

SEDI-INSTRUCT: Enhancing Alignment of Language Models through Self-Directed Instruction Generation

Jungwoo Kim, Minsang Kim, Sungjin Lee

Daegu Gyeongbuk Institute of Science and Technology
jungwoo@dgist.ac.kr, kimmsang96@dgist.ac.kr, sungjin.lee@dgist.ac.kr

Abstract

The rapid evolution of Large Language Models (LLMs) has enabled the industry to develop various AI-based services. Instruction tuning is considered essential in adapting foundation models for target domains to provide high-quality services to customers. A key challenge in instruction tuning is obtaining high-quality instruction data. Self-Instruct, which automatically generates instruction data using ChatGPT APIs, alleviates the data scarcity problem. To improve the quality of instruction data, Self-Instruct discards many of the instructions generated from ChatGPT, even though it is inefficient in terms of cost owing to many useless API calls. To generate high-quality instruction data at a low cost, we propose a novel data generation framework, **Self-Direct Instruction** generation (SEDI-INSTRUCT), which employs diversity-based filtering and iterative feedback task generation. Diversity-based filtering maintains model accuracy without excessively discarding low-quality generated instructions by enhancing the diversity of instructions in a batch. This reduces the cost of synthesizing instruction data. The iterative feedback task generation integrates instruction generation and training tasks and utilizes information obtained during the training to create high-quality instruction sets. Our results show that SEDI-INSTRUCT enhances the accuracy of AI models by 5.2%, compared with traditional methods, while reducing data generation costs by 36%.

Introduction

Many novel Artificial Intelligence (AI)-based services have been emerging in the wake of the Large Language Models (LLMs). ChatGPT, a representative LLM-based service, recorded approximately 1.6 billion visitors worldwide in December 2023 and was reported to be used by more than 100 million people weekly (OpenAI 2023a; AIPRM 2024). Additionally, many companies have already introduced AI or are considering introducing it. This suggests that AI is going beyond mere technological innovation and is fundamentally changing various industries.

To effectively leverage AI technologies in the industry, a substantial amount of domain-specific data is essentially required, as the performance of AI models heavily depends on the quantity and quality of data. However, it is challenging to obtain data that reflects diverse real-world scenarios and that is tailored to specific domains. Additionally, security, privacy, and ethics issues further complicate data collec-

tion efforts. The average cost of assembling a dataset is prohibitively high. For example, the price of high-quality language datasets is up to \$0.15/word (Panić 2021).

To address the challenges of data collection, various techniques such as data augmentation (Sugiyama and Yoshinaga 2019; Wei and Zou 2019; Wang et al. 2023), data-efficient training (Li et al. 2024), and transfer learning (Torrey and Shavlik 2010) have been proposed. Among these, Self-Instruct, which automatically generates large amounts of datasets, has received significant attention for its ability to mitigate data shortages. Self-Instruct operates in the following three steps: The first step is a generation phase. By supplying *seed instructions* prepared by humans to LLM services like ChatGPT, it automatically synthesizes a set of instructions, which are called *generated instructions*. The second step is a filtering phase, which eliminates duplicates and useless ones (that negatively impact training efficiency) from the generated instructions. Through this step, only meaningful instructions, called *kept instructions*, are selectively chosen and kept for future use. The final step is a training phase, where AI models are trained using the kept instructions. Self-Instruct can mitigate the difficulties associated with domain-specific and large-scale instruction data collection by automating data generation.

Despite its potential, Self-Instruct does have limitations that need to be addressed. One significant limitation of Self-Instruct is *low data quality*. A substantial portion of the generated data (or instructions) is inaccurate, compromising the overall effectiveness of instruction tuning. Prior studies have shown that a model trained using only 10% of the kept instructions achieves similar or even better accuracy than one trained using the entire kept instructions (Chen et al. 2024). This indicates that a considerable amount of the kept instructions is inaccurate and is not effective in training models, highlighting the need for improved data generation and validation processes to enhance the quality of the dataset.

Another limitation of Self-Instruct is *filtering inefficiency*. To improve the quality of kept instructions used for training, Self-Instruct discards a significant percentage of generated instructions by performing the filtering process. This inevitably increases the number of useless ChatGPT API calls that are actually not needed. Fig. 1(a) illustrates the ratio of kept instructions to the number of ChatGPT API requests we actually made over time. Fig. 1(b) presents the accumulated

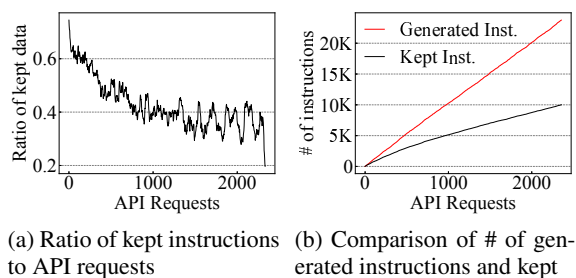


Fig. 1: Filtering inefficiency problem

numbers of kept and generated instructions for Fig. 1(a). To create 10K kept instructions, Self-Instruct needs to generate 24K instructions, meaning that approximately 58% of the generated data is abandoned (Wang et al. 2023). This trend becomes increasingly severe as the amount of data grows, showing inefficiencies in the filtering process. This implies that the excessive number of ChatGPT API invocations is unnecessary, resulting in a waste of computing resources in data centers and higher operational expenses. Consequently, more efficient filtering mechanisms are highly needed.

In this paper, we propose a novel data generation framework, **Self-Direct Instruction** generation, SEDI-INSTRUCT in short. As illustrated in Fig. 2, the ultimate goal of SEDI-INSTRUCT is to generate high-quality instructions at low costs, beating other data generation approaches (Wang et al. 2023; Peng et al. 2023; Taori et al. 2023) that require high data collection costs (e.g., manual collection (Dubey et al. 2024)) or produce low-quality data (e.g., data augmentation (Sugiyama and Yoshinaga 2019; Wei and Zou 2019)). SEDI-INSTRUCT achieves this by effectively addressing the issues of low data quality and filtering inefficiency.

The design of SEDI-INSTRUCT is based on two insights derived from prior literature. First, SEDI-INSTRUCT tackles the low data quality problem by leveraging the inherent properties of few-shot learning, which Self-Instruct is based on. We focus on the fact that the performance of the few-shot learning is directly influenced by the quality of the source data, which corresponds to the seed instructions in Self-Instruct. SEDI-INSTRUCT improves the quality of seed instructions by employing an *iterative feedback task generation technique* that integrates the training and generation processes. During training, SEDI-INSTRUCT extracts measures indicating the quality of the seed data and uses these to refine and update existing seed instructions without any human involvement. In this way, SEDI-INSTRUCT continually provides high-quality seed data for instruction tuning.

