From Selection to Refinement: Iterative Optimization for Instruction Data

Hang Hu[◇*], **Ziyan Liu**[◇*], **Rujie Wen**[♡], **Ruihui Hou**[◇], **Xueyan Wu**[◇], **Mu Zhang**[◇], **Jianxing Yu**[♠], **Tong Ruan**[◇†], **Jingping Liu**[♣†]

[◇]School of Information Science and Engineering, East China University of Science and Technology, Shanghai, China

[♡]College of Intelligent Systems Science and Engineering, Chengdu Neusoft University, Chengdu, China

[♠]School of Artificial Intelligence, Sun Yat-sen University, Zhuhai, China

[♣]School of Software Engineering, Sun Yat-sen University, Zhuhai, China

{y80240124,y30241069}@mail.ecust.edu.cn, liujp68@mail.sysu.edu.cn

Abstract

Instruction tuning plays a crucial role in enhancing large language models (LLMs) to better understand complex user instructions. While various data selection and revision methods have been explored to optimize instruction tuning datasets, they face two main challenges: unreasonable pruning of potentially valuable low-quality data and the persistence of noise or semantic drift during revision. To address these issues, we propose a novel automated iterative framework for instruction data optimization. Our framework introduces Instruction Quality Differentiation to identify valuable high-quality and low-quality data across multiple dimensions. For low-quality data, we propose a Feedback-driven Iterative Refinement mechanism with an "evaluate-refine-review" process and design an Output Alignment module to improve data quality. Experiments on seven public benchmark datasets show that our framework outperforms state-of-the-art methods, achieving 2.09% and 2.60% improvements on the Alpaca and Dolly datasets, respectively, with high data efficiency. Our code and data are available at the anonymous link https://github.com/surihuhang/From-Selection-to-Refinement--Iterative-Optimization-for-Instruction-Data.

1 Introduction

Large language models (LLMs) have recently shown strong performance on diverse downstream tasks, including knowledge understanding and logical reasoning (Zhao et al., 2024; Xu et al., 2025; Huang and Chang, 2023; Hou et al., 2026). A key driver of this success is instruction tuning, which improves instruction-following ability by training models on instruction–input–output tuples in a supervised manner (Ye et al., 2024; Zhang et al., 2025c).

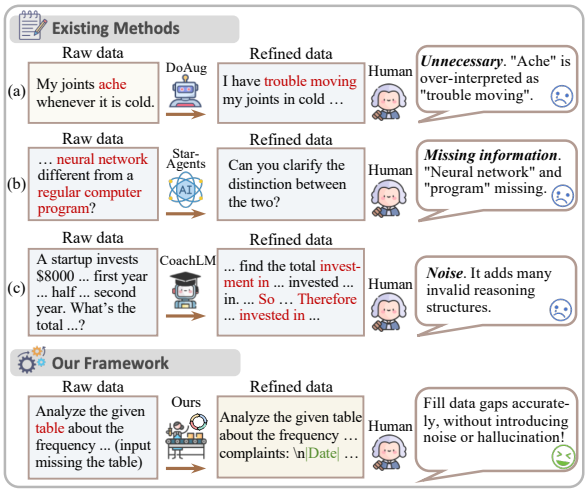

Figure 1: Limitations of existing instruction data optimization methods.

High-quality instruction data are a prerequisite for effective instruction tuning (Zhang et al., 2025a; Wu et al., 2023). Early instruction datasets are mainly constructed through manual annotation, which ensures high quality but suffers from high cost and limited scalability (Zhang et al., 2025b, 2023). Hence, researchers have further turned to automatically generating instruction data via knowledge distillation from stronger teacher models (Fang et al., 2025; Shirgaonkar et al., 2024). However, due to hallucinations and instability in teacher models, the generated data are often noisy, exhibiting issues such as incorrect answers, missing information, and redundant content (Huang et al., 2025; Saha et al., 2023). Training on such imperfect data makes it challenging to fully exploit the performance potential of LLMs (Amin et al., 2025; Chen et al., 2024). Consequently, efficiently selecting and revising instruction tuning data has become a critical problem in practice.

Current data optimization methods can be categorized into two main approaches: data selection and data revision, both of which still fail to achieve sufficient optimization. Data selec-

[*]Equal contributions.

[†]Corresponding authors.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 40670–40684

tion focuses on choosing subsets with desirable properties from the raw dataset, typically emphasizing quality-based selection (e.g., LIMA (Zhou et al., 2023a)), diversity-based selection (e.g., InsTag (Lu et al., 2023)), difficulty-based selection (e.g., IFD (Li et al., 2024b)), and combined criteria (e.g., DEITA (Liu et al., 2024a)). However, a key limitation of these approaches is their tendency to discard potentially valuable data that could still be useful despite imperfections, leading to data waste. Hence, data revision methods have been introduced to improve data quality by addressing flaws in the original data. Typical studies along this line include rule-based filling (e.g., MathFusion (Pei et al., 2025)), teacher-model distillation (e.g., TRAIT (Liang et al., 2024)), and preference-guided revision (e.g., DoAug (Wang et al., 2025), WizardLM (Xu et al., 2024)). However, these methods still face two significant challenges: 1) They fail to effectively distinguish between useful data for model training and data that needs refinement, as some data may be of little value or already high quality. 2) They often rely on "single-pass rewriting" strategy, which may not fully optimize the data, resulting in rewritten versions with noise and semantic drift, as illustrated in Figure 1.

To this end, we propose an automated framework for instruction data optimization, which uses an iterative strategy that first selects and then refines data. The core idea is to identify valuable instruction data through multi-dimensional differentiation, retaining high-quality data while refining low-quality data through a feedback-driven iterative mechanism and output alignment. Specifically, we first introduce an Instruction Quality Differentiation (IQD) strategy, which identifies useful data for model training and distinguishes high- and low-quality data based on three dimensions: expert-guided quality labeling, semantic diversity, and instruction difficulty. For low-quality data, we then propose a Feedback-driven Iterative Refinement (FIR) mechanism that follows an "evaluate-refine-review" process. This process compares model predictions with original outputs to identify errors and generate feedback. A refinement expert improves the data, and a review expert verifies each refinement, iterating until optimization stabilizes. Finally, we design an Output Alignment (OA) module, which ensures the semantic and logical consistency between instruction-input pairs and their outputs by anchoring keywords. The optimized data is then combined with the high-quality data to create a final dataset for training.

Our contributions are summarized as follows:

- We propose an automated iterative framework for instruction data optimization, which integrates a multi-dimensional IQD module, an "evaluate-refine-review" FIR module, and an OA module.
- We apply our framework to optimize two widely used instruction datasets, Alpaca and Dolly, generating and releasing 35K and 12K high-quality instruction samples.
- Experimental results show that our framework outperforms the state-of-the-art (SoTA) method (WizardLM) on 7 public benchmarks, achieving average improvements of 2.09% and 2.60% on Alpaca and Dolly, respectively. Our framework also demonstrates high data efficiency, using only 35K and 12K samples, far fewer than the 70K and 59K required by WizardLM.

2 Related Work

Related work in this study can be divided into two categories: data selection and data revision.

