity and engagement by providing clear guidelines and constraints, specifying format and content expectations, and offering specific prompts or guidelines for tasks that require creative writing.
6. Provide clear and comprehensive context: Instructions should include relevant context, examples, and specific details to guide the AI assistant in generating accurate and informative responses.
7. Align instructions with the language model's capabilities and limitations to ensure accurate and meaningful responses.
8. Ensure accurate and relevant responses by providing complete and specific instructions, avoiding hallucinations or unsupported claims, and maintaining consistency and coherence between inputs and outputs.
9. Clear and specific instructions: Instructions should provide clear and specific details to guide the model's response, including the desired outcome, required input format, expected output format, and any specific elements or explanations needed. Ambiguity should be avoided, and guidelines for multiple definitions or categorizations should be clarified.

Table 9: The high-level principles for Llama3-8B-Instruct.