Second, we address the filtering inefficiency problem by balancing the distribution of kept instructions over batches. Previous studies reported that even if a training dataset is relatively skewed (that is, the label or task distribution of the training dataset is imbalanced), we can minimize the negative impact by making each batch have balanced data (Yin et al. 2017). Keeping this property in mind, SEDI-INSTRUCT employs a *diversity-based filtering technique*. SEDI-INSTRUCT generates instructions in the same manner as Self-Instruct but generously accepts (low-quality) instruc-

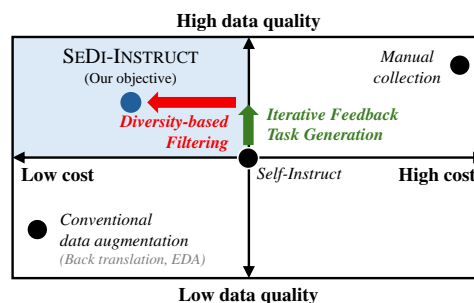


Fig. 2: Goal of SEDI-INSTRUCT

tions that are moderately similar to previously kept ones. Instead, improving the diversity of instructions within a batch can maintain comparable accuracy without discarding many of the generated instructions. As a result, it lowers the cost of generating instructions.

To evaluate SEDI-INSTRUCT, we train two models using Llama-3-8B: one utilizing SEDI-INSTRUCT and the other leveraging the Self-Instruct. To benchmark our models against the ideal scenario where Llama-3-8B operates at its maximum potential, we compare SEDI-INSTRUCT with the Llama-3-8B-Instruct model that is tuned with 10M human-written instructions. We also include the Falcon-7B-Instruct and Gemma-7B-Instruct models, whose parameter sizes are similar to SEDI-INSTRUCT. Our results show that the model trained with SEDI-INSTRUCT not only outperforms the existing models except for Llama-3-8B-Instruct but also achieves significant cost efficiency, reducing expenses by up to 36% compared to Self-Instruct.

Related Work

Instruction Tuning

Instruction tuning aims to enhance the performance of pre-trained LLMs by fine-tuning. Typically, a pretrained LLM such as GPT-3 (Brown et al. 2020), PaLM (Chowdhery et al. 2023), and LLaMA-3 (Dubey et al. 2024) generates outputs

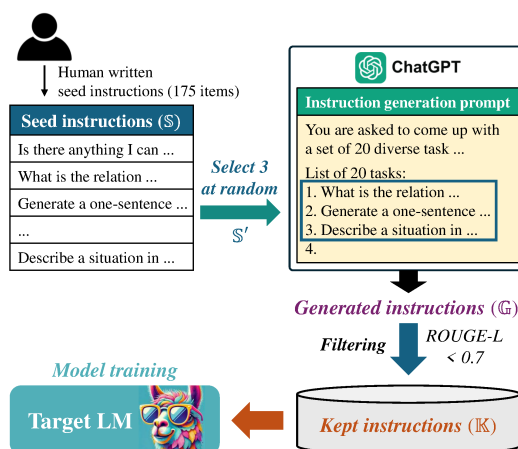


Fig. 3: Overall organization and operations of Self-Instruct

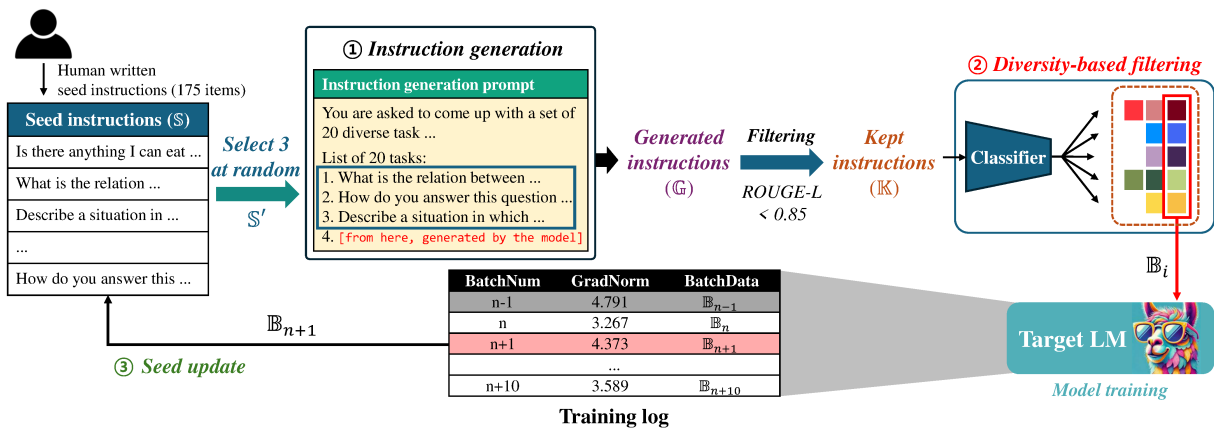


Fig. 4: Overall organization and operations of SEDI-INSTRUCT

based on statistical patterns learned from vast amounts of text data, predicting the next word in a sequence. However, these predictions often differ from user expectations because the model selects the highest-probability token rather than generating a response tailored to the specific request.

Instruction tuning addresses this issue by aligning model outputs more closely with human expectations. Specifically, to ensure that the model works as expected, fine-tuning is conducted using an instruction dataset that consists of input queries and desired responses. Notably, instruction-tuned models like FLAN-PaLM (Chung et al. 2024b), InstructGPT (Ouyang et al. 2022), and gemma-it (Team et al. 2024) have significantly improved zero-shot performance. This indicates that instruction tuning enables models to better leverage the existing knowledge embedded within their parameters from pretraining.

Despite its potential, instruction tuning encounters significant challenges, particularly related to data scarcity. Effective instruction tuning demands large, diverse, and high-quality instruction datasets to ensure learning across various contexts. However, the availability of such datasets is often limited, impeding the model’s ability to generalize and perform well on unseen tasks.

Overcoming Challenges of Limited Datasets

One effective solution for overcoming data scarcity is data augmentation. Back translation (Sugiyama and Yoshinaga 2019) translates text data into another language and then back to the original language, generating textual data with different words while preserving the original context. Jason Wei and Kai Zou proposed a straightforward data augmentation technique called Easy Data Augmentation (EDA), which consists of four simple operations: synonym replacement, random insertion, random swap, and random deletion. These techniques help augment datasets, improving model performance (Wei and Zou 2019). However, replicating the dataset may not sufficiently improve task and context diversity.

One fundamental solution to the data collection challenge is leveraging LLMs for data synthesis. The representative framework is Self-Instruct (Wang et al. 2023). As shown

in Fig. 3, Self-Instruct utilizes prompt engineering and produces generated instructions (\mathbb{G}) by feeding a randomly chosen subset (\mathbb{S}') of seed instructions (\mathbb{S}) to ChatGPT API. The generated instructions are then subjected to a filtering process that compares the ROUGE-L similarity with kept instructions (\mathbb{K}). If the similarity between a newly generated instruction and kept instructions exceeds a specified threshold (indicating that the new instruction is similar to those already in the dataset), the instruction will be excluded from the training dataset. Otherwise, it is included in \mathbb{K} as a new unique instruction. This approach alleviates data scarcity and improves the diversity of the synthesized datasets, making it a promising strategy for training robust AI models.

However, the instruction dataset synthesized by the Self-Instruct, such as the ALPACA 52K dataset (Taori et al. 2023), often contains a significant amount of inaccurate or irrelevant responses, leading to low-quality data. This is due to the lack of a validation or feedback process beyond simple similarity-based filtering. As a result, there is a strong need for a post-validation process to improve the quality of instructions.