Data selection. These methods aim to extract useful instances from large raw dataset to improve model performance. Existing approaches can be divided into the following types: 1) Quality-centric methods focus on reducing noise to preserve clean data. AlpaGasus (Chen et al., 2023) introduces an LLM-based strategy to score and filter data. 2) Diversity-centric methods aim to broaden semantic coverage to mitigate overfitting. InsTag (Lu et al., 2023) curates a diverse subset based on label distribution. 3) Difficulty-centric methods evaluate how hard each instance is for the model. Li et al. (Li et al., 2024b) propose the IFD metric, which measures the loss difference between responses with instructions and those without instructions. 4) Combined-criteria strategies integrate multiple metrics to balance different data properties. Approaches like DEITA (Liu et al., 2024a) and MoDs (Du et al., 2023) jointly consider quality, diversity, and complexity for comprehensive data selection. However, a significant limitation of these methods is their tendency to discard potentially valuable data that could be useful despite a few defects, leading to considerable data waste.

Data revision. Data revision methods improve data quality by directly correcting flaws in raw

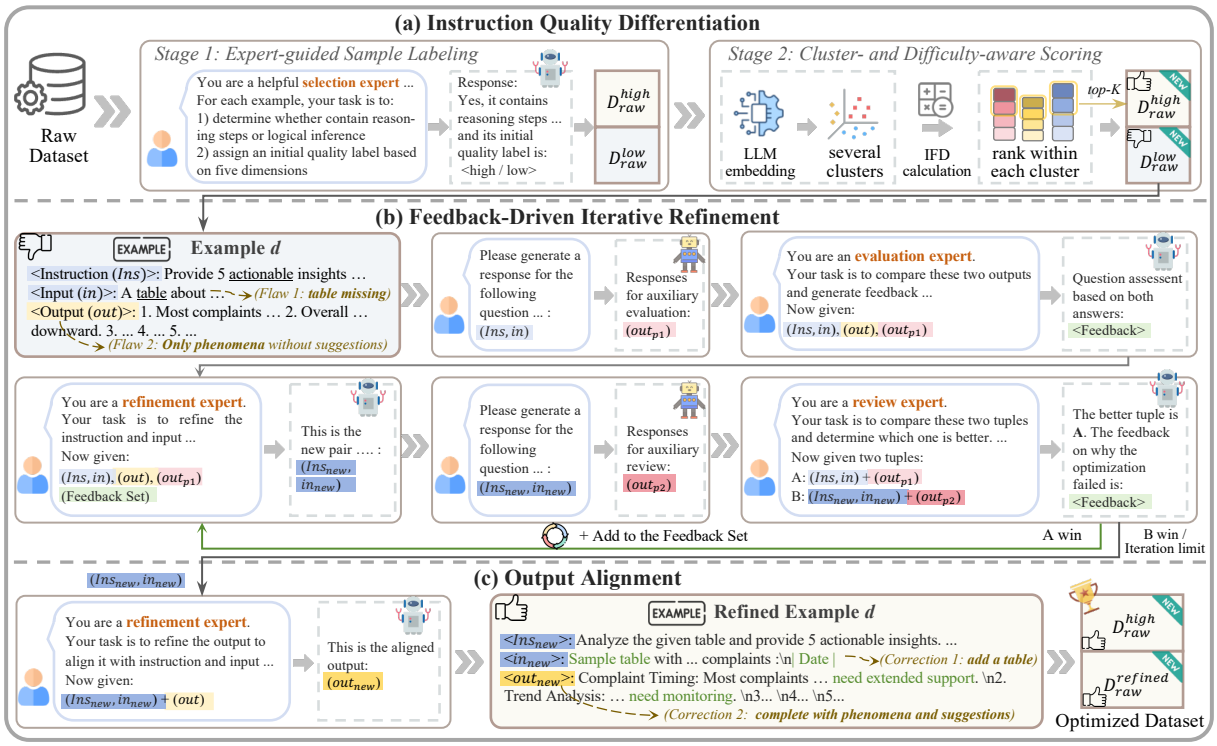

Figure 2: Overview of our instruction data optimization framework.

instances. These methods fall into three categories: 1) Rule-based filling. These methods revise data using predefined rules or structural templates. MathFusion (Pei et al., 2025) combines two mathematical problems according to sequential, parallel, or conditional rules. 2) Teacher-model distillation. These methods improve data quality by using advanced models as teachers to distill knowledge into training data. TRAIT (Liang et al., 2024) augments the data by directing teacher models to generate detailed solution steps or rationales. CoachLM (Liu et al., 2024b) extends this approach by training a teacher model on human-revised data. 3) Preference-guided revision. These methods revise data based on explicit, customizable preferences. DoAug (Wang et al., 2025) learns human preferences using the DPO algorithm. WizardLM (Xu et al., 2024) focuses on complexity preferences by iteratively revising instructions. Star-Agents (Zhou et al., 2024) uses multi-agent scoring to promote data diversity and quality during data optimization. However, these methods face two key limitations: first, they rewrite data without distinction, which may degrade the semantic precision of original high-quality data; second, they rely on single-pass generation, often failing to fully correct errors and sometimes introducing noise and causing semantic drift.

3 Overview

Problem definition. Instruction data optimization refers to improving the quality of the tuples consisting of an instruction, input, and output for LLM training. Formally, given an instruction dataset $D_{raw} = \{(Ins_i, in_i, out_i)\}_{i=1}^{N}$, where each tuple (Ins_i, in_i, out_i) denotes one instruction sample, the goal is to transform $(Ins_i, in_i, out_i) \rightarrow (Ins_{new_i}, in_{new_i}, out_{new_i})$ to obtain an optimized dataset $D_{opt} = \{(Ins_{new_i}, in_{new_i}, out_{new_i})\}_{i=1}^{M}$, where $M \leq N$. The optimized dataset must satisfy:

$$\mathcal{M}(D_{opt}) > \mathcal{M}(D_{raw}), \quad (1)$$

where $\mathcal{M}(\cdot)$ denotes the model's evaluation metric on downstream tasks after training on dataset D.

Framework overview. As illustrated in Figure 2, our framework consists of three main modules. First, we detect valuable instruction data for model training and differentiate the raw instruction dataset D_{raw} into high-quality (D_{raw}^{high}) and low-quality (D_{raw}^{low}) subsets based on three criteria: instruction quality, diversity, and difficulty. This differentiation is achieved using a combination of a prompt-based LLM, clustering technique, and difficulty assessment metric. Second, for the low-quality dataset D_{raw}^{low}, we introduce a feedback-driven iterative refinement method to improve the instruction-input pairs. This process involves three

stages: evaluation, refinement, and review. Initially, the data is evaluated, and feedback is given to a refinement expert to make improvements. A review expert then checks the changes, and if the result is unsatisfactory, the process is repeated until approval is given or the maximum number of iterations is reached. Finally, we design an output alignment module to optimize the output, ensuring it aligns with the refined instruction and input. The optimized low-quality data is then merged with the high-quality dataset D_{raw}^{high} for model training.

4 Methodology

In this section, we provide a detailed description of the three modules in our framework.

