Rethinking the Instruction Quality: LIFT is What You Need

Anonymous ACL submission

Abstract

Instruction tuning, a specialized technique to enhance large language model (LLM) performance via instruction datasets, relies heavily on the quality of the employed data. Existing quality improvement methods alter instruction data through dataset expansion or curation. However, the expansion method introduces the risk of data deficiency and redundancy, potentially compromising the correctness and accuracy of the LLM's knowledge, while the curation approach confines the LLM's potential to the original dataset. Our aim is to surpass the original data quality without confronting these shortcomings. To achieve this, we propose **LIFT** (**L**LM **I**nstruction **F**usion **T**ransfer), a novel and versatile paradigm designed to elevate the instruction quality to new heights. LIFT strategically broadens data distribution to encompass more high-quality subspaces and eliminates redundancy, concentrating on high-quality segments across overall data subspaces. Experimental results demonstrate that, even with a limited quantity of high-quality instruction data selected by our paradigm, LLMs not only consistently uphold robust performance across natural language understanding and code generation tasks but also surpass many state-of-the-art results, highlighting the significant improvement in instruction quality achieved by our paradigm.

1 Introduction

In recent years, Large Language Models (LLMs) have gained prominence for their remarkable effectiveness in natural language comprehension tasks (OpenAI, 2023; Yang et al., 2023; Qi et al., 2023). High-quality pretrained LLMs are readily available, facilitating their customization for versatile applications (Wei et al., 2021; Huang et al., 2023). One popular fine-tuning approach, known as instruction tuning (Wei et al., 2022; Ouyang et al., 2022), involves fine-tuning pre-trained LLMs using datasets accompanied by natural language instructions. Its relative simplicity and affordability make it a preferred method for improving LLMs' performance on specific tasks.

The quality of current instruction datasets, whether manually crafted or generated by LLMs, often falls short of the desired standard. Human-crafted datasets depend on human annotators to generate a substantial corpus with human instructions, resulting in a lack of detailed context and explanation within the instruction dataset. Additionally, these datasets may contain vague or subjective descriptions. On the other hand, LLM-generated datasets utilize advanced LLMs to generate or complete instructions and responses but lack supervision regarding the diversity and quality of the generated data.

The concern surrounding the quality of instruction datasets has prompted researchers to explore methods aimed at enhancing their overall quality. Current approaches to instruction quality enhancement can be broadly categorized into two groups: data expansion and data curation. Data expansion methods involve leveraging advanced LLMs with a suitable prompt template to generate new instructions and corresponding answers based on the original dataset (Xu et al., 2023; Luo et al., 2023; Taori et al., 2023). On the other hand, data curation methods entail the meticulous selection of high-quality data from the original dataset, employing specific quality evaluation criteria (Zhou et al., 2023; Du et al., 2023).

However, both existing methods exhibit limitations that hinder their ability to further enhance performance. Expansion methods introduce redundancy into the dataset (Xu et al., 2023; Luo et al., 2023) as the newly generated instructions typically derive from the original ones. While the effectiveness of curation methods heavily relies on the quality of the original dataset, limiting the quality of the curated dataset (Du et al., 2023; Li et al., 2023a). These limitations necessitate a reliance on specific expansion or curation strategies to achieve superior performance on certain benchmarks, at the expense of losing the ability to generalize the approach.

In this paper, we delve into the distribution of instruction quality to address the mentioned issues. We posit that both current methods essentially function as data distribution transfers: expansion enables the distribution to cover a broader range of data subspaces, typically characterized by higher quality, while curation concentrates the distribution on a higher-quality subset of the original dataset. Building on this perspective, we propose a novel paradigm for improving LLM instruction quality, termed **LIFT** (**L**LM **I**nstruction **F**usion **T**ransfer). LIFT is designed to amalgamate the advantages of data expansion and curation, mitigating their shortcomings to generate a diverse and high-quality dataset while significantly reducing quantity. Our paradigm consists of two phases. Firstly, we employ "Dataset Distribution Expansion", broadening the data distribution to cover more high-quality subspaces. Then, we utilize "Dataset Variety and Quality Curation" to eliminate redundancy and densify the data distribution, focusing on the high-quality segments of overall data subspaces. The data distribution transfer patterns of three methods are described in Fig.1.

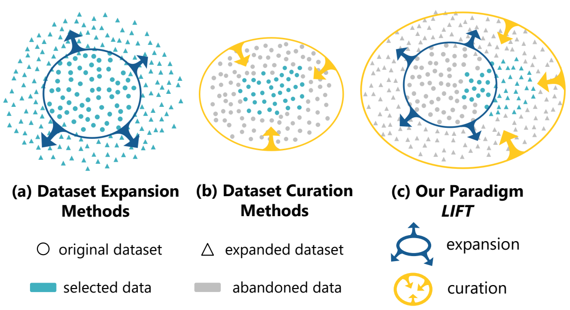


Figure 1: Different Instruction Data Distribution Transfer Patterns.

To validate the effectiveness of LIFT, we employ the finally curated instructions for fine-tuning open-source LLMs. Through extensive experiments evaluating the performance of these fine-tuned LLMs in both natural language understanding (NLU) tasks and code generation tasks, the results consistently demonstrate that LLMs achieve robust SOTA or nearly-SOTA performance even with a limited quantity of high-quality instruction data. Furthermore, they even outperform models trained on larger datasets on certain benchmarks.

To summarize, our main contribution are:

- We propose a highly effective and versatile paradigm, LIFT, which challenges the conventional single-mode enhancement for instruction datasets. LIFT rethinks data quality by focusing on data distribution transfer. It aims to elevate the quality of the instruction dataset to new heights, overcoming redundancy and quality limitations present in current methods.

- Throughout the expansion and curation phases of the paradigm, we prioritize both variety and quality as essential goals for quality enhancement. Unlike existing works that concentrate on only one stage, we posit that considering these characteristics at both stages is crucial for incorporating more high-quality data.