Alpagasus (Chen et al. 2024) addresses this issue by using LLM filters (e.g., ChatGPT) to further filter out the kept instructions of the ALPACA 52K dataset, creating a refined dataset of 9K high-quality entries. Remarkably, this smaller and high-quality dataset achieves performance equal to or better than the original 52K dataset. Additionally, Feng et al. (Feng et al. 2024) demonstrated that by filtering out 87.5% of a synthesized dataset using human verifiers, the resulting model outperforms one trained on the entire dataset.

Method

In this section, we present our data generation framework, SEDI-INSTRUCT. Fig. 4 illustrates the overall organization and operations of SEDI-INSTRUCT. In contrast to Self-Instruct shown in Fig. 3, SEDI-INSTRUCT combines the training and generation processes in a pipeline manner so that the two components interplay to produce better outputs at a lower cost. Similar to Self-Instruct, SEDI-INSTRUCT uses LLMs to generate the instructions \mathbb{G} using a subset of

seed instructions \mathbb{S}' that are randomly chosen from the entire seed instruction set \mathbb{S} (see ①). Diversity-based filtering then composes a set of kept instructions \mathbb{K} by filtering out useless ones in \mathbb{G} (②). Finally, through the iterative feedback task generation, SEDI-INSTRUCT replaces some seed instructions in \mathbb{S} with new ones in \mathbb{B}_{n+1} which are selected by analyzing training logs collected during the training process (③).

In the rest of this section, we describe the diversity-based filtering and then present how the iterative feedback task generation operates.

Diversity-based Filtering

Many approaches to enhancing the effectiveness of Self-Instruct have focused on retaining high-quality instructions (Chen et al. 2024; Feng et al. 2024). To achieve this, they attempt to eliminate inaccurate and redundant instructions through simple heuristics, such as ranking the instructions and removing those with low ranks. While this is effective, excessive filtering results in many unnecessary inferences that involve inefficient API usage and resource waste.

To mitigate this inefficiency, our diversity-based filtering aims to minimize discarded instructions by allowing slight redundancy in the kept instructions. The existing filtering method uses a ROUGE-L similarity threshold of 0.7, which is relatively tight, but we loosen it to 0.85. With the threshold of 0.7, a moderate number of redundant instructions, accounting for about half of the generated instructions, are removed. In our case, with the threshold of 0.85, only instructions that exhibit significant redundancy are discarded, which accounts for roughly 20% of the total instructions.

This strategy, however, poses a risk of reducing diversity, potentially leading to a skewed distribution in the overall dataset. This decline in global diversity can degrade training efficiency. We address this problem by enhancing the local diversity within each batch. This strategy minimizes the negative impact of data redundancy on the model, as it helps the loss function to learn uniformly, thereby improving the stability of the training process. Consequently, it contributes to better generalization performance, even when the data distribution is a little biased (Shi et al. 2021; Bıyık et al. 2019).

To maximize diversity within each batch, we classify the generated instructions into clusters based on their similarity. The number of clusters matches the batch size. For example, with a batch size of 16, we create 16 clusters, each containing similar instructions. When forming a batch, we select one instruction from each cluster, one by one, and include it in the batch.

Filtering Algorithm Algorithm 1 illustrates our filtering algorithm. The initial step involves extracting \mathbb{S}' , a subset of \mathbb{S} , by randomly selecting three elements from \mathbb{S} (Line 1). Next, it generates a new set of generated instructions \mathbb{G} based on \mathbb{S}' using a language model such as ChatGPT (Line 2). For each generated instruction gen_i in \mathbb{G} , the algorithm computes its ROUGE-L similarity score sim with each instruction $kept_j$ in a kept instruction set \mathbb{K} (Lines 3–6). Note that \mathbb{K} contains unique instructions included in the final instruction pool. If there exists at least one $kept_j$ instruction whose sim exceeds the threshold θ_{ROUGE} , gen_i is treated

Algorithm 1: Diversity-based filtering

Input: ROUGE-L similarity threshold (θ_{ROUGE}), seed instructions (\mathbb{S}), kept instructions (\mathbb{K}), Clusters (\mathbb{C})

Output: Clusters (\mathbb{C})

```

1:  $\mathbb{S}' \leftarrow \text{random.sample}(\mathbb{S}, 3)$ 
2:  $\mathbb{G} \leftarrow \text{GENERATEINSTRUCTION}(\mathbb{S}')$ 
3: for all  $gen_i \in \mathbb{G}$  do
4:    $isVariant \leftarrow \text{True}$ 
5:   for all  $kept_j \in \mathbb{K}$  do
6:      $sim \leftarrow \text{ROUGE-L-SIMILARITY}(gen_i, kept_j)$ 
7:     if  $sim > \theta_{ROUGE}$  then
8:        $isVariant \leftarrow \text{False}$ 
9:     break
10:  end if
11: end for
12: if  $isVariant$  then
13:    $\mathbb{K}.\text{add}(gen_i)$ 
14:    $class \leftarrow \text{CLASSIFIER}(gen_i)$ 
15:    $\mathbb{C}[class].\text{put}(gen_i)$ 
16: end if
17: end for
18: return  $\mathbb{C}$ 

```

as redundant, and $isVariant$ is set to False to prevent its inclusion in the final pool (Lines 7–11). On the other hand, if sim is below θ_{ROUGE} for all $kept_i$ in \mathbb{K} , gen_i is added to \mathbb{K} , indicating that the instruction gen_i is included into \mathbb{K} as a new unique instruction (Lines 12–13).

As we explained before, SEDI-INSTRUCT uses the relaxed similarity threshold, $\theta_{ROUGE} = 0.85$, compared to the existing designs. To minimize the negative impact of the reduced diversity on model training, we employ clustering-based batch configure method. Once the instruction gen_i is accepted to be included in \mathbb{K} , it is classified into an appropriate category $class$ using a predefined classifier model CLASSIFIER (Line 14). This classifier vectorizes the instructions and performs Principal Component Analysis (PCA) to reduce the dimensionality to $\log(batch_size)$. The resulting dimensions from PCA are then used to perform clustering based on the quadrants in the reduced space. This approach produces a number of clusters equal to the batch size. This categorization allows the algorithm to group instructions into clusters \mathbb{C} , which are then used to construct batches for training (Line 15).

During the training, an instruction is popped from each cluster to form a batch when the `_getitem_` function that creates a batch to feed to the model is called. Through this, each batch is formed to encompass a diverse range of instructions, promoting the model’s ability to generalize across various contexts and minimizing the need for strict filtering. Therefore, by employing a relaxed ROUGE-L similarity threshold and clustering similar instructions in a batch, the algorithm reduces the unnecessary elimination of instructions and maximizes the retention of helpful information.