4.1 Instruction Quality Differentiation

This module aims to detect valuable data for LLM training and divide instruction samples into high- and low-quality subsets. Given a raw instruction dataset $D_{raw} = \{(Ins_i, in_i, out_i)\}_{i=1}^{N}$, we first introduce a selection expert (a powerful LLM with a designed prompt) to select samples that contribute to the model's reasoning ability and assign each selected sample an initial quality label (high or low), forming two subsets D_{raw}^{high} and D_{raw}^{low}. The prompt consists of two components (refer to Appendix A): (1) determining whether an instruction sample contains explicit reasoning steps or latent logical inference, and (2) assigning an initial quality label based on five predefined dimensions: Correctness, Completeness, Clarity, Instruction Consistency, and Logical Reasoning (see Appendix B).

Since different LLMs may exhibit distinct preferences, the resulting coarse labels may be biased, leading to an unreliable distinction between high- and low-quality data (Thomas et al., 2024; Yu et al., 2024). Hence, we further introduce a cluster- and difficulty-aware scoring strategy that jointly considers semantic diversity and instruction difficulty. Specifically, we use the embedding layer of an LLM to obtain the representation of every samples in D_{raw}^{high} and D_{raw}^{low}, and then employ the K-means algorithm (Likas et al., 2003) to cluster the dataset, grouping semantically similar instructions while ensuring diversity across clusters. Next, for each cluster, we compute an IFD (Li et al., 2024b) score for every sample using an LLM. The IFD score quantifies the difficulty of each instruction for the model, with higher values indicating more challenging samples. Within each cluster, we rank all samples primarily based on the initial quality labels in D_{raw}^{high} and D_{raw}^{low}, and secondarily on the IFD score. Based on this ranking, we select the top-K samples from each cluster as high-quality data, forming the new D_{raw}^{high}, and classify the remaining samples as low-quality data, forming the new D_{raw}^{low}. The detailed analysis of the cluster number and K is in Appendix D.

4.2 Feedback-Driven Iterative Refinement

The goal of this module is to enhance the quality of instruction-input pairs in D_{raw}^{low} produced in the previous module. This is achieved through an automated "evaluate-refine-review" iterative process. Specifically, for each sample $d = (Ins, in, out) \in D_{raw}^{low}$, we first use an LLM to generate a response, denoted as out_{p1}, for the (Ins, in) pair. Next, we design an evaluation expert (a more powerful LLM with specialized prompts) to compare out_{p1} with the original output out, identify any discrepancies, and generate detailed feedback (such as vague instructions or missing information). This feedback, along with the original sample d and out_{p1}, is then provided to a refinement expert, which refines both the instruction and input, resulting in a new tuple $(Ins_{new}, in_{new}, out)$.

Direct refinement based on a single round of feedback is often unstable and may not reliably guide the model to generate the desired response. To address this, we employ an iterative optimization strategy. After the first round of refinement, we utilize an LLM to generate a new response for (Ins_{new}, in_{new}), denoted as out_{p2}. A review expert is then employed to compare the tuple (Ins, in, out_{p1}) with the newly generated tuple $(Ins_{new}, in_{new}, out_{p2})$. If the latter outperforms the former, the optimization is considered successful, and the final high-quality instruction-input pair (Ins_{new}, in_{new}) is selected. Otherwise, the former is retained, and a reflection mechanism is triggered, prompting the model to generate detailed feedback on why the optimization failed. This feedback is accumulated into the feedback set E_{iter}, which guides the adjustments in the next iteration, continuing until the new tuple outperforms the original tuple or the iteration limit is reached.

4.3 Output Alignment

The objective of this module is to eliminate semantic discrepancies between the refined instruction-input pairs and the original output. To achieve this, given the (Ins_{new}, in_{new}) pair from the previous

Method	Data Size	Understanding		Math & Code			Reasoning		Avg.
		C-Eval	ARC	HumanEval	GPQA	GSM8K	TriviaQA	Winogrande	
Alpaca Dataset									
Raw Data	52.0k	51.34	84.19	48.17	24.25	44.51	66.99	60.85	54.33
Data Selection									
IFD	5.2k	51.04	86.19	46.95	**28.02**	72.40	62.07	62.59	58.47
ALPAGASUS	15.0k	49.78	84.58	**51.83**	26.93	66.11	66.98	61.48	58.24
NUGGETS	9.2k	51.19	**86.64**	45.12	22.73	67.78	70.59	58.41	57.49
Superfiltering	5.0k	**52.76**	86.58	48.21	24.92	74.07	63.90	59.19	58.52
MoDS	31.0k	52.01	84.50	48.78	23.24	43.44	66.12	59.27	53.91
Data Refinement									
WizardLM	70.0k	51.34	85.65	48.17	26.85	73.12	67.57	59.83	58.93
DoAug	52.0k	43.24	82.22	41.46	15.68	25.25	14.85	57.14	39.98
CoachLM	52.0k	51.26	85.37	15.24	27.18	69.52	70.83	58.64	54.01
Ours	35.0k	52.67	86.33	49.39	27.68	**76.95**	**71.09**	**63.06**	**61.02**
Dolly Dataset									
Raw Data	15.0k	51.04	84.58	37.20	22.31	72.18	63.24	59.12	55.67
Data Selection									
IFD	1.5k	51.56	86.47	37.20	19.80	71.04	65.82	45.86	53.96
ALPAGASUS	4.5k	52.30	86.42	35.98	21.48	75.06	67.63	48.30	55.31
NUGGETS	2.7k	51.41	86.92	37.20	21.98	75.66	**70.78**	34.41	54.05
Superfiltering	1.5k	52.15	86.58	42.07	22.32	74.15	66.90	53.04	56.74
MoDS	11.0k	52.45	85.88	40.85	25.59	71.37	62.16	55.64	56.28
Data Refinement									
WizardLM	59.0k	51.93	**87.06**	33.54	27.77	77.86	66.62	59.75	57.79
DoAug	15.0k	50.59	84.42	37.20	21.90	67.78	11.94	52.33	46.59
CoachLM	14.9k	53.86	85.57	25.61	25.59	71.80	70.25	55.49	55.45
Ours	12.0k	**54.31**	86.83	**45.12**	**27.87**	**82.26**	65.79	**60.62**	**60.39**

Table 1: Model comparisons (%) across two instruction tuning datasets. All results are averaged over three runs. The best and second best results are highlighted in **bold** and underline, respectively.

module and the original output out, we introduce a refinement expert that identifies the core keywords in $(Ins_{new}, in_{new}, out)$. Based on these keywords, the expert refines the output, resulting in a new version, out_{new}. The (Ins_{new}, in_{new}) pair is combined with the refined output out_{new} to form a new, high-quality tuple. Finally, the refined results of each sample in D_{raw}^{low} are merged with D_{raw}^{high} to create the final optimized instruction dataset, D_{opt}, which is used for model training.

5 Experiment

In this section, we first conduct extensive experiments to evaluate our proposed framework. We further provide a series of detailed analyses to gain a deeper understanding of the framework.

5.1 Experimental Setup

Training datasets. We employ the framework for data optimization on two widely used instruction tuning datasets respectively: (1) Alpaca (Taori et al., 2023), which contains 52,000 GPT-3–generated instruction-following examples that have been manually curated; and (2) Dolly (Conover et al., 2023), which consists of 15,000 human-written examples designed to improve LLMs' ability to follow instructions. **New instruction dataset**: After optimization by our framework on Alpaca and Dolly, we generate 35K and 12K new high-quality instruction samples, respectively. A detailed analysis of these results is provided in Appendix E.