- Our extensive experiments demonstrate that with a significantly reduced quantity of high-quality instructions selected by our paradigm, LLMs consistently achieve SOTA performance on many NLU tasks and code generation tasks. This provides valuable insights, suggesting that a selective approach based on the principles of data distribution transfer is not only more effective but also cost-effective compared to the indiscriminate feeding of large volumes of data.

2 Related Works

The current methods for enhancing instruction quality can be broadly categorized into two types based on how data is manipulated: dataset expansion and curation.

2.1 Instruction Dataset Expansion

The original instruction dataset often consists of concise and straightforward prompts, yielding simplistic responses with limited semantic information. To address this limitation, researchers have proposed using LLMs to expand these original instructions to introduce more high-quality data. Alpaca (Taori et al., 2023) suggested adopting the self-instruct method, utilizing ChatGPT to generate data for fine-tuning. Vicuna (Chiang et al., 2023) employed data collected from ShareGPT.com, a platform where users share their conversations with ChatGPT, for fine-tuning their models. WizardLM (Xu et al., 2023) and WizardCoder (Luo et al.,

2023) introduced the *Evol-Instruct* method, involving the evolution of existing instruction data to generate more diverse and complex data.

2.2 Instruction Dataset Curation

One challenge in the instruction tuning process arises from the observation that fine-tuning with larger expanded instruction datasets does not always guarantee better results, yet demanding more computational resources. To address this, some researchers have focused on filtering out low-quality data during the fine-tuning stage. LIMA (Zhou et al., 2023) demonstrates that fine-tuning a robust pre-trained language model on 1000 high-quality, human-curated examples can yield remarkable and competitive results. Instruction Mining (Cao et al., 2023) introduces a linear rule for selecting high-quality instruction data, eliminating the need for human annotation. Du et al. (2023) present a model-oriented data selection (MoDS) approach, which selects instruction data based on new criteria considering three aspects: quality, coverage, and necessity. Li et al. (2023a) introduce a self-guided methodology for LLMs to autonomously discern and select cherry-picked samples from vast open-source datasets, effectively minimizing manual curation and potential costs.

3 LLM Instruction Fusion Transfer

3.1 Data Distribution Transfer

Current methods for enhancing instruction quality, either through data expansion or curation, do enhance the original dataset to some extent. However, the effectiveness of these methods is constrained by inherent limitations. To scrutinize these limitations and explore innovative approaches to break from conventional enhancement modes, we propose a novel perspective for rethinking instruction data quality: **data distribution transfer**.

3.1.1 Rethinking Existing Methods from Data Distribution Perspective

Our hypothesis is that, during the process of enhancing instruction quality, there is a transfer of data distribution from the original dataset to the final enhanced dataset. This transfer increases the quantity or proportion of high-quality data. In data expansion, generating high-quality instructions based on the original ones effectively extends the coverage of the high-quality data subspace within the original data distribution, thereby increasing the quantity of high-quality data in the final distribution. On the other hand, in data curation, by using carefully designed quality evaluation metrics, low-quality components are removed from the final distribution, directing the distribution to concentrate on high-quality data and increasing its proportion in the final distribution.

From this perspective, we can delve into the origin of limitations in these processes. For expansion, the areas around the original instructions may contain similar ones, leading to redundancy in the final distribution. Moreover, low-quality instructions and those derived from them persist in the final distribution, maintaining a proportion similar to the original dataset. On the other hand, curation selects a portion of high-quality instructions from the original dataset, resulting in a decrease in the total number of high-quality instructions. If the original dataset has a limited number of high-quality instructions, the quality of the curated dataset will significantly decrease.

3.1.2 Fusing Expansion and Curation

Analyzing the data distribution transfer patterns of expansion and curation, we propose that their integration can effectively address their individual limitations. Data expansion broadens subspaces, enabling the curation method to explore beyond the original distribution. Conversely, data curation assists in identifying duplicates and low-quality items from the expansion, contributing to a more concentrated and refined distribution.

Building on these insights, we introduce a novel paradigm called **LIFT** (LLM Instruction Fusion Transfer). Comprising two phases, this paradigm orchestrates the distribution of instruction data as follows: in the "Dataset Distribution Expansion" phase, we broaden the data distribution to encompass more diverse and high-quality subspaces, acknowledging the presence of duplications at this stage. Subsequently, in the "Dataset Variety and Quality Curation" phase, we systematically eliminate redundancy and low-quality elements, creating a densified distribution for the final curated dataset. In contrast to the existing works, which require intricate strategies to focus on the original dataset, our paradigm offers a versatile perspective to surpass the limitations of the original dataset's quality. These two phases are intricately connected, ensuring a smooth transfer of data from the original dataset to the final curated dataset.

3.2 Paradigm LIFT

As described in Fig.2, our paradigm LIFT follows a two-stage structure. In both stages, we value the diversity and quality as the crucial criterion and we believe the "Dataset Distribution Expansion" and "Dataset Variety and Quality Curation" equally contribute to the quality enhancement.

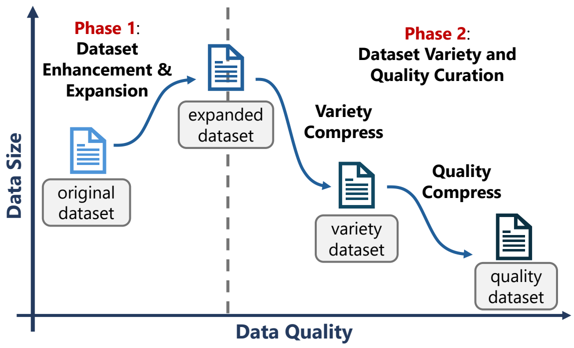


Figure 2: Instruction Dataset Curation Paradigm **LIFT**

3.2.1 Dataset Distribution Expansion

The goal of dataset distribution expansion is to encompass a more diverse and high-quality range of data within the distribution, while ensuring a certain distance from the original instructions. To achieve this, it is crucial to employ carefully designed instruction-generation prompts. Drawing inspiration from the instruction rewrite method proposed by Xu et al. (2023) and Luo et al. (2023), our approach focuses on generating diverse and intricate instructions. We guide GPT to act as a prompt re-writer, generating challenging instructions based on specified generation rules. For further details, refer to Appx.A. We iterate this process for k rounds, merging the expanded datasets with the original dataset to create the final expanded dataset. We also compare the quality of the original and expanded dataset in Appx. B.