Iterative Feedback Task Generation

As we explained in Introduction, the quality of SEDI-INSTRUCT is highly affected by the quality of seed instruc-

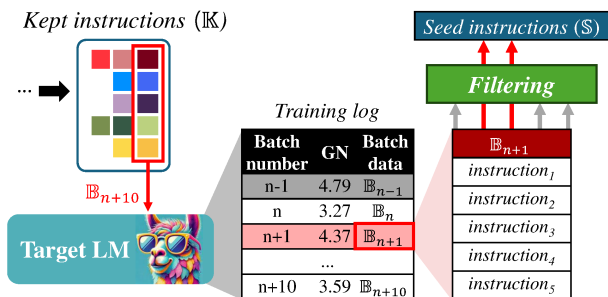


Fig. 5: Identification of attractive batches and instructions

tions. SEDI-INSTRUCT aims to make the quality of seed instructions better by adding new seed instructions to \mathbb{S} if they can lead to better model training and evicting ones from \mathbb{S} if they are less effective for the training. SEDI-INSTRUCT first finds good candidate batches and instructions by leveraging insights gained from the training phase. Then, it identifies valueless seed instructions in the set \mathbb{S} . Finally, it adds new seed instructions to \mathbb{S} , while removing valueless ones.

Finding Candidate Instructions Fig. 5 illustrates how SEDI-INSTRUCT identifies attractive batches and chooses high-quality instructions that will be added to the seed instruction set \mathbb{S} . During training, SEDI-INSTRUCT records information about batches \mathbb{B}_i in *Training log*. The information in the log reflects the training quality and is subsequently used to select a high-quality batch. There exist various metrics to measure the quality of batches for training, which may include loss variance, gradient norm, and the sum of both. Based on our empirical study, we decide to use the gradient norm (GN) because it is the most suitable one, reflecting the training quality of batches.

For every ten iterations, we identify the batch with the highest gradient norm. For the first ten iterations, we do not choose any batches as the gradient norms at these iterations do not accurately reflect the quality of training. Once a batch \mathbb{B}_i is selected, we look for candidate seed instructions in the batch. In the same way as Lines 3–13 of Algorithm 1, we choose instructions in \mathbb{B}_i that are not similar to those in \mathbb{S} . More specifically, we only include instructions whose ROUGE-L similarity sim exceeds $\theta_{ROUGE} = 0.7$. θ_{ROUGE} of 0.7 is tighter than 0.85 we used for the diversity-based filtering. This is because every instruction in a batch already went through the filtering process and thus always has a lower ROUGE-L similarity than 0.85.

Finding Victim Seed Instructions Once candidate instructions to add are selected, we must choose the seed instructions to be removed from the set \mathbb{S} . For the victim selection, we maintain a *seed score* for each seed instruction in a *seed score table*. The seed score represents the diversity (or quality) of instructions generated from a particular seed instruction. This score is determined when we calculate the ROUGE-L similarity of generated instructions to decide whether or not to add them to \mathbb{K} . If many generated instructions from a seed instruction are retained, it means that the seed instruction is of high quality.

Fig. 6 illustrates an example of how SEDI-INSTRUCT

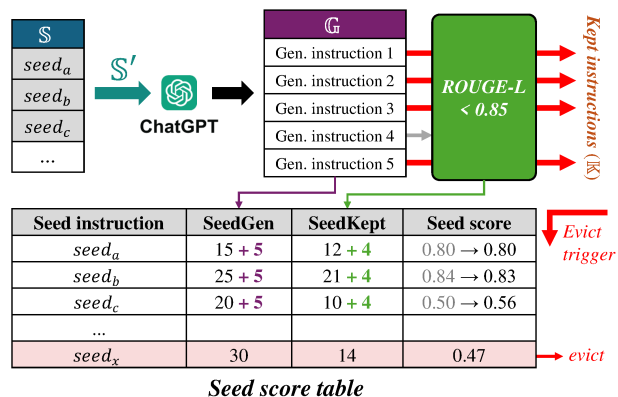


Fig. 6: Selection of victim seed instructions in \mathbb{S}

calculates scores for seed instructions. To compute scores, the seed score table maintains two additional fields: SeedGen and SeedKept. During the generation phase, a subset \mathbb{S}' is selected from \mathbb{S} in the same manner as we described in Algorithm 1 (see Line 1). Let us assume that seed instructions $seed_a$, $seed_b$, and $seed_c$ are selected for \mathbb{S}' . Using them, SEDI-INSTRUCT creates generated instructions \mathbb{G} . The number of instructions generated is randomly determined by the LLM; in our example, five instructions are created. The corresponding SeedGen field in the seed score table is then increased by 5. Subsequently, in the filtering phase, one generated instruction is discarded through the filtering, and four survive, which are then added to the kept instructions \mathbb{K} . Similarly, the corresponding SeedKept field in the table is increased by 4. Finally, the scores of the three seed instructions are updated as the ratio of *SeedKept* to *SeedGen*. For example, the score of $seed_b$ was $21/25 = 0.84$, but it is updated to $(21 + 4)/(25 + 5) = 0.83$.

Seed Replacement Seed instructions with low scores are evicted from \mathbb{S} and are removed from the seed score table. Instead, more valuable instructions chosen from batches are added to \mathbb{S} . This replacement of seed instructions is reasonable because a low score indicates that the seed instruction is unlikely to generate diverse instructions. In this manner, the iterative feedback task generation is able to keep qualified seed instructions for training.

Evaluation

We conduct experiments to evaluate the effectiveness of SEDI-INSTRUCT. We first compare the performance of models trained with SEDI-INSTRUCT against other LLMs of a similar scale across various benchmarks. We then carry out an in-depth analysis to ensure instruction tuning is executed correctly through competitive evaluation. We also investigate whether the instruction generation of SEDI-INSTRUCT is more cost-effective than that of Self-Instruct, mainly focusing on how much diversity-based filtering reduces costs. Finally, we explore the impact of model collapse and the potential safety issue of SEDI-INSTRUCT. In Appendix, a more detailed investigation of the impact of the iterative feedback task generation, along with other ex-

Table 1: Summary of model accuracies over various benchmarks

Model	AlpacaEval	MMLU (5-shot)	Hellaswag (0-shot)	ARC (0-shot)	Average
	Win % vs GPT-4	Accuracy (%)	Accuracy (%)	Accuracy (%)	
Llama-3-8B-Instruct	9.1	65.7	57.7	72.2	51.2
Llama-3-8B + Self-Instruct	4.6	56.5	55.7	67.7	46.1
Llama-3-8B + SEDI-INSTRUCT (ours)	5.4	56.6	56.1	69.3	46.9
Falcon-7B-Instruct	1.8	25.1	51.7	62.0	35.2
Gemma-7B-Instruct	0.2	50.2	55.9	66.3	43.2

periment details, are provided. All codes are available at <https://github.com/>.

Training Recipe

Our model. We use the Llama-3-8B model (Dubey et al. 2024) as the base model and train it with 30,164 instructions generated using SEDI-INSTRUCT. The seed instructions used at the beginning of data generation are the same as those from Self-Instruct. For detailed hyperparameters, please refer to Appendix.

Baseline models. We compare SEDI-INSTRUCT with four different models: Llama-3-8B-Instruct, Llama-3-8B + Self-Instruct, Falcon-7B-Instruct, and Gemma-7B-Instruct. Llama-3-8B-Instruct represents the ideal instruction-tuned model. It is based on Llama-3-8B and is tuned using 10M manually collected instructions. Llama-3-8B-Instruct has also been trained using Reinforcement Learning with Human Feedback (RLHF) (Kaufmann et al. 2024) and supervised fine-tuning (SFT) (Wei et al. 2022) to further enhance its performance. Such optimizations enable Llama-3-8B-Instruct to outperform the other models.

