Middo: Model-Informed Dynamic Data Optimization for Enhanced LLM Fine-Tuning via Closed-Loop Learning

Anonymous ACL submission

Abstract

Supervised Fine-Tuning (SFT) Large Language Models (LLM) fundamentally rely on high-quality training data. While data selection and data synthesis are two common strategies to improve data quality, existing approaches often face limitations in static dataset curation that fail to adapt to evolving model capabilities. In this paper, we introduce **Middo**, a self-evolving **M**odel-**i**nformed **d**ynamic **d**ata **o**ptimization framework that unifies model-aware data selection with context-preserving data refinement. Unlike conventional one-off filtering/synthesis methods, our framework establishes a closed-loop optimization system: (1) A self-referential diagnostic module proactively identifies suboptimal samples through tri-axial model signals - *loss patterns (complexity)*, *embedding cluster dynamics (diversity)*, and *self-alignment scores (quality)*; (2) An adaptive optimization engine then transforms suboptimal samples into pedagogically valuable training points while preserving semantic integrity; (3) This optimization process continuously evolves with model capability through curriculum learning principles. Experiments on multiple benchmarks demonstrate that our Middo consistently enhances the quality of seed data and boosts LLM's performance with improving accuracy by 7.15% on average while maintaining the original dataset scale. This work establishes a new paradigm for sustainable LLM training through dynamic human-AI co-evolution of data and models.

1 Introduction

Large Language Models (LLMs) have revolutionized artificial intelligence by achieving state-of-the-art performance across diverse domains, from natural language understanding (Zhou et al., 2023c; Hendrycks et al., 2021a) to mathematical reasoning (Cobbe et al., 2021; Hendrycks et al., 2021b) and code generation (Chen et al., 2021; Austin et al., 2021). This success is largely attributed to Supervised Fine-Tuning (SFT), where models undergo rigorous training on high-quality, human-aligned datasets to ensure outputs closely match human expectations. Crucially, the quality of these datasets directly dictates the model's ultimate capabilities: noisy or suboptimal training data can lead to degraded performance, while meticulously curated data unlocks advanced reasoning, generalization, and robustness. As LLMs scale, the adage "garbage in, garbage out" becomes increasingly important—highlighting the urgent need for systematic methods to optimize training data quality.

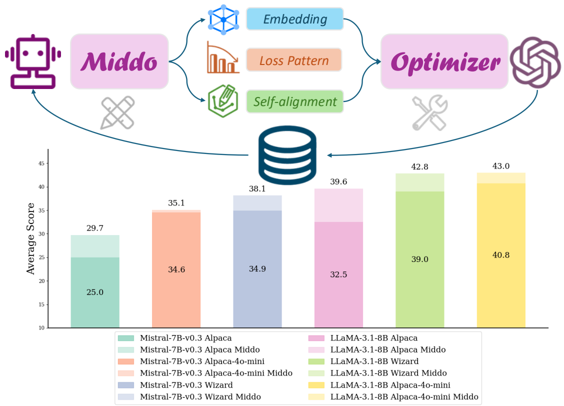


Figure 1: Comparison of different dataset and different models before and after Middo optimization.

Existing approaches primarily fall into two categories to improve data quality: data selection (Cao et al., 2024; Zhou et al., 2023b; Li et al., 2024d; Jia et al., 2024; Zhou et al., 2024; Li et al., 2024e,a) and data synthesis (Dai et al., 2023; WANG et al., 2023; Mukherjee et al., 2023; Xu et al., 2025; Liu et al., 2024a). Data selection methods filter raw datasets using heuristic rules (e.g., length filters) (Zhao et al., 2024a) or statistical metrics like perplexity (PPL) (Liu et al., 2024a) and Instruction-Following Difficulty (IFD) (Li et al., 2024c) to retain "high-quality" samples. Conversely, data synthesis leverages advanced LLMs (e.g., GPT-4 (Achiam et al., 2023)) to generate new training examples, often through prompting or distillation (Li et al., 2024f). While both strategies improve data quality, they suffer from critical limitations. Selection methods are typically static, applying fixed criteria that ignore the evolving needs of the model during training. Similarly, synthesis approaches often discard original data, wasting potentially valuable information, and risk generating distributional narrow or redundant examples. These one-time data curation methods fail to adaptively refine data in tandem with the model's progress.

To overcome these limitations, we propose **Middo**, Model-informed Dynamic Data Optimization, a self-evolving framework that unifies model-aware data selection with context-preserving data refinement. Unlike static approaches, Middo establishes a closed-loop optimization system where data curation dynamically adapts to the model's evolving capabilities. The framework operates through three core mechanisms: (1) A self-referential diagnostic module that proactively identifies suboptimal training samples using three model signals: *loss patterns* (to detect **complexity** mismatches between data and model proficiency), *embedding cluster dynamics* (to assess **diversity** gaps in the latent space), and *self-alignment scores* (to evaluate data **quality** against the model's own knowledge). (2) An adaptive optimization engine that transforms these suboptimal samples into pedagogically valuable training points. For example, overly complex samples may be simplified through stepwise decomposition, while low-diversity clusters are enriched with controlled extension—all while preserving the original data's semantic intent. (3) A dynamic curriculum that iteratively updates the training dataset based on the model's progress, ensuring that data difficulty and diversity scale with the model's capabilities. By integrating these components, Middo not only maximizes the utility of existing data but also bridges the gap between static data curation and adaptive model training.

Experiments across multiple benchmarks demonstrate Middo's effectiveness especially on low-quality datasets. Models trained with Middo optimized data achieve consistent performance gains over baselines, improving accuracy by 7.15% on average while maintaining the original dataset scale. Further analysis highlights Middo's ability to "grow" with the model: early training phases focus on complex, high-loss examples, whereas later stages prioritize diversity extension. Notably, Middo trained models exhibit stronger abilities to addresses hard problems, solving more than three times the number of challenging test problems (e.g., MATH, GPQA) compared to models trained on static datasets. These results validate that sustainable LLM advancement requires co-evolving data and models—a paradigm shift from today's disjointed curation practices.

2 Related Work

2.1 Synthetic Data Generation

Synthetic data generation is a key technique for augmenting LLM fine-tuning. Early methods (Edunov et al., 2018; Wieting and Gimpel, 2018) introduced perturbation-based approaches to enhance data diversity, using character-level (Belinkov and Bisk, 2018) and word-level (Wei and Zou, 2019) modifications. These methods relied on fixed transformation rules, limiting adaptability.

LLMs have been leveraged for scalable data synthesis (Sudalairaj et al., 2024; Jung et al., 2024; Dai et al., 2023; WANG et al., 2023; Mukherjee et al., 2023; Xu