3.2.2 Dataset Variety and Quality Curation

An effective curation method ought to eliminate duplicated or low-quality instructions from the original dataset, while preserving representative and high-quality ones. To meet this criterion, curation should be approached with meticulous attention to both variety and quality.

Variety Curation. Current variety curation typically involves clustering methods such as k-means or spectral clustering (Du et al., 2023; Wei et al., 2023) . We argue that this approach may lack generalizability and be less effective when dealing with new datasets. This is because these methods require prior knowledge of the number of clusters, and choosing cluster numbers that are either too large or too small may reduce their effectiveness in selecting representatives.

Our variety curation method take another route, as depicted in Fig.3. Initially, GPT generates embeddings with 1536 dimensions for each item, we aim to reduce the embedding dimension and devise a method to represent data differentiation. We achieve this by calculating the covariance matrix of the given features and performing eigenvalue decomposition on the covariance matrix to obtain eigenvalues and eigenvectors. We then choose the top k eigenvectors corresponding to the largest k eigenvalues, where k is the target reduced dimension.

This procedure aids us in analyzing the distribution of data in orthogonal space. To simultaneously maintain data diversity while utilizing fewer data cases, we perform balanced sampling of the data in orthogonal space. Starting from each eigenvector and guided by the corresponding eigenvalues. We sample more data from significant eigenvectors than those less significant ones. This approach ensures that while reducing data cases, the distribution of data embeddings remains rational, thereby preserving data diversity.

Quality Curation. Following variety curation is the quality curation phase, where we discern high-quality instruction data. Rating instruction quality is challenging due to the lack of official quantitative metrics. Employing professional annotators for scoring is impractical due to dataset size and costs. Therefore, we use GPT as an instruction scorer, generating GPT quality scores across four dimensions: accuracy, explanation, clarity, and difficulty, with proportions based on their contributions to overall quality. The guiding template for GPT scores is in Appx.C.

To address the problem that GPT consistently assigns high scores to all instructions, we implement the following steps for more differentiated scores: first we instruct GPT to provide a comprehensive rationale along with a score yields more reasonable results. Mandating GPT to articulate its reasoning offers an additional self-checking opportunity. Secondly, we present manually scored examples as guidelines. Offering three examples with scores representing poor, average, and high quality helps GPT recognize low-quality data and understand

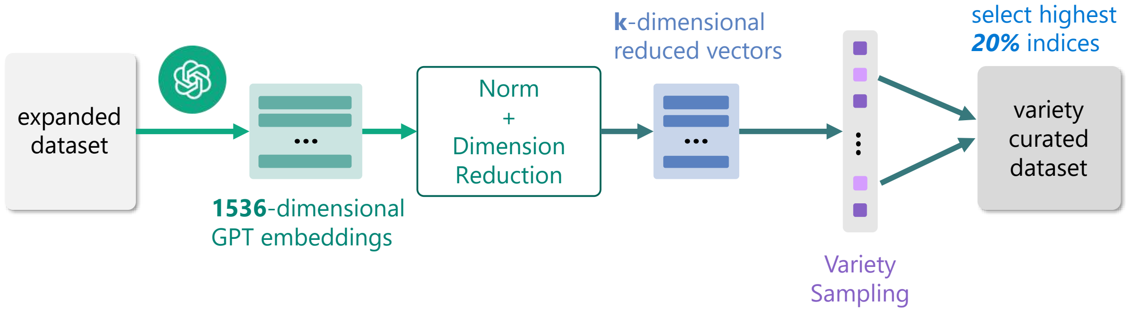


Figure 3: Variety Curation with Dimension Reduction and Row Variances

how to appropriately score it.

We also incorporate A positively correlated mapping function derives a lengthwise semantic score based on instruction data's length. Combining GPT quality score and lengthwise semantic score produces the final quality score. High-quality scores compose the final quality-curated dataset, as illustrated in Fig.4. Appx.D presents total quality score distributions.

4 Experiments

To validate the effectiveness of our paradigm, we apply our method to two extensively studied tasks: Natural Language Understanding (NLU) tasks and Code Generation tasks, where we conduct comprehensive experiments to evaluate the performance of our paradigm.

4.1 Experiments Setup

4.1.1 Basic Foundation Models and Base Datasets

We adopt distinct foundation models and base dataset configurations for the two tasks under consideration. In NLU tasks, we employ foundation models Mistral 7B (Jiang et al., 2023) and LLaMA2 7B (Touvron et al., 2023b), known for their exceptional performance relative to other 7B models and ability to surpass larger models in specific benchmarks. Our base dataset for NLU tasks is the Open Platypus dataset (Lee et al., 2023), comprising 25k curated examples focused on enhancing LLMs' STEM and logic knowledge. While in the realm of code generation tasks, we harness the capabilities of StarCoder 15B (Li et al., 2023b) and CodeLLaMA (Rozière et al., 2023), both widely-utilized code LLMs trained on a diverse array of programming-related sources. Our base dataset, Code Alpaca (Chaudhary, 2023), consists of 20k instruction-following code instances for fine-tuning Code LLMs.

We employ both GPT-3.5 and GPT-4 as assistants for expansion and quality curation due to our paradigm's flexibility in selecting an assistant. For comprehensive implementation details pertaining to instruction tuning in both tasks, please refer to Appx. E.