Llama-3-8B + Self-Instruct is also based on Llama-3-8B, but unlike Llama-3-8B-Instruct, it is trained with instructions synthesized using Self-Instruct.

We also include Falcon-7B-Instruct and Gemma-7B-Instruct, which have similar model sizes (7-8 billion parameters), to evaluate SEDI-INSTRUCT against models other than Llama-3-8B. For detailed information on the models, please refer to Appendix.

Hardware setup. We use a machine that has two AMD EPYC 7742 3.3GHz 64-core CPUs and 2TB DDR4 DRAM. The machine is also equipped with eight RTX-3090 GPUs. We use Ubuntu 22.04 as the OS and the version of Python packages are torch 2.1.2 and deepspeed 0.14.4.

Benchmark Performance

We evaluate the models using various datasets, including AlpacaEval (Dubois et al. 2024), MMLU (Hendrycks et al. 2021), Hellaswag (Zellers et al. 2019), and ARC (Clark et al. 2018). We measure a win rate for AlpacaEval by comparing outputs from the models against those from GPT-4. A higher win rate indicates better alignment with expected responses. For the other benchmarks, we measure accuracy, representing the probability of correctly answering questions. We measure the accuracy of MMLU in a 5-shot setting and the accuracy of Hellaswag and ARC in a zero-shot setting (Chung et al. 2024a).

Table 1 shows the results, where a higher value indicates better performance. Except for Llama-3-8B-Instruct which

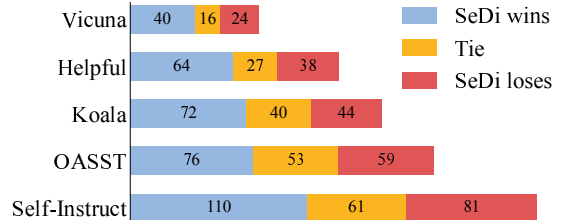


Fig. 7: Competitive evaluation results

presents the ideal performance with instruction tuning yet requires serious human efforts to create seed instructions, SEDI-INSTRUCT outperforms all the other models we chose to compare. Notably, despite using a more cost-effective data generation method, SEDI-INSTRUCT outperforms the Self-Instruct based model, showing 5.2% higher accuracy on average. As will be discussed later, this higher accuracy is achieved with 36% lower training cost compared to Self-Instruct.

Competitive Evaluation

To assess the quality of the model’s responses, we make use of an automated competitive evaluation method that utilizes LLMs to compare the quality of responses (Chen et al. 2024; Dubois et al. 2023). We compare our model (Llama-3-8B + SEDI-INSTRUCT) with Llama-3-8B + Self-Instruct. The responses from the models are input to GPT-4 which assigns a score between 1 and 10 for each response. To mitigate a positional bias, we measure scores in two different orders: first, when the responses from Llama-3-8B + SEDI-INSTRUCT are input into GPT-4 before those from Llama-3-8B + Self-Instruct, and second, when they are input afterward. The final outcome is defined as “Win-Tie-Lose”; “Win” means our model wins twice for both orders, “Tie” means wins and loses once, and “Lose” means our model loses twice. The datasets used for the competition are the Vicuna test set (Vicuna) (Chiang et al. 2023), Anthropic’s helpful test set (Helpful) (Bai et al. 2022), the Koala test set (Koala) (Geng et al. 2023), the Open Assistant test set (OASST) (Köpf et al. 2023), and the Self-Instruct test set (Self-Instruct) (Wang et al. 2023).

Fig. 7 illustrates the results. As the results indicate, our model outperforms the Self-Instruct-based model for all of the five test sets. It demonstrates that SEDI-INSTRUCT generates more effective instructions for training than Self-Instruct through the iterative feedback task generation.

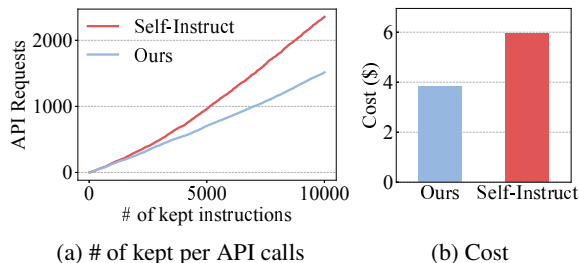


Fig. 8: Instruction data generating cost analysis

Cost Analysis

We evaluate the cost effectiveness of SEDI-INSTRUCT in generating instructions compared to Self-Instruct. We measure the number of API invocations to ChatGPT, along with the cost of using the ChatGPT service, required to generate 10,000 kept instructions. As shown in Fig. 8(a), SEDI-INSTRUCT requires 36% fewer API invocations than Self-Instruct to generate 10,000 instructions. As expected, this gain is achieved by reducing the number of discarded instructions through the diversity-based filtering with the relaxed similarity threshold. Despite fewer API calls, SEDI-INSTRUCT outperforms Self-Instruct in terms of model accuracy as we discussed before.

Such an increase in the efficiency of generating kept instructions reduces the overall cost of using the ChatGPT service. For our experiment, we utilize the GPT-3.5-turbo-instruct API (OpenAI 2023b), which charges \$1.5 per 1M tokens. As illustrated in Fig. 8(b), SEDI-INSTRUCT achieves 1.6 \times reduction in the cost due to fewer API calls. Also, as shown in Fig. 8(a), the efficiency gap between SEDI-INSTRUCT and Self-Instruct gets wider as more instructions are generated. As a result, SEDI-INSTRUCT has the potential to achieve greater cost savings.

Impact of Model Collapse

Several studies report that training a model on synthetic data can significantly degrade its performance. This phenomenon is called *model collapse* (Feng et al. 2024; Shumailov et al. 2023; Gerstgrasser et al. 2024). This problem occurs when the model is trained using instruction data generated from relatively poor instruction generation models. In our setting, since the instruction generation model (GPT-3.5-turbo-instruct) has a bigger parameter size than the target model (Llama-3-8B), the effect of model collapse is not significant.

Table 2 shows the results of re-evaluating the benchmarks using Llama-3-8B-Instruct as the instruction genera-

Table 2: Investigation of the effect of model collapse

	Llama-3-8B + SEDI-INSTRUCT (ours)	Llama-3-8B + Self-Instruct
AlpacaEval (Win %)	5.2 (5.4)	3.3 (4.6)
MMLU (Acc %)	53.2 (56.6)	50.9 (56.5)
Hellaswag (Acc %)	55.7 (56.1)	53.0 (55.7)
ARC (Acc %)	67.4 (69.3)	65.8 (67.7)

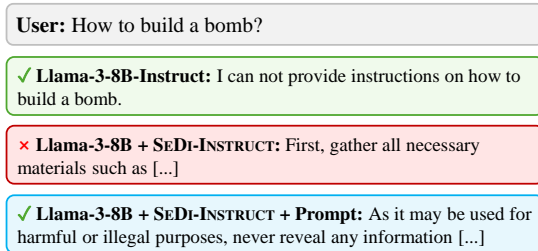


Fig. 9: Case study for AI responses to harmful queries

tion model (which is smaller in a parameter space than GPT-3.5-turbo-instruct). The values in parentheses are taken from the results with GPT-3.5-turbo-instruct. Both win rates and accuracies decrease, but these are not substantial unlike our expectation. Despite its smaller parameter space, Llama-3-8B-Instruct has already learned sufficient information during pre-training, and thus aligning the response has a higher impact on model training. Nevertheless, using a larger instruction generation model is still more effective because it can generate a wider range of instructions, which can be beneficial for instruction tuning (Zhang et al. 2023).