Benchmarks. We evaluate our framework on seven benchmarks covering three categories: knowledge understanding, math and code problem solving, and knowledge reasoning. Knowledge understanding is evaluated using C-Eval (Huang et al., 2023) and ARC (Clark et al., 2018). Math and code problem solving is assessed with HumanEval (Chen, 2021), GSM8K (Cobbe et al., 2021), and GPQA (Rein et al., 2024). Knowledge reasoning is evaluated using TriviaQA (Joshi et al., 2017) and Winogrande (Sakaguchi et al., 2021).

Baselines. We compare our framework with several SoTA methods, which fall into two categories: data selection and data revision. Data selection methods include IFD, ALPAGASUS, NUGGETS (Li et al., 2024c), Superfiltering (Li et al., 2024a) and MoDS, while data revision methods include WizardLM, DoAug, and CoachLM.

Method	Data Size	Understanding		Math & Code			Reasoning		Avg.
		C-Eval	ARC	HumanEval	GPQA	GSM8K	TriviaQA	Winogrande	
Raw Data	52k	51.34	84.19	48.17	24.25	44.51	66.99	60.85	54.33
w/o IQD	52k	52.38	86.58	43.90	20.55	75.13	64.09	60.93	57.65
w/o FIR	35k	51.86	84.47	50.00	27.52	58.23	68.06	62.59	57.53
w/o OA	35k	51.23	86.90	15.85	28.60	83.47	69.11	60.14	56.47
Ours	35k	52.67	86.33	49.39	27.68	76.95	71.09	63.06	**61.02**

Table 2: Ablation study on the main modules of our framework.

Method	Data Size	Understanding		Math & Code			Reasoning		Avg.
		C-Eval	ARC	HumanEval	GPQA	GSM8K	TriviaQA	Winogrande	
Raw Data	52k	51.34	84.19	48.17	24.25	44.51	66.99	60.85	54.33
w/o IQL	52k	52.97	85.88	42.07	26.26	78.85	66.71	62.51	59.32
w/o DIV	35k	52.52	86.53	47.56	23.91	74.68	66.92	63.77	59.41
w/o DIF	35k	51.64	84.55	50.61	25.92	54.81	67.78	63.30	56.94
Ours	35k	52.67	86.33	49.39	27.68	76.95	71.09	63.06	**61.02**

Table 3: Ablation study on the Instruction Quality Differentiation module. "IQL", "DIV", and "DIF" denote the "initial quality label", semantic "diversity", and instruction "difficulty" dimensions, respectively.

Metrics. Following prior work (Du et al., 2024), we use Accuracy as the evaluation metric for C-Eval, ARC, GSM8K, TriviaQA, and Winogrande. In addition, we adopt Pass@1 and Average Pass@1 as the evaluation metrics for HumanEval and GPQA, respectively.

Implementation details. Unless otherwise specified, all experts in this paper are built on Llama3.1-70B-Instruct (Grattafiori et al., 2024), while Llama3.1-8B-Instruct (Grattafiori et al., 2024) is adopted as the default backbone and is also used for fine-tuning on the optimized data. During fine-tuning, we employ a cosine learning rate schedule with an initial learning rate of 1×10^{-6} and a warm-up ratio of 0.1, training the models for three epochs. During inference, the temperature is set to 0.7 and top-p is set to 0.1. All experiments are conducted on Ubuntu 20.04.6 LTS, each equipped with four NVIDIA A100 GPUs (80 GB). Detailed computational resource statistics are provided in Appendix F.

5.2 Main Results

We apply our proposed framework and eight baseline methods on the Alpaca and Dolly datasets to construct optimized training sets. Each optimized dataset is then used to fine-tune the same backbone model, and the resulting models are evaluated on seven benchmarks. The detailed results are reported in Table 1.

Overall, our framework consistently outperforms all baselines under both instruction dataset settings. Specifically, on Alpaca, the model fine-tuned with data optimized by our framework achieves an average performance of 61.02%, exceeding the best data selection and data revision baselines by 2.50% and 2.09%, respectively. Similarly, on Dolly, our framework attains an average performance of 60.39%, surpassing the previous best results by 3.65% and 2.60%. These results demonstrate that our data optimization framework effectively improves overall data quality by selecting to retain high-quality data and correcting low-quality samples, thereby enhancing model performance. Moreover, our framework achieves these gains with substantially less training data. Compared to the strongest baseline (WizardLM), which expands Alpaca and Dolly to 70K and 59K samples, our method uses only 35K and 12K samples, reducing data usage by 50.00% and 79.66%, respectively. Finally, under both dataset settings, our framework yields significant improvements over models trained on raw data across all benchmarks, indicating strong robustness. For example, on GSM8K, our approach improves performance by 32.44% and 10.08% compared to models fine-tuned on the raw Alpaca and Dolly datasets, respectively.

5.3 Ablation Study

Ablation of main modules. To evaluate the importance of each module in our framework, we sequentially remove one module and assess the effectiveness of the remaining module combinations on Alpaca dataset. The experimental results are reported

Max Iterations	Understanding		Math & Code			Reasoning		Avg.
	C-Eval	ARC	HumanEval	GPQA	GSM8K	TriviaQA	Winogrande	
Raw Data	51.34	84.19	48.17	24.25	44.51	66.99	60.85	54.33
$T = 1$	52.45	48.17	76.19	25.17	86.36	67.04	63.30	59.81
$T = 2$	52.82	45.12	77.03	28.02	85.88	67.94	62.43	59.89
$T = 3$	52.67	49.39	76.95	27.68	86.33	71.09	63.06	**61.02**
$T = 4$	51.93	42.68	76.57	25.42	85.54	66.09	62.51	58.68
$T = 5$	52.82	45.12	76.88	25.42	86.05	66.42	52.82	57.93

Table 4: Analysis of maximum iterative refinement rounds.

Model	Method	Understanding		Math & Code			Reasoning		Avg.
		C-Eval	ARC	HumanEval	GPQA	GSM8K	TriviaQA	Winogrande	
Llama3.1-8B	Raw	51.34	84.19	48.17	24.25	44.51	66.99	60.85	54.33
	Ours	52.67	86.33	49.39	27.68	76.95	71.09	63.06	**61.02**
Mistral-7B-v0.3	Raw	41.01	74.97	21.95	22.48	18.27	64.75	57.54	43.00
	Ours	41.01	74.38	29.27	24.83	21.43	66.02	59.83	**45.25**
Qwen2.5-7B	Raw	71.99	87.54	43.90	25.76	66.72	53.97	63.46	59.05
	Ours	74.00	90.11	44.51	27.27	77.33	66.62	66.06	**63.70**
Qwen2.5-14B	Raw	75.78	91.18	51.22	28.86	75.36	68.57	71.98	66.14
	Ours	79.72	92.28	48.17	28.94	89.08	71.70	75.69	**69.37**
Qwen2.5-32B	Raw	76.30	92.11	45.12	30.03	75.13	70.79	76.48	66.57
	Ours	80.39	93.37	46.34	35.32	89.76	72.58	79.48	**71.03**

Table 5: Results of adapting our framework to different backbone models. Each model includes two rows: "Raw" shows fine-tuning with the original dataset, and "Ours" shows fine-tuning with the optimized dataset.

in Table 2. As shown in the table, removing any single module leads to a noticeable performance degradation, with drops of 3.37%, 3.49%, and 4.55%, respectively. Nevertheless, all ablated variants still significantly outperform the "Raw data" baseline, indicating that each module contributes positively to the overall framework. In particular, removing the IQD module allows a large amount of simple or low-quality noisy data to be included in training, which weakens the model's ability to handle challenging tasks; for example, performance on GPQA drops by 7.13%. Removing the FIR module results in substantial performance drops on mathematical and reasoning tasks, with decreases of 18.72% on GSM8K and 3.03% on TriviaQA, respectively. Furthermore, removing the OA module has the most severe impact on code-related tasks such as HumanEval, where performance sharply decreases from 49.39% to 15.85%. This is because the absence of OA introduces severe semantic deviation, breaking the logical consistency between instruction-input pairs and outputs.