4.2 Benchmarks and Metrics

We have chosen six widely-recognized benchmarks spanning both tasks. In the domain of NLU tasks, we have incorporated HellaSwag, ARC Challenge, TruthfulQA, and MMLU. For code generation tasks, our selection encompasses HumanEval and MBPP. Detailed information about these benchmarks is provided in Appx. F.1.

For NLU tasks, we adopt accuracy as the metric, aligning with the methodology embraced by other researchers. This metric is calculated as the number of correct questions divided by the total number of questions.

In code generation tasks, our metric of choice is pass@k, defined in the same manner as by Chen et al. (2021). The formula for calculating pass@k is presented as:

$$pass@k := \mathbb{E}_{problems}[1 - \frac{C(n-c,k)}{C(n,k)}] \quad (416)$$

Here, n represents the number of generated answers for each question, and c denotes the number of correct answers for each question. In our experiments, we specifically choose pass@1 as the designated metric.

4.3 Experiment Results

To validate the effectiveness of our paradigm, we conduct comparisons between models fine-tuned

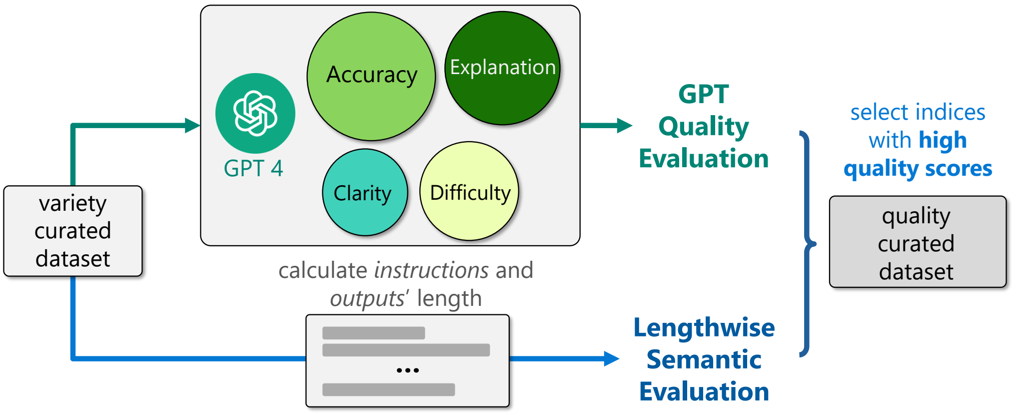


Figure 4: Quality Curation with GPT Quality Evaluation and Lengthwise Semantic Evaluation

on LIFT's final curated dataset and other SOTA pre-trained LLMs as well as instruction-tuned LLMs across both tasks. The details of the selected models for comparison are provided in Appx. F.2.

4.3.1 NLU Tasks

Tab.1 presents the NLU task comparison results. Notably, our final 7B instruction-tuned Mistral model consistently outperforms other 7B models on all benchmarks. Comparing with 13B models, our 7B model even outperforms in all benchmarks except TruthfulQA. With only 7 billion parameters and 15k instructions, significantly fewer than other instruction-tuned models, our model achieves the highest average benchmark score at 0.656.

4.3.2 Code Generation Tasks

As illustrated in Table 2, our paradigm's fine-tuned model consistently outperforms most models in code generation tasks. Although our fine-tuned CodeLLaMA model trails the current state-of-the-art 15B model, WizardCoder, by approximately 2% on both benchmarks, it is noteworthy that our paradigm utilizes only about **one-eighth** of the instruction data employed by WizardCoder. Considering the disparity in the size of the instruction dataset, our paradigm demonstrates robust performance, highlighting its capability to achieve performance levels close to the state-of-the-art with a significantly smaller amount of data.

We also compared our paradigm's final curated dataset with a randomly selected dataset of the same size in both tasks. The results demonstrate that merely reducing the dataset quantity, without accounting for the diversity and quality in the perspective of data distribution, does not lead to performance improvement. These experiments affirm our paradigm's versatile effectiveness in NLU and code generation tasks. The paradigm excels in generating diverse, high-quality data, leveraging it in the instruction-tuning process to achieve SOTA or near-SOTA performance.

4.4 Paradigm Ablation Experiments Results

Our paradigm ablation experiment begins with the original base dataset serving as the input for LIFT. Subsequently, we generate the expanded dataset, variety-curated dataset, and the quality-curated dataset. These datasets are then utilized for fine-tuning the basic foundation models. We assess the benchmark performance of these models to validate the effectiveness of each component of our paradigm. Besides the paradigm ablation experiments discussed here, we also provide the component ablation results in Appx. G.

4.4.1 NLU Tasks

Tab.3 presents our paradigm experiment results on four NLU benchmarks. For data expansion, we iteratively perform the expansion step 3 times, resulting in a 100k size instruction dataset.

The table results affirm our paradigm's effectiveness in NLU. Despite a reduction in size by 10k instances compared to the original dataset, our final curated dataset maintains robust performance, showing improvements ranging from nearly 2% to 4% on each benchmark. Furthermore, we observe a consistent improvement in model performance on both benchmarks after each step of the paradigm. This implies that the instruction's quality is steadily increasing at each stage.