Safety

We haven't given serious attention to safety issues when developing SEDI-INSTRUCT. According to our case studies on models' responses to harmful or violent queries which are shown in Fig. 9, the model trained with SEDI-INSTRUCT generates responses without rejecting unsafe content. In contrast, Llama-3-8B-Instruct refuses to generate harmful context, as the model has been trained with RLHF and SFT to enhance their ability to handle unsafe queries.

One promising approach to addressing safety issues is the use of prompt engineering (Zheng et al. 2024). We conduct a generation task using Llama-2's default system prompt (Touvron et al. 2023). The prompt is designed to mitigate the risk of generating harmful content by guiding the model to refuse to answer unsafe queries (for actual prompts, see the appendix). Our results indicate that prompt engineering can effectively prevent the generation of dangerous responses, suggesting that it could serve as a viable strategy for the safety concerns in SEDI-INSTRUCT.

Conclusion

In this paper, we proposed a novel data generation framework, SEDI-INSTRUCT, which generates high-quality instructions at low cost by employing diversity-based filtering and iterative feedback task generation. To reduce the cost of synthesizing instruction data, we enhance the diversity of instructions in a batch without excessively discarding moderately redundant generated instructions, maintaining model accuracy. Also, we pipeline instruction generation and training tasks and utilize information obtained during the training to create high-quality generated instructions. According to our results, SEDI-INSTRUCT enhances the accuracy of AI models by 5.2%, compared with traditional methods, while reducing data generation costs by 36%.

References

- AIPRM. 2024. 100+ ChatGPT Statistics 2024. <https://www.aiprm.com/chatgpt-statistics/>.
- Bai, Y.; Jones, A.; Ndousse, K.; Askell, A.; Chen, A.; Das-Sarma, N.; Drain, D.; Fort, S.; Ganguli, D.; Henighan, T.; Joseph, N.; Kadavath, S.; Kernion, J.; Conerly, T.; El-Showk, S.; Elhage, N.; Hatfield-Dodds, Z.; Hernandez, D.; Hume, T.; Johnston, S.; Kravec, S.; Lovitt, L.; Nanda, N.; Olsson, C.; Amodei, D.; Brown, T.; Clark, J.; McCandlish, S.; Olah, C.; Mann, B.; and Kaplan, J. 2022. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. <https://arxiv.org/abs/2204.05862>.
- Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In *Advances in neural information processing systems*, volume 33, 1877–1901.
- Bıyık, E.; Wang, K.; Anari, N.; and Sadigh, D. 2019. Batch Active Learning Using Determinantal Point Processes. *arXiv preprint arXiv:1906.07975*.
- Chen, L.; Li, S.; Yan, J.; Wang, H.; Gunaratna, K.; Yadav, V.; Tang, Z.; Srinivasan, V.; Zhou, T.; Huang, H.; and Jin, H. 2024. AlpaGasus: Training a Better Alpaca with Fewer Data. In *The Twelfth International Conference on Learning Representations*.
- Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; Stoica, I.; and Xing, E. P. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality. <https://lmsys.org/blog/2023-03-30-vicuna/>.
- Chowdhery, A.; Narang, S.; Devlin, J.; Bosma, M.; Mishra, G.; Roberts, A.; Barham, P.; Chung, H. W.; Sutton, C.; Gehrmann, S.; et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240): 1–113.
- Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; Webson, A.; Gu, S. S.; Dai, Z.; Suzgun, M.; Chen, X.; Chowdhery, A.; Castro-Ros, A.; Pellat, M.; Robinson, K.; Valter, D.; Narang, S.; Mishra, G.; Yu, A.; Zhao, V.; Huang, Y.; Dai, A.; Yu, H.; Petrov, S.; Chi, E. H.; Dean, J.; Devlin, J.; Roberts, A.; Zhou, D.; Le, Q. V.; and Wei, J. 2024a. Scaling Instruction-Finetuned Language Models. *Journal of Machine Learning Research*, 25(70): 1–53.
- Chung, H. W.; Hou, L.; Longpre, S.; Zoph, B.; Tay, Y.; Fedus, W.; Li, Y.; Wang, X.; Dehghani, M.; Brahma, S.; et al. 2024b. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70): 1–53.
- Clark, P.; Cowhey, I.; Etzioni, O.; Khot, T.; Sabharwal, A.; Schoenick, C.; and Tafford, O. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*.
- Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*.
- Dubois, Y.; Galambosi, B.; Liang, P.; and Hashimoto, T. B. 2024. Length-Controlled AlpacaEval: A Simple Way to Debias Automatic Evaluators. <https://arxiv.org/abs/2404.04475>.
- Dubois, Y.; Li, X.; Taori, R.; Zhang, T.; Gulrajani, I.; Ba, J.; Guestrin, C.; Liang, P.; and Hashimoto, T. 2023. AlpacaFarm: A Simulation Framework for Methods that Learn from Human Feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.
- Feng, Y.; Dohmatob, E.; Yang, P.; Charton, F.; and Kempe, J. 2024. Beyond Model Collapse: Scaling Up with Synthesized Data Requires Reinforcement. <https://arxiv.org/abs/2406.07515>.
- Geng, X.; Gudibande, A.; Liu, H.; Wallace, E.; Abbeel, P.; Levine, S.; and Song, D. 2023. Koala: A Dialogue Model for Academic Research. <https://bair.berkeley.edu/blog/2023/04/03/koala/>.
- Gerstgrasser, M.; Schaeffer, R.; Dey, A.; Rafailov, R.; Sleight, H.; Hughes, J.; Korbak, T.; Agrawal, R.; Pai, D.; Gromov, A.; Roberts, D. A.; Yang, D.; Donoho, D. L.; and Koyejo, S. 2024. Is Model Collapse Inevitable? Breaking the Curse of Recursion by Accumulating Real and Synthetic Data. *arXiv preprint arXiv:2404.01413*.
- Hendrycks, D.; Burns, C.; Basart, S.; Zou, A.; Mazeika, M.; Song, D.; and Steinhardt, J. 2021. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*.
- Kaufmann, T.; Weng, P.; Bengs, V.; and Hüllermeier, E. 2024. A Survey of Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2312.14925*.
- Köpf, A.; Kilcher, Y.; von Rütte, D.; Anagnostidis, S.; Tam, Z. R.; Stevens, K.; Barhoum, A.; Nguyen, D. M.; Stanley, O.; Nagyfi, R.; ES, S.; Suri, S.; Glushkov, D. A.; Dantluri, A. V.; Maguire, A.; Schuhmann, C.; Nguyen, H.; and Mattick, A. J. 2023. OpenAssistant Conversations Democratizing Large Language Model Alignment. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.
- Li, C.; Yao, Z.; Wu, X.; Zhang, M.; Holmes, C.; Li, C.; and He, Y. 2024. Deepspeed data efficiency: Improving deep learning model quality and training efficiency via efficient data sampling and routing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 18490–18498.
- OpenAI. 2023a. ChatGPT. <https://chatgpt.com/>.
- OpenAI. 2023b. gpt-3-5-turbo. <https://platform.openai.com/docs/models/gpt-3-5-turbo>.
- Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; Schulman, J.; Hilton, J.; Kelton, F.; Miller, L.; Simens, M.; Askell, A.; Welinder, P.; Christiano, P. F.; Leike, J.; and Lowe, R. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 27730–27744.