Ablation of dimensions in IQD. The IQD module distinguishes high- and low-quality instruction data based on an LLM-generated initial quality label (IQL), data diversity (DIV) and difficulty (DIF). To evaluate the importance of these dimensions, we conduct ablation studies on Alpaca dataset, with results shown in Table 3. We observe that removing any single dimension leads to a noticeable degradation in overall performance, demonstrating that all dimensions contribute to the effectiveness of the framework. In particular, removing the DIF dimension causes the largest drop in average performance (a 4.08% decrease compared to our full framework), indicating that, beyond general capabilities, improving the model's ability to address more challenging knowledge and skills is crucial.

5.4 Detailed Analysis

Impact of iterative refinement rounds. We analyze the effect of the number of refinement iterations T in the FIR module (Section 4.2). Specifically, we vary T from 1 to 5, and report the results in Table 4. Performance improves substantially during the first few iterations and reaches its best at $T = 3$, suggesting that a small number of iterations is sufficient to form an effective "evaluate–refine–review" cycle. When further increasing T, performance slightly degrades, which we attribute to over-editing introduced by excessive refinements. Accordingly, we adopt $T = 3$ as the default setting in our final framework.

Versatility across different backbone models. We evaluate our framework on a diverse set of open-source LLMs with varying architectures (e.g.,

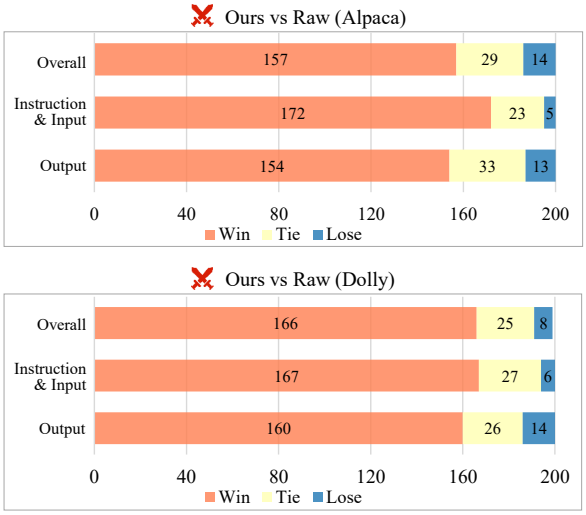

Figure 3: Human evaluation results on Alpaca and Dolly. Evaluations are reported at three levels: Overall data, Instruction & Input, and Output.

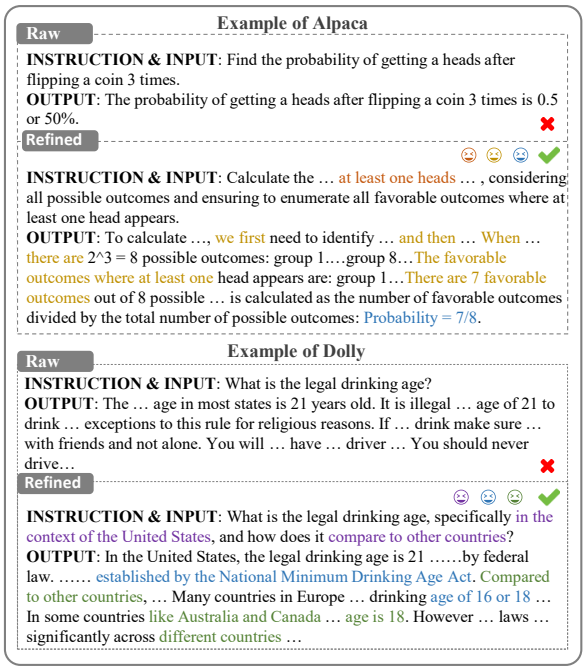

Figure 4: Case study on Alpaca and Dolly. The color meanings are explained in the main text.

Llama, Mistral, and Qwen) and parameter scales (7B, 14B, and 32B). The results are summarized in Table 5. We notice that: 1) our framework consistently improves performance across different backbone architectures. For instance, it yields average gains of 6.69% on Llama3.1-8B and 4.65% on Qwen2.5-7B, indicating that its effectiveness is not limited to a specific model family. 2) Within the same model family (Qwen2.5), our framework delivers stable improvements across different model sizes, achieving gains of 4.65%, 3.23%, and 4.46% on the 7B, 14B, and 32B variants, respectively. These results demonstrate the robustness and scalability of our framework across both architectures and model scales.

Human evaluation. We further validate the quality of the refined data through human evaluation. Specifically, we randomly sample 200 instances from the Alpaca and Dolly datasets respectively and conduct anonymized pairwise comparisons between the raw data and the versions refined by our framework. Three evaluators with computer science backgrounds independently assess each pair, labeling the refined version as Win, Tie, or $Lose$ relative to the original, with the final label determined by majority voting. As shown in Figure 3, our method demonstrates a clear advantage in overall quality. The refined data achieve win rates of 78.5% (157/200) on Alpaca and 83.0% (166/200) on Dolly, respectively, indicating substantial improvements over the raw data. Notably, the win rate for the Instruction–Input component is consistently higher than that for the Output component, which further supports the effectiveness of the feedback-driven iterative refinement strategy.

Case study. To demonstrate the significant improvement in data quality, we compare the samples before and after optimization, as illustrated in Figure 4. The optimized data shows enhancements in five key dimensions: completing missing key information in the instruction (orange), clearly defining output constraints (purple), adding necessary reasoning steps to the output (yellow), correcting errors in the answers or conclusions (blue), and aligning the output with all key points in the instruction (green). In Case 1 (Alpaca data), the optimized instruction completes the missing key phrase "at least one head", ensuring clarity, while the output adds reasoning and corrects the original error. In Case 2 (Dolly data), the instruction is clarified by adding the constraint "United States", and the output is enriched with a comparison to other countries, ensuring full alignment with the instruction. Overall, our optimization method effectively enhances data quality across multiple dimensions.

6 Conclusion

In this paper, we propose an automated framework for instruction data optimization. The framework distinguishes high-quality and low-quality instruction samples based on multi-dimensional analysis, and refines low-quality data using a feedback-

driven iterative process and output alignment. Experimental results on the Alpaca and Dolly datasets demonstrate that our framework outperforms current SoTA methods, achieving significant performance improvements with high data efficiency.

Limitations

Instruction data optimization is an offline task, and computational efficiency is also an important consideration. Although our framework improves data quality and model performance, it has limitations in computational efficiency. The iterative refinement process incurs higher computational costs and longer processing times, particularly when dealing with large datasets, as compared to a single-pass revision approach. In future work, we aim to optimize this process through parallel processing and more efficient algorithms to reduce costs while maintaining data quality.