It's crucial to note that the Open Platypus (Lee et al., 2023) dataset for NLU tasks is already carefully curated. The results for this dataset under-

Table 1: LLMs Performance Comparison in NLU Tasks

Model	Data Size	HellaSwag	ARC	TruthfulQA	MMLU
LLaMA-7B		0.778	0.509	0.343	0.357
LLaMA-13B		0.809	0.561	0.395	0.476
LLaMA2-7B	*Pretrained*	0.771	0.432	0.333	0.444
LLaMA2-13B		0.807	0.488	0.419	0.556
Mistral-7B		0.823	0.602	0.426	0.627
Vicuna-7B	70k conv.	0.775	0.537	0.489	0.456
Vicuna-13B	70k conv.	0.801	0.530	0.518	0.513
WizardLM-7B	70k inst.	0.771	0.516	0.447	0.427
WizardLM-13B	70k inst.	0.777	0.572	0.505	0.523
Platypus2-13B	25k inst.	0.826	0.613	0.449	0.567
Camel-Platypus2-13B	25k inst.	0.836	0.608	0.496	0.565
Stable-Platypus2-13B	25k inst.	0.822	0.627	**0.525**	0.583
LLaMA2-7B Fine-tuned	15k inst.	0.786	0.545	0.347	0.574
Mistral-7B Fine-tuned	15k inst.	0.820	0.607	0.438	0.625
LLaMA2-7B w/ LIFT(GPT-3.5)	15k inst.	0.794	0.566	0.443	0.607
Mistral-7B w/ LIFT(GPT-3.5)	15k inst.	0.839	0.613	0.485	0.644
LLaMA2-7B w/ LIFT(GPT-4)	**15k** inst.	0.802	0.576	0.456	0.612
Mistral-7B w/ LIFT(GPT-4)	**15k** inst.	**0.843**	**0.644**	0.491	**0.645**

Table 2: LLMs Performance Comparison in Code Generation Tasks (pass@1)

Model	Data Size	HumanEval	MBPP
CodeT5+		0.309	-
CodeLLaMA	-*	0.360	0.470
StarCoder		0.336	0.436
InstructCodeT5+	20k	0.350	-
WizardCoder	78k	**0.573**	**0.518**
StarCoder Fine-tuned	10k	0.381	0.431
CodeLLaMA Fine-tuned	10k	0.393	0.465
StarCoder w/ LIFT(GPT-3.5)	10k	0.546	0.491
CodeLLaMA w/ LIFT(GPT-3.5)	10k	0.549	0.493
StarCoder w/ LIFT(GPT-4)	**10k**	0.550	0.495
CodeLLaMA w/ LIFT(GPT-4)	**10k**	0.551	0.498

* *Pretrained models*

score that our paradigm is effective not only for LLM-generated datasets but also in elevating the quality of already high-quality datasets, contributing to improved fine-tuned model performance while reducing the dataset size.

4.4.2 Code Generation Tasks

Tab.4 provides an overview of the paradigm experiments conducted on code generation tasks. For data expansion, we repeatedly perform the expansion step 2 times, resulting in a 60k size instruction dataset.

The table illustrates our paradigm leads to a significant enhancement in the performance of the fine-tuned model across both benchmarks. Notably, our final curated dataset, although roughly half the size of the original dataset, outperforms the latter by nearly 15% on the HumanEval and 3% on the MBPP. The observed pattern of performance improvement in NLU tasks also extends to code generation tasks, where each step contributes to enhancing data quality. This further underscores that each component of our paradigm plays a vital role in elevating the overall instruction quality.

5 Discussions

5.1 Composition of The Final Curated Dataset

We take a step further to analyze the composition of the final curated dataset, unraveling the origins of diverse and high-quality instruction items. Fig.5 presents the source proportions of the final curated dataset for NLU and code generation tasks, yielding several noteworthy conclusions.

For LLM-generated instruction datasets like Code Alpaca, only a small proportion of the final dataset emanates from the original dataset (Fig.5b). The majority of high-quality data is derived from our paradigm's first step—the expanded dataset. This emphasizes our paradigm's significant role in generating and covering a diverse and high-quality dataset, especially for datasets without meticulous curation.

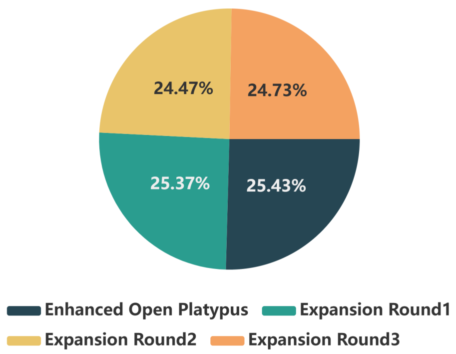
In contrast, for a curated and high-quality instruction dataset like Open Platypus, the portion of the original dataset in the final dataset increases (Fig.5a). The proportions of the final curated

Table 3: Paradigm Ablation Experiment Results in NLU Tasks (Foundation Model: Mistral)

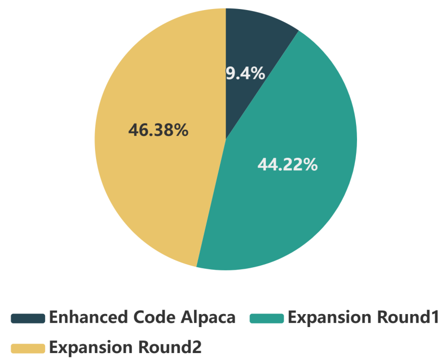
Dataset Type	Dataset Size	HellaSwag	ARC Challenge	TruthfulQA	MMLU
Base Platypus Dataset	25k	0.828	0.615	0.445	0.626
Data Expansion	100k	0.833	0.624	0.447	0.631
Variety Compress	20k	0.840	0.633	0.456	0.642
Quality Compress	**15k**	**0.843**	**0.644**	**0.491**	**0.645**

Table 4: Paradigm Ablation Experiment Results (Pass@1) in Code Generation Tasks (Foundation Model: Code LLaMA)

Dataset Type	Size	HumanEval	MBPP
Base Dataset	20k	0.410	0.467
Data Expansion	60k	0.535	0.488
Variety Curation	12k	0.542	0.490
Quality Curation	**10k**	**0.551**	**0.498**



(a) Source of the final curated 15k dataset in NLU tasks



(b) Source of the final curated 10k dataset in code generation tasks

Figure 5: Composition of The Final Curated Dataset

dataset in NLU tasks reveal an almost equal distribution among the four sub-datasets, demonstrating that even for an initially high-quality dataset, our paradigm also excels in generating and selecting numerous high-quality data points based on the original dataset.