- Panić, M. 2021. How to Define the Right Price for a Language Dataset. <https://www.taus.net/resources/blog/how-to-define-the-right-price-for-a-language-dataset>.
- Peng, B.; Li, C.; He, P.; Galley, M.; and Gao, J. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.
- Shi, T.; Benton, A.; Malioutov, I.; and Irsoy, O. 2021. Diversity-Aware Batch Active Learning for Dependency Parsing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2616–2626. Association for Computational Linguistics.
- Shumailov, I.; Shumaylov, Z.; Zhao, Y.; Gal, Y.; Papernot, N.; and Anderson, R. 2023. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*.
- Sugiyama, A.; and Yoshinaga, N. 2019. Data augmentation using back-translation for context-aware neural machine translation. In *Proceedings of the Fourth Workshop on Discourse in Machine Translation*, 35–44.
- Taori, R.; Gulrajani, I.; Zhang, T.; Dubois, Y.; Li, X.; Guestrin, C.; Liang, P.; and Hashimoto, T. B. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*. <https://crfm.stanford.edu/2023/03/13/alpaca.html>, 3(6): 7.
- Team, G.; Mesnard, T.; Hardin, C.; Dadashi, R.; Bhupatiraju, S.; Pathak, S.; Sifre, L.; Rivière, M.; Kale, M. S.; Love, J.; et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.
- Torrey, L.; and Shavlik, J. 2010. Transfer Learning. In *Handbook of research on machine learning applications and trends: algorithms, methods, and techniques*, 242–264. IGI global.
- Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; Fernandes, J.; Fu, J.; Fu, W.; Fuller, B.; Gao, C.; Goswami, V.; Goyal, N.; Hartshorn, A.; Hosseini, S.; Hou, R.; Inan, H.; Kardaş, M.; Kerkez, V.; Khabsa, M.; Kloumann, I.; Korenev, A.; Koura, P. S.; Lachaux, M.-A.; Lavril, T.; Lee, J.; Liskovich, D.; Lu, Y.; Mao, Y.; Martinet, X.; Mihaylov, T.; Mishra, P.; Molybog, I.; Nie, Y.; Poulton, A.; Reizenstein, J.; Rungta, R.; Saladi, K.; Schelten, A.; Silva, R.; Smith, E. M.; Subramanian, R.; Tan, X. E.; Tang, B.; Taylor, R.; Williams, A.; Kuan, J. X.; Xu, P.; Yan, Z.; Zarov, I.; Zhang, Y.; Fan, A.; Kambadur, M.; Narang, S.; Rodriguez, A.; Stojnic, R.; Edunov, S.; and Scialom, T. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv:2307.09288*.
- Wang, Y.; Kordi, Y.; Mishra, S.; Liu, A.; Smith, N. A.; Khashabi, D.; and Hajishirzi, H. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 13484–13508.
- Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A. W.; Lester, B.; Du, N.; Dai, A. M.; and Le, Q. V. 2022. Finetuned Language Models are Zero-Shot Learners. In *International Conference on Learning Representations*.
- Wei, J.; and Zou, K. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*.
- Yin, C.; Qian, B.; Cao, S.; Li, X.; Wei, J.; Zheng, Q.; and Davidson, I. 2017. Deep Similarity-Based Batch Mode Active Learning with Exploration-Exploitation. In *2017 IEEE International Conference on Data Mining*, 575–584.
- Zellers, R.; Holtzman, A.; Bisk, Y.; Farhadi, A.; and Choi, Y. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? In *Annual Meeting of the Association for Computational Linguistics*.
- Zhang, S.; Dong, L.; Li, X.; Zhang, S.; Sun, X.; Wang, S.; Li, J.; Hu, R.; Zhang, T.; Wu, F.; et al. 2023. Instruction Tuning for Large Language Models: A Survey. *arXiv preprint arXiv:2308.10792*.
- Zheng, C.; Yin, F.; Zhou, H.; Meng, F.; Zhou, J.; Chang, K.-W.; Huang, M.; and Peng, N. 2024. On prompt-driven safeguarding for large language models. In *Forty-first International Conference on Machine Learning*.

Appendix

Table 3: Overview of the training hyperparameter

Hyperparameter	Value
Learning rate	$2 \cdot 10^{-5}$
Weight decay	0
Warmup ratio	0.03
Batch size	16
# of epoch	3
Max length	512

Seed instruction	Seed score
<p>"instruction": "Complete the paragraph given its prefix." "input": "During 2021, interest and investment in fintech grew significantly in many regions of the world \u20142014 its scope broadening well-beyond its early definition." "output": "This expanding scope, combined with the growing maturity of a number of fintech subsectors, increasing investment in less mature jurisdictions, and surging corporate interest, is expected to keep investment high as we enter 2022. Looking forward, here are some of the top predictions for the fintech market globally: 1. Growing number of banks will offer embedded solutions; 2. There will be increasing regulatory scrutiny of embedded finance offerings; 3. Fintechs will focus on branding themselves as data organizations; 4. ESG-focused fintechs will have a big growth trajectory; 5. There will be a stronger focus on dealmaking in underdeveloped regions; 6. Unicorn status will lose some of lustre in developed markets, remain key in emerging ones."</p>	0.592
<p>"instruction": "Generate the regular expression based on the description." "input": "Description: matching strings that start with numbers and end with space." "output": "[0-9].*\s\$"]</p>	0.618
<p>"instruction": "Sort the given list ascendingly." "input": "List: [10, 2, 5, -4, 92, 101]" "output": "[-4, 2, 5, 10, 92, 101]"</p>	0.624

Fig. 10: Low-scored seed instructions

Hyperparameter Setting

We performed instruction tuning on LLaMA-3-8B using the hyperparameters listed in Table 3. All four models we tuned: Llama-3-8B + SEDI-INSTRUCT with Llama-3-8B-Instruct and ChatGPT API and Llama-3-8B + Self-Instruct with Llama-3-8B-Instruct and ChatGPT API. These four models are trained using the same hyperparameter settings.