Ethical Consideration

All training and benchmark datasets used in this study are publicly available, and their sources are clearly acknowledged in the paper. Therefore, this work does not involve any privacy, confidentiality, or personal data concerns.

Acknowledgments

This paper was supported by the National Natural Science Foundation of China (No. 62306112), Guangdong Basic and Applied Basic Research Foundation (No. 2026A1515010253), Key-Area Research and Development Program of Guangdong Province (2026B0101100004), National Natural Science Foundation of China (62276279), and Guangdong Basic and Applied Basic Research Foundation (2024B1515020032).

References

Kareem Amin, Sara Babakniya, Alex Bie, Weiwei Kong, Umar Syed, and Sergei Vassilvitskii. 2025. Escaping collapse: The strength of weak data for large language model training. *Preprint*, arXiv:2502.08924.

Jie Chen, Yupeng Zhang, Bingning Wang, Wayne Xin Zhao, Ji-Rong Wen, and Weipeng Chen. 2024. Unveiling the flaws: Exploring imperfections in synthetic data and mitigation strategies for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 14855–14865, Miami, Florida, USA. Association for Computational Linguistics.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and 1 others. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

Mark Chen. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

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. *Preprint*, arXiv:1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

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

Zhengxiao Du, Aohan Zeng, Yuxiao Dong, and Jie Tang. 2024. Understanding emergent abilities of language models from the loss perspective. In *Advances in Neural Information Processing Systems*, volume 37, pages 53138–53167. Curran Associates, Inc.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? Comparing the geometry of BERT, ELMo, and GPT-2 embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 55–65, Hong Kong, China. Association for Computational Linguistics.

Luyang Fang, Xiaowei Yu, Jiazhang Cai, Yongkai Chen, Shushan Wu, Zhengliang Liu, Zhenyuan Yang, Haoran Lu, Xilin Gong, Yufang Liu, Terry Ma, Wei Ruan, Ali Abbasi, Jing Zhang, Tao Wang, Ehsan Latif, Wei Liu, Wei Zhang, Soheil Kolouri, and 5 others. 2025. Knowledge distillation and dataset distillation of large language models: emerging trends, challenges, and future directions. *Artificial Intelligence Review*, 59(1):17.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Ruihui Hou, Dongge Xue, Hongli Sun, Ping He, Weiyan Zhang, and Tong Ruan. 2026. Cdaflow: Enhancing llm clinical decision-making through agentic workflow. *Expert Systems with Applications*, 316:131806.

Jie Huang and Kevin Chen-Chuan Chang. 2023. Towards reasoning in large language models: A survey. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1049–1065, Toronto, Canada. Association for Computational Linguistics.

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2025. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Trans. Inf. Syst.*, 43(2).

Yuzhen Huang, Yuzhuo Bai, Zhihao Zhu, Junlei Zhang, Jinghan Zhang, Tangjun Su, Junteng Liu, Chuancheng Lv, Yikai Zhang, jiayi lei, Yao Fu, Maosong Sun, and Junxian He. 2023. C-eval: A multi-level multi-discipline chinese evaluation suite for foundation models. In *Advances in Neural Information Processing Systems*, volume 36, pages 62991–63010. Curran Associates, Inc.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024a. Superfiltering: Weak-to-strong data filtering for fast instruction-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14255–14273.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024b. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7602–7635, Mexico City, Mexico. Association for Computational Linguistics.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024c. One-shot learning as instruction data prospector for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4586–4601, Bangkok, Thailand. Association for Computational Linguistics.

Xiao Liang, Xinyu Hu, Simiao Zuo, Yeyun Gong, Qiang Lou, Yi Liu, Shao-Lun Huang, and Jian Jiao. 2024. Task oriented in-domain data augmentation. *CoRR*, abs/2406.16694.

Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. 2003. The global k-means clustering algorithm. *Pattern Recognition*, 36(2):451–461. Biometrics.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024a. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *ICLR*.

Yilun Liu, Shimin Tao, Xiaofeng Zhao, Ming Zhu, Wenbing Ma, Junhao Zhu, Chang Su, Yutai Hou, Miao Zhang, Min Zhang, and 1 others. 2024b. Coachlm: Automatic instruction revisions improve the data quality in llm instruction tuning. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 5184–5197. IEEE.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *Preprint*, arXiv:2308.07074.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Qizhi Pei, Lijun Wu, Zhuoshi Pan, Yu Li, Honglin Lin, Chenlin Ming, Xin Gao, Conghui He, and Rui Yan. 2025. Mathfusion: Enhancing mathematic problem-solving of llm through instruction fusion. *CoRR*, abs/2503.16212.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2024. GPQA: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*.

Swarnadeep Saha, Peter Hase, and Mohit Bansal. 2023. Can language models teach? teacher explanations improve student performance via personalization. In *Advances in Neural Information Processing Systems*, volume 36, pages 62869–62891. Curran Associates, Inc.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106.

Anup Shirgaonkar, Nikhil Pandey, Nazmiye Ceren Abay, Tolga Aktas, and Vijay Aski. 2024. Knowledge distillation using frontier open-source llms: Generalizability and the role of synthetic data. *Preprint*, arXiv:2410.18588.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 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.

Paul Thomas, Seth Spielman, Nick Craswell, and Bhaskar Mitra. 2024. Large language models can accurately predict searcher preferences. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 1930–1940, New York, NY, USA. Association for Computing Machinery.

Zaitian Wang, Jinghan Zhang, Xinhao Zhang, Kunpeng Liu, Pengfei Wang, and Yuanchun Zhou. 2025. Diversity-oriented data augmentation with large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22265–22283, Vienna, Austria. Association for Computational Linguistics.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. Wizardlm: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Fangzhi Xu, Qika Lin, Jiawei Han, Tianzhe Zhao, Jun Liu, and Erik Cambria. 2025. Are large language models really good logical reasoners? a comprehensive evaluation and beyond. *IEEE Transactions on Knowledge and Data Engineering*, 37(4):1620–1634.

Seonghyeon Ye, Hyeonbin Hwang, Sohee Yang, Hyeongu Yun, Yireun Kim, and Minjoon Seo. 2024. Investigating the effectiveness of task-agnostic prefix prompt for instruction following. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19386–19394.

Xiao Yu, Zexian Zhang, Feifei Niu, Xing Hu, Xin Xia, and John Grundy. 2024. What makes a high-quality training dataset for large language models: A practitioners' perspective. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, ASE '24, page 656–668, New York, NY, USA. Association for Computing Machinery.

Kai Zhang, Lingbo Mo, Wenhu Chen, Huan Sun, and Yu Su. 2023. Magicbrush: A manually annotated dataset for instruction-guided image editing. In *Advances in Neural Information Processing Systems*, volume 36, pages 31428–31449. Curran Associates, Inc.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. 2025a. Instruction tuning for large language models: A survey. *ACM Comput. Surv.* Just Accepted.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Guoyin Wang, and Fei Wu. 2025b. Instruction tuning for large language models: A survey. *ACM Comput. Surv.* Just Accepted.

Yi-Kai Zhang, De-Chuan Zhan, and Han-Jia Ye. 2025c. Capability instruction tuning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(24):25958–25966.

Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2024. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6889–6907.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. Lima: Less is more for alignment. In *Advances in Neural Information Processing Systems*, volume 36, pages 55006–55021. Curran Associates, Inc.

Hang Zhou, Yehui Tang, Haochen Qin, Yujie Yang, Renren Jin, Deyi Xiong, Kai Han, and Yunhe Wang. 2024. Star-agents: Automatic data optimization with llm agents for instruction tuning. *Advances in Neural Information Processing Systems*, 37:4575–4597.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023b. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*.

A Prompt of the IQD Module

The core prompt used for the powerful LLM of IQD is shown below:

The Prompt Template For Sample Differentiation and Quality Labeling

```
You are a data selection expert. Given an instruction, input and the corresponding output, perform the following tasks:

1. Differentiation: Determine whether the instruction sample contains explicit reasoning steps or latent logical inference (such as inference, comparison, or multi-step logic). This includes abstract thinking, causality, or analogies.

· If it does not, output:
- RESPONSE: No.
and stop.

2. Labeling: If applicable, assign an initial quality label based on the following five dimensions:

· Correctness: ...
· Completeness: ...
· Clarity: ...
· Instruction Consistency: ...
· Logical Reasoning: ...

Assign a binary score:

· High label: It is correct, clear, mostly complete, and shows in-depth reasoning ...
· Low label: It is generally valid but weaker in one or more aspects (e.g., shallow reasoning, missing detail, unclear expression) ...

Consider whether the reasoning is explicit and step-by-step to help with model learning.

# Output Format (strictly follow):
- RESPONSE:
- Determination: <Yes or No>
- Quality label: <High or Low>

# Now given:
- Instruction: {}
- Input: {}
- Output: {}
```

B Quality Evaluation Dimensions

In the selection module, the selection expert assigns an initial quality label to each sample based on five predefined dimensions. These dimensions are designed to evaluate the multifaceted quality of instruction-output pairs, ensuring that only high-quality samples contribute to the model's reasoning ability. The specific definitions for each dimension are listed in Table 6.

Terms	Definition
Correctness	Whether the content is accurate and grounded in reliable facts and data.
Completeness	Whether the content covers all necessary information, key points, and relevant details.
Clarity	Whether the expression is clear, concise, and free from ambiguity or redundancy.
Instruction Consistency	Whether the instruction specifies clear goals and constraints, and whether the output fully and accurately satisfies them.
Logical Reasoning	Whether the instruction requires a specific logical structure, and whether the output presents a coherent, sufficient, and reasonable reasoning process.

Table 6: Definitions of the five dimensions for data quality evaluation

C Instruction-Following Evaluation

To assess the generality of our framework, we further evaluate the optimized datasets on two downstream instruction-following benchmarks: IFEval (Zhou et al., 2023b) and AlpacaEval (Li et al., 2023). As shown in Table 7, the optimized data consistently improves all instruction-following metrics on both Alpaca and Dolly. In particular, On IFEval, the model improves by 13.30% (Strict) and 13.31% (Loose) on Alpaca, and by 18.85% (Strict) and 17.38% (Loose) on Dolly, while AlpacaEval improves by 4.25% and 3.22%, respectively. These results suggest that our framework enhances not only reasoning performance but also general instruction-following ability.

D Impact of Cluster Number and Data Partition Ratio

Within the Instruction Quality Differentiation (IQD) module, we introduce two pivotal hyperparameters: the number of semantic cluster centers (C) and the retention ratio of high-quality samples (K). Specifically, C controls the granularity of semantic partitioning, while K determines the strictness of high-quality sample retention. To identify suitable configurations for these parameters, we conduct a series of sensitivity experiments, with the results summarized in Table 8. For the hyperparameter of C, we systematically explored values ranging from 50 to 200. In particular, we include a dedicated setting at $C = 132$, motivated by the commonly adopted heuristic $C \approx \sqrt{n/2}$ (where n denotes the total number of samples) frequently cited in relevant literature. Meanwhile, we progressively adjust the retention ratio by varying the Top-K proportion from 20% to 90%. From the experimental results, we make the following observations: 1) Model performance first improves and then deteriorates as C increases, peaking precisely at the heuristic value of $C = 132$ (Avg 61.02%). This trend indicates that an excessively small C results in coarse clustering that conflates distinct task types, whereas an overly large C leads

Dataset	Data Size	IFEval (Strict)	IFEval (Loose)	AlpacaEval
Alpaca	52k	58.60	60.63	22.36
Alpaca_ours	35k	71.90 (+13.30)	73.94 (+13.31)	26.61 (+4.25)
Dolly	15k	48.06	50.46	18.76
Dolly_ours	12k	66.91 (+18.85)	67.84 (+17.38)	21.98 (+3.22)

Table 7: Instruction-following evaluation before and after optimization.

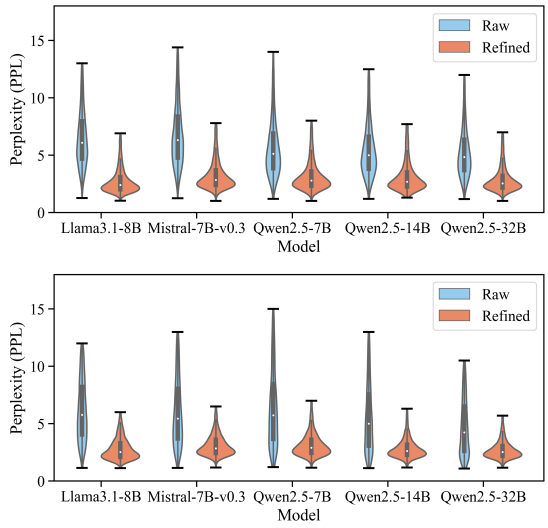

Figure 5: Perplexity distributions before and after data refinement. The upper plot shows results for Alpaca, and the lower for Dolly. Each plot includes five pairs of violins (blue: Raw, orange: Refined), with each pair representing the PPLs of a dataset before and after refinement on one backbone model.

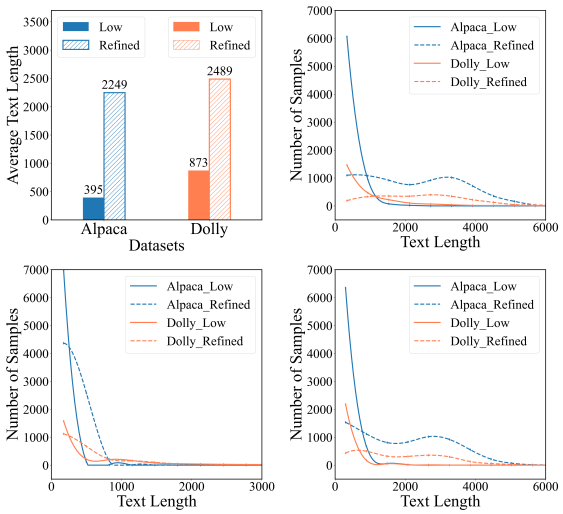

Figure 6: Statistics of text length in the data. The top-left panel shows the average text length of low-quality samples before and after refinement, while the top-right panel presents their overall length distributions. The bottom panels depict the length distributions of the instruction & input fields (bottom left) and the output field (bottom right). Solid lines indicate the raw data, and dashed lines indicate the refined versions.

to fragmented clusters that undermine the model's cross-topic generalization capabilities. 2) Similarly, performance varies non-monotonically with the retention ratio K, reaching its zenith at $K = 80\%$. This indicates that a restrictive K value leads to the inadvertent exclusion of valuable high-quality samples, while a permissive K value introduces noise by lowering the filtering threshold, thereby compromising optimization precision.