These conclusions affirm that the bulk of the final dataset primarily comprises data from the expanded dataset. While proportions of original dataset data contributing to the final dataset may vary based on the original dataset's quality, our paradigm consistently showcases its ability to extract high-quality segments from the original dataset and augment them with diverse and high-quality data subspaces.

5.2 Limitations and Future Works

The main limitation of our paradigm lies in the subjectivity of our quality evaluation process, as it heavily relies on GPT quality evaluation. Despite we carefully design some criteria, additional statistical analysis beyond the length factor could also enhance the precision of high-quality data selection. Future work will involve integrating more comprehensive metrics to provide a more nuanced assessment of instruction quality.

6 Conclusions

This paper presents a novel paradigm, LIFT, that departs from the traditional single-mode quality enhancement approach for instruction datasets, opting for a fresh perspective on data quality through data distribution transfer. By combining the strengths of data expansion and curation while mitigating their limitations, LIFT significantly enhances elevate the quality of the instruction dataset to new heights. Extensive experimental results demonstrate that our fine-tuned models consistently attain either SOTA or nearly SOTA performance in both NLU and code generation. These experiments underscore the paradigm's versatile effectiveness, showcasing its capacity to encompass and select diverse and high-quality data. The integration of the curated data into the instruction-tuning process empowers LLMs to achieve superior performance across various tasks and benchmarks.

References

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen

Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: When data mining meets large language model finetuning.

Sahil Chaudhary. 2023. Code alpaca: An instruction-following llama model for code generation. `https://github.com/sahil280114/codealpaca`.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code.

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.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. 2023. Mods: Model-oriented data selection for instruction tuning. *ArXiv*, abs/2311.15653.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Xiaodong Song, and Jacob Steinhardt. 2020. Measuring massive multitask language understanding. *ArXiv*, abs/2009.03300.

J. Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *ArXiv*, abs/2106.09685.

Yufan Huang, Mengnan Qi, Yongqiang Yao, Maoquan Wang, Bin Gu, Colin Clement, and Neel Sundaresan. 2023. Program translation via code distillation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10903–10914, Singapore. Association for Computational Linguistics.

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

Ariel N. Lee, Cole J. Hunter, and Nataniel Ruiz. 2023. Platypus: Quick, cheap, and powerful refinement of llms. *ArXiv*, abs/2308.07317.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2023a. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. *ArXiv*, abs/2308.12032.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023b. Starcoder: may the source be with you!

Stephanie Lin, Jacob Hilton, and Owain Evans. 2022. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland. Association for Computational Linguistics.

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, MING GONG, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, and Shujie LIU. 2021. Codexglue: A machine learning benchmark dataset for code understanding and generation. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

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 *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Mengnan Qi, Yufan Huang, Maoquan Wang, Yongqiang Yao, Zihan Liu, Bin Gu, Colin Clement, and Neel Sundaresan. 2023. SUT: Active defects probing for transcompiler models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 14024–14034, Singapore. Association for Computational Linguistics.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. Zero: Memory optimizations toward training trillion parameter models. *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, I. Evtimov, Joanna Bitton, Manish P Bhatt, Cristian Cantón Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre D'efossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code llama: Open foundation models for code. *ArXiv*, abs/2308.12950.

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`.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D. Q. Bui, Junnan Li, and Steven C. H. Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. *ArXiv*, abs/2305.07922.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners. *ArXiv*, abs/2109.01652.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners.

Lai Wei, Zihao Jiang, Weiran Huang, and Lichao Sun. 2023. Instructiongpt-4: A 200-instruction paradigm for fine-tuning minigpt-4. *ArXiv*, abs/2308.12067.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *ArXiv*, abs/2304.12244.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Bing Yin, and Xia Hu. 2023. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *CoRR*, abs/2304.13712.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, L. Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *ArXiv*, abs/2305.11206.

A GPT Prompt Templates For Data Expansion

```
SYSTEM MESSAGE:
I want you act as a professional prompt
refinement specialist.
USER PROMPTS:
Your task is to transform a provided
prompt into a more intricate version
utilizing a structured data format,
introducing complexity to challenge
well-known AI systems. However, ensure
that the revised prompt remains
reasonable, comprehensible, and capable
of being understood and addressed by
humans.
You can enhance the complexity through
various methods, including but not
limited to:
(1) The depth and breadth of the inquiry
can be increased.
(2) Replace general concepts with more
specific concepts.
(3) If original problem can be solved
with just a few simple thinking processes,
you can rewrite it to explicitly request
multiple-step reasoning.

#Instruction#
{Instruction}
#Input#
{Input}
```

B Assessment of Original and Expanded Dateset Quality

Assessing quality distribution in instruction datasets is challenging, with no widely accepted tool for this quantification. Our method is comparative evaluation: randomly selecting a data case from the original dataset, we pair it with one of its augmented counterparts and shuffle the order. Volunteers evaluate the pair across three dimensions: clarity, complexity, and explanation, selecting the better one for each dimension. We then calculate the proportions of volunteer selections for both original and augmented datasets. See the Fig. 6 below for specific metrics.