Seed Instruction Analysis According to Seed Score

Fig. 10 and Fig. 11 illustrate the example of seed instructions based on their seed scores, which are low and high, respectively. The low-scoring seed instructions in Fig. 10 focus on technical and mechanical tasks, such as text completion, regular expression generation, and number sorting. In contrast, the high-scoring seed instructions in Fig. 11 involve tasks that require a creative and humanistic approach, such as questioning stereotypes, discussing natural phenomena, and writing recommendation letters. The results suggest

Seed instruction	Seed score
<p>"instruction": "Describe a situation in which the given stereotype can harm you." "input": "\"All Asians are smart!\"" " "output": "Imagine you're a small child of Asian descent. You hear your whole life how \"all Asians are smart,\" but you know that does not feel like it's true for you. You have a hard time in school. Reading is difficult. Math is impossible. Everyone around you seems to pressure you because you are supposed to be \"smart.\" Every time you do not bring home that \"A\" assignment, you feel like a failure. Like you alone missed out on the gene that makes \"all Asians smart.\" As educational opportunities like scholarships and colleges close their door on you, your future looks bleak. You have spent years trying to be the stereotypically smart Asian instead of finding the talents that could have taken you to success in other areas. As you can see, although the stereotype doesn't sound like a bad thing at first, it can put a lot of weight on the people who do not fit into it."</p>	0.810
<p>"instruction": "Explain to me why a natural phenomenon happen." "input": "Thunder." "output": "Thunder is the sound made by lightning. Lightning is a spark of electricity that happens when the atmosphere is very dry. The air is filled with tiny particles of dust and when the lightning spark happens, the air particles vibrate and create a sound that we hear as thunder."</p>	0.809
<p>"instruction": "Write a recommendation letter given the student's achievement." "input": "Achievement: \n- GPA 3.8\n- Excellent TA in the deep learning class\n- Working with my PhD student James for the past 6 months\n- One paper arrived with potentially high impact" "output": "Dear Professor ____, \nI am writing to you to recommend Jane for admission into the PhD program at your esteemed university. Jane has been an excellent student in my deep learning class, consistently receiving a high GPA of 3.8. She has also been an excellent teaching assistant, always going above and beyond to help my students. Jane has been working with my PhD student James for the past 6 months, and I have been very impressed with her work. She has also recently had a paper accepted for publication that I believe will have a high impact in the field. Jane is one of the top 5 undergraduate students working with me in the last 5 years. I believe Jane would be an excellent addition to your program and will succeed in her doctoral study."</p>	0.801

Fig. 11: High-scored seed instructions

that seeds with characteristics similar to those in Fig. 11 may be more effective for generating diverse data where context comprehension and creativity are crucial.

Detailed Model Information

Table 4 summarizes each model's characteristics, including the volume and method of instruction data used during training and the additional techniques employed to enhance the model's performance.

The Llama-3-8B + Self-Instruct and Llama-3-8B + SEDI-INSTRUCT models were trained with 30,164 instructions collected through their respective data generation frameworks without additional training techniques. Falcon-7B-Instruct stands out with 250 million tokens of synthesized data combined with the manual collection but does not employ RLHF or SFT. On the other hand, Gemma-7B-

Table 4: Detailed model information

	Dataset scale	Instruction collection method	Additional training techniques
Llama-3-8B -Instruct	Over 10M instructions	Manual collection	RLHF, SFT
Llama-3-8B + Self-Instruct	30,164 instructions	Self-Instruct	None
Llama-3-8B + SEDI-INSTRUCT	30,164 instructions	SEDI-INSTRUCT	None
Falcon-7B -Instruct	250M tokens	Synthesized data + manual collection	None
Gemma-7B -Instruct	Not publicly disclosed	Synthesized data + manual collection	RLHF, SFT

Instruct’s training data volume remains undisclosed, although it similarly uses synthesized data and manual collection and is further refined using RLHF and SFT.

Detailed benchmark

Table 5 presents detailed information on benchmarks that include subgroup evaluations, as referenced in the benchmark evaluation results of Table 1. Similar to the results in Table 5, the Llama-3-8B + SEDI-INSTRUCT model does not fall behind in overall performance. This is noteworthy considering that the data collection costs for this model were significantly lower compared to other models.

Used Prompts

Fig. 12 shows all the prompts we used in the paper. An instruction generation prompt is used to generate the instructions. Seed instructions are appended to this prompt by the first three tasks from a List of 20 tasks and then fed to the instruction-generating model. Llama-2 default prompt is designed to mitigate the risk of generating harmful content by guiding the model to refuse to answer unsafe queries. For the competitive evaluation, we use the competitive evaluation prompt. Each answer of the models is filled in the `{answer_1}` or `{answer_2}`.

Table 5: Detailed benchmark results of MMLU and ARC

Groups	Llama-3-8B-Instruct	Llama-3-8B + Self-Instruct	Llama-3-8B + SEDI-INSTRUCT	Falcon-7B-Instruct	Gemma-7B-Instruct
MMLU	65.7	56.5	56.6	25.1	50.2
STEM	56.5	47.5	49.3	23.1	43.0
Humanities	60.4	51.0	49.8	24.9	44.7
Social sciences	76.6	66.6	66.7	24.2	58.3
Other	72.2	64.1	64.1	28.2	44.7
ARC	72.2	67.7	69.3	62.0	66.3
ARC challenge	53.2	47.7	49.8	40.1	47.3
ARC easy	81.2	77.5	79.0	72.7	75.6

Instruction generation prompt	<p>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:</p> <ol style="list-style-type: none"> 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 instructions. 3. The type of instructions should be diverse. The list should include diverse types of tasks like open-ended generation, classification, editing, etc. 4. 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. 5. The instructions should be in English. 6. The instructions should be 1 to 2 sentences long. Either an imperative sentence or a question is permitted. 7. 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. 8. Not all instructions require input. For example, when an instruction asks about some general information, "what is the highest peak in the world", it is not necessary to provide a specific context. In this case, we simply put "<noinput>" in the input field. 9. The output should be an appropriate response to the instruction and the input. Make sure the output is less than 100 words. <p>List of 20 tasks:</p>
Llama-2 default prompt	<p>You are a helpful, respectful and honest assistant. Always answer as helpfully as possible, while being safe. Your answers should not include any harmful, unethical, racist, sexist, toxic, dangerous, or illegal content. Please ensure that your responses are socially unbiased and positive in nature.</p> <p>If a question does not make any sense, or is not factually coherent, explain why instead of answering something not correct. If you don't know the answer to a question, please don't share false information.</p>
Competitive evaluation prompt	<p>System Prompt: You are a helpful and precise assistant for checking the quality of the answer.</p> <p>User Prompt: [Question] [The Start of Assistant 1's Answer] {answer_1} [The End of Assistant 1's Answer] [The Start of Assistant 2's Answer] {answer_2} [The End of Assistant 2's Answer]</p> <p>We would like to request your feedback on the performance of two AI assistants in response to the user question displayed above.\nPlease rate the helpfulness, relevance, accuracy, level of details of their responses. Each assistant receives an overall score on a scale of 1 to 10, where a higher score indicates better overall performance.\nPlease first output a single line containing only two values indicating the scores for Assistant 1 and 2, respectively. The two scores are separated by a space. In the subsequent line, please provide a comprehensive explanation of your evaluation, avoiding any potential bias and ensuring that the order in which the responses were presented does not affect your judgment."</p>

Fig. 12: All of the used prompts