E Statistical Analysis of Datasets Pre- and Post-Optimization

To examine the intrinsic properties of data refined by our frameword, we perform a statistical analysis along three dimensions: perplexity (PPL), sequence length and semantic diversity, as shown in Figure 5, Figure 6 and Table 9.

Perplexity Analysis. We compute perplexity scores for datasets before and after rectification using multiple base language models, including Mistral-7B, Llama3.1-8B, and the Qwen2.5 series. As illustrated in Figure 5, the rectified data consistently exhibits lower PPL values across all evaluated architectures, accompanied by a more compact and concentrated distribution. The simultaneous reduction in both the mean and dispersion of PPL indicates that our framework effectively mitigates noisy or anomalous samples, thereby enhancing the overall data quality.

Length Analysis. We further analyze the changes in sequence length statistics before and after refinement. As illustrated in Figure 6, the average sequence length increases substantially after refinement (e.g., from 395 to 2249 on the Alpaca dataset). This transformation strongly demonstrates that our framework effectively enhances textual richness, by enriching previously information-sparse samples with essential contextual background and finer-grained logical structure, thereby ensuring a more comprehensive semantic

Hyperparameter	Understanding		Math & Code			Reasoning		Avg.
	C-Eval	ARC	HumanEval	GPQA	GSM8K	TriviaQA	Winogrande	
Proportion of High-Quality Data								
$C = 50$	52.53	85.51	42.08	28.94	66.26	67.03	62.27	57.80
$C = 80$	52.00	85.99	45.12	25.33	71.27	66.51	59.75	58.00
$C = 110$	51.26	85.96	47.56	23.74	77.03	66.82	63.06	59.35
$C = 132$	52.67	86.33	49.39	27.68	76.95	71.09	63.06	**61.02**
$C = 140$	51.78	86.11	46.95	20.13	70.74	66.51	63.06	57.90
$C = 170$	54.61	86.02	42.78	26.44	77.85	66.80	63.14	59.66
$C = 200$	53.42	85.97	41.46	22.14	75.66	67.09	62.83	58.37
Cluster Center Threshold								
$K = 20$	51.93	86.73	43.90	26.01	76.72	67.26	61.40	59.14
$K = 40$	54.09	86.33	41.46	24.83	79.53	66.53	60.22	59.00
$K = 60$	52.08	86.36	40.85	24.66	79.98	66.44	62.43	58.97
$K = 80$	52.67	86.33	49.39	27.68	76.95	71.09	63.06	**61.02**
$K = 90$	52.82	85.15	45.12	25.25	73.09	67.42	61.80	58.66

Table 8: Sensitivity analysis of the two hyperparameters, cluster number (C) and data partition ratio (K)%

Dataset	Data Size	APCS (%)	Δ APCS	Total Variance	Δ Variance
Alpaca	52k	58.53	–	2499.37	–
Alpaca (Optimized)	35k	56.78	-1.75	2671.23	+171.86
Dolly	15k	46.42	–	3245.61	–
Dolly (Optimized)	12k	45.57	-0.85	3343.09	+97.48

Table 9: Diversity statistics before and after optimization. Lower APCS indicates lower sample similarity, while higher total variance reflects broader semantic coverage.

representation.

Diversity Analysis. We examine the diversity of the refined data to ensure that the iterative process does not lead to a loss of linguistic diversity. On the Alpaca and Dolly datasets, we compute two diversity metrics before and after optimization: APCS (Average Pairwise Cosine Similarity (Ethayarajh, 2019) and Total Variance (Trace of Covariance) (Mu et al., 2017). Lower APCS indicates a more dispersed sample distribution, while higher Total Variance reflects broader coverage of the semantic space. The results are summarized in Table 9. After optimization, APCS decreases by 1.75% and 0.85% on Alpaca and Dolly, respectively, while Total Variance increases by 171.86 and 97.48. Overall, the optimized data exhibit lower mutual similarity and broader semantic coverage, indicating that the iterative refinement process does not induce a collapse in linguistic diversity and instead improves semantic diversity.

F Resource Consumption Analysis

To provide a comprehensive evaluation of the computational efficiency of our framework, we conduct a detailed analysis of the models employed, computational requirements, and actual wall-clock time for each module. These statistics are collected using the Alpaca dataset (52k samples) in an experimental environment equipped with three NVIDIA A100 GPUs. The results are summarized in Table 10, from which we draw two primary conclusions:

High offline processing efficiency. The total processing time for the entire dataset is approximately two hours. This indicates that our framework achieves high efficiency in offline data processing, enabling the cleaning and optimization of large-scale datasets with minimal time overhead, thereby demonstrating strong engineering practicality.

Manageable computational overhead. Most modules incur short execution times and negligible computational costs. The Feedback-driven Iterative Refinement (FIR) module is the most time-consuming component, as it involves a closed-loop process of multi-round text generation and expert evaluation. Nevertheless, given the substantial quality improvements it delivers, this additional computational cost is both justified and well controlled.

We provide a more detailed cost-benefit analysis

Method Component		Model	Hardware Requirements	Time
IQD Module	Initial Quality Labeling	Llama3.1-70B	2 x A100 GPUs	<10mins
	Cluster- & Difficulty-Aware Scoring	Llama3.1-8B	1 x A100 GPU	<20mins
FIR Module	Expert evaluation	Llama3.1-8B & Llama3.1-70B	3 x A100 GPUs	<20mins
	Instruction & Input Refinement	Llama3.1-70B	2 x A100 GPUs (Iter 3 times)	<60mins
	Expert Review	Llama3.1-8B & Llama3.1-70B	3 x A100 GPUs (Iter 3 times)	
OA Module	Output Refinement	Llama3.1-70B	2 x A100 GPUs	<25mins
Ours	Complete Pipeline	Llama3.1-8B & Llama3.1-70B	3 x A100 GPUs	135mins

Table 10: Resource consumption for each module within our framework.

Iteration Number	Time Cost	Accuracy Improvement	Improvement per Minute
$T = 1$	∼95 mins	5.48%	0.0577%
$T = 2$	∼120 mins	5.56%	0.0463%
$T = 3$	∼135 mins	6.69%	0.0496%
$T = 4$	∼147 mins	4.35%	0.0296%
$T = 5$	∼160 mins	3.60%	0.0225%

Table 11: Cost-benefit analysis under different iteration numbers on Alpaca.

on the Alpaca dataset. Using the average accuracy improvement across seven downstream tasks (relative to the raw dataset) as the benefit metric, the results are summarized in Table 11. The results show that $T = 1$ yields the highest gain per unit time. For $T = 2 \sim 3$, the return per unit time remains relatively high, while it drops substantially when $T \geq 4$. A similar trend is observed on the Dolly dataset, where the optimal range is also concentrated at $T \leq 3$, typically $2 \sim 3$ iterations. We therefore recommend $T = 2 \sim 3$ as a robust empirical range for different datasets.