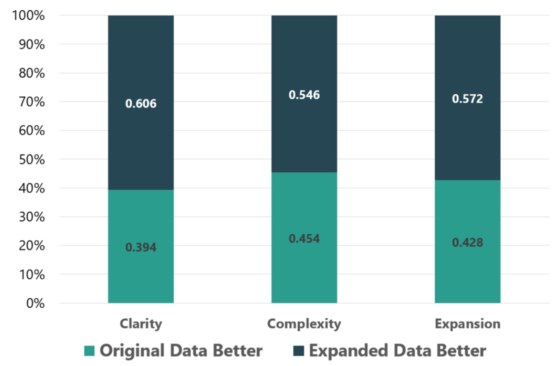


Figure 6: Quality Assessments of the Original and Expanded datasets

C GPT Quality Score Template

```
SYSTEM MESSAGE:
We would like to request your feedback
on the performance of an AI assistant.
The assistant provides outputs for
instruction and input (if any).
USER PROMPTS:
Please score the response to the
instruction and input according to the
following criteria.
The maximum score is 100 points, and it
consists of 4 parts:
1. Clarity (15 points): Assign a score
based on how effectively the instruction
conveys the problem. High-quality, clear
questions score higher.
2. Difficulty (25 points): Rate the
complexity of the instruction's problem.
Higher difficulty should receive a higher
score.
3. Explanations (25 points): Assess if the
response includes detailed explanations
alongside any code provided. The more
comprehensive the explanation, the higher
the score.
4. Accuracy (35 points): Score the
response based on the accuracy and
correctness of the solution to the
instruction's problem. Higher accuracy
should receive a higher score.
Here's some examples and socres you can
follow:
### Example 1:
### Instruction: {EXAMPLE INSTRUCTION 1}
### Response: {EXAMPLE OUTPUT 1}
### Score for Example 1: {SCORE 1}
### Example 2:
### Instruction: {EXAMPLE INSTRUCTION 2}
### Input: {EXAMPLE INPUT 2}
### Response: {EXAMPLE OUTPUT 2}
### Score for Example 2: {SCORE 2}
```

```
### Example 3:
### Instruction: {EXAMPLE INSTRUCTION 3}
### Response: {EXAMPLE OUTPUT 3}
### Score for Example 3: {SCORE 3}

Please score the upcoming Instruction,
Input and Response based on these examples
across four dimensions, and then add the
four scores together to get the total
score. Try to avoid getting a full score
as much as possible.
Please first output a single line
containing the total score number only.
In the subsequent line, please provide
a comprehensive explanation of your
evaluation, avoiding any potential bias.
### Instruction:
{INSTRUCTION}
### Input:
{INPUT}
### Response:
{OUTPUT}
```

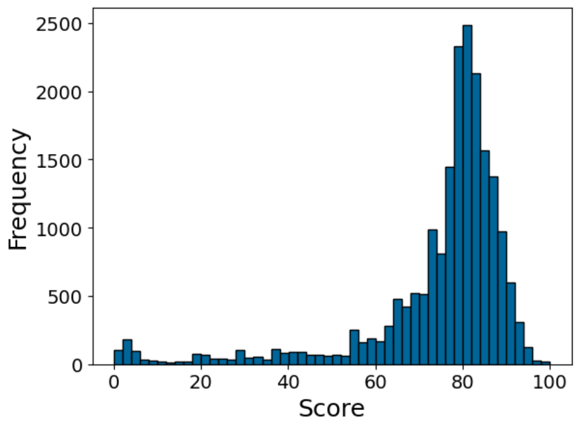
D Quality Score Distribution

We have gathered the quality scores for the variety curated dataset following our paradigm, both in NLU and code generation tasks. The score distributions are depicted in Fig.7. Notably, the quality scores exhibit an approximately normal distribution within the score interval of 60 to 100 for both tasks. This observation validates the effectiveness of our scoring strategies in discerning low-quality data. It should be noted that the minor bumps near 0 stem from connection errors or OpenAI API calling ratio constraints, resulting in GPT scores of 0 for certain instructions.

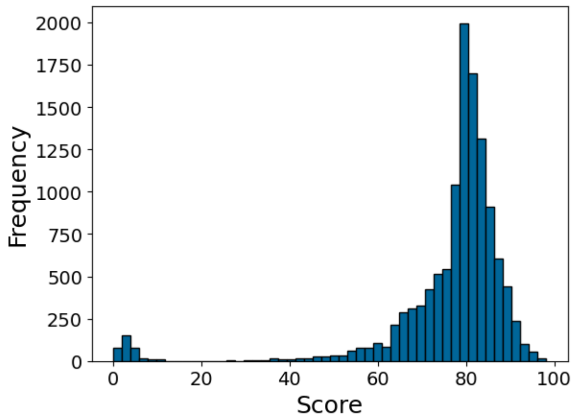
E Experiments Implementation Details

For both foundation models, we conduct training on Azure Machine Learning Studio's cluster [1], utilizing 4 nodes, each equipped with 8 V100 GPUs featuring DeepSpeed Zero-3 (Rajbhandari et al., 2019) offload. Specifically, during the fine-tuning of Mistral 7B, we employ LoRA (Hu et al., 2021). This strategy is chosen for its ability to ensure a more steady convergence of loss, resulting in better performance. The detailed fine-tuning arguments are outlined in Tab.5.

[1] https://ml.azure.com/



(a) Quality Score Distribution in NLU Tasks (20k Data)



(b) Quality Score Distribution in Code Generation Tasks (12k Data)

Figure 7: Quality Score Distribution

F Benchmarks and Compared LLMs

F.1 Benchmarks

Large language model benchmarks serve as standardized tests to evaluate how well models understand, generate, and manipulate human-like language (Lu et al., 2021; Chen et al., 2021). Below is an introduction to these chosen benchmarks:

- **HellaSwag** (Zellers et al., 2019). HellaSwag is a challenge dataset containing 70k multiple-choice questions for evaluating commonsense Natural Language Inference (NLI). While its questions may be trivial for humans (>95% accuracy), they pose a challenge for state-of-the-art models.

- **ARC Challenge** (Clark et al., 2018). The AI2's Reasoning Challenge (ARC) dataset is a multiple-choice question-answering dataset containing questions from science exams ranging from grade 3 to grade 9. It is split into two

Table 5: Fine-tuning Arguments for StarCoder 15B and Mistral 7B

Arguments	StarCoder	Mistral
model_max_length	1024	2048
batch_size	8	8
num_epoch	3	3
learning_rate	2e-5	2e-5
fp16	True	True
lora_r	-	16
lora_alpha	-	32
lora_dropout	-	0.05

partitions: Easy and Challenge. The Challenge partition consists of 25k questions that require reasoning.

- **TruthfulQA** (Lin et al., 2022). TruthfulQA is a benchmark designed to measure whether a language model is truthful in generating answers to questions. The benchmark comprises 817 questions spanning 38 categories. Questions are crafted so that some humans might answer falsely due to false beliefs or misconceptions.

- **MMLU** (Hendrycks et al., 2020). MMLU (Massive Multitask Language Understanding) is a new benchmark intended to measure knowledge acquired during pretraining. It evaluates models exclusively in zero-shot and few-shot settings, making it more challenging and akin to human evaluation. The benchmark covers 57 subjects across STEM, humanities, social sciences, and more, ranging in difficulty from elementary to advanced professional levels, testing both world knowledge and problem-solving ability.

- **HumanEval** (Chen et al., 2021). HumanEval is utilized to gauge functional correctness in synthesizing programs from docstrings. Comprising 164 original programming problems, it assesses language comprehension, algorithms, and simple mathematics.

- **MBPP** (Austin et al., 2021). The MBPP (Mostly Basic Python Problems) dataset consists of around 1,000 crowd-sourced Python programming problems. These are designed to be solvable by entry-level programmers, covering programming fundamentals and standard library functionality. In our experiments, to align with others, we select 400 questions.

F.2 Compared LLMs

The selected models for comparison in NLU tasks include:

- **LLaMA** (Touvron et al., 2023a). LLaMA is a collection of foundation language models trained on trillions of tokens from publicly available datasets.

- **LLaMA2** (Touvron et al., 2023b). Llama 2 is an updated version of Llama, trained on a new mix of publicly available data. It increased the size of the pretraining corpus by 40%, doubled the context length of the model, and adopted grouped-query attention in training.

- **Mistral** (Jiang et al., 2023). Mistral is a state-of-the-art 7B foundational model, fast-deployed, easily customizable, and supports English and code with an 8k context length. It's also one of the foundation models in our paradigm experiments.

- **Vicuna** (Chiang et al., 2023). Vicuna is an open-source chatbot trained by fine-tuning LLaMA on 70K user-shared conversations collected from the ShareGPT website.

- **WizardLM** (Xu et al., 2023). WizardLM is instruction fine-tuned on LLaMA with 70k instruction data generated through the Evol-Instruct strategy.

- **Platypus** (Lee et al., 2023). Platypus is a family of fine-tuned and merged LLMs achieving strong performance. It uses Open Platypus as its instruction dataset and applies the LoRA strategy to train adaptors that can be merged into different foundation models, creating many variant models.

The selected models for comparison in code generation tasks include:

- **CodeT5+ & InstructionCodeT5+** (Wang et al., 2023). CodeT5+ is a new family of open code LLMs with an encoder-decoder architecture trained on various pretraining tasks. InstructionCodeT5+ is further fine-tuned on the Code Alpaca dataset.

Table 6: Component Ablation Experiment Results in NLU Tasks (Foundation Model: Mistral)

Component Type	HellaSwag	ARC Challenge	TruthfulQA	MMLU
Original Dataset (25k)	0.828	0.615	0.445	0.626
Expansion Only (100k)	0.833	0.624	0.447	0.630
Curation Only (15k)	0.830	0.621	0.443	0.623
Paradigm LIFT (15k)	**0.843**	**0.644**	**0.491**	**0.645**

Table 7: Component Ablation Experiment Results (Pass@1) in Code Generation Tasks (Foundation Model: Code LLaMA)

Component Type	HumanEval	MBPP
Original Dataset (20k)	0.410	0.467
Expansion Only (60k)	0.535	0.488
Curation Only (10k)	0.475	0.480
Paradigm LIFT (10k)	**0.551**	**0.498**

Table 8: Analysis of GPU hours and Carbon Emission with different dataset size. GPU hours in the table are measured in *hour*, CO_2 emission is in *kg CO_2 eq*.

Dataset Size	GPU Hours	CO_2 Emission
NLU Tasks		
Original 25k	40.24	3.62
Expanded 100k	149.76	13.48
Curated 15k	23.71	2.13
Code Generation Tasks		
Original 20k	50.82	4.58
Expanded 60k	185.6	16.7
Curated 10k	31.6	2.84

- **Code LLaMA** (Rozière et al., 2023). Code Llama is a code-specialized version of Llama2 (Touvron et al., 2023b) trained on code-specific datasets.
- **StarCoder** (Li et al., 2023b). StarCoder is a widely-used large code language model trained on diverse sources, including 80+ programming languages, Git commits, GitHub issues, and Jupyter notebooks. It's also one of the foundation models in our paradigm experiments.
- **WizardCoder** (Luo et al., 2023). WizardCoder is instruction fine-tuned on StarCoder with 78k instruction data generated through the application of Code Evol-Instruct.

G Component Ablation Experiments

Tab. 6 and 7 are the component ablation result for NLU tasks and code generation tasks, demonstrating the impact of each component in our methodology.

H GPU Hours and Carbon Emission

By compressing the size of the instruction dataset, we aim to reduce the GPU hours required for instruction tuning, contributing to a subsequent decrease in carbon emissions. Tab.8 illustrates the impact of different dataset sizes on GPU hours and CO_2 emissions. We consider three datasets for each task: the original dataset, the expanded dataset after the first step of our paradigm, and the final curated dataset. GPU hours are calculated under the same settings of training epoch and batch size, while carbon emissions are computed using an online machine learning CO_2 calculator[2].

The table shows a substantial reduction in GPU hours and lower carbon emissions when fine-tuning with the final curated dataset. Specifically, compared to the original dataset, we observe a 36.8% and 41.1% reduction in GPU hours for code generation and NLU tasks, respectively. This comparison demonstrates that our paradigm not only accelerates fine-tuning but also promotes environmental sustainability while maintaining robust high performance.

[2] https://mlco2.github.io/impact/#co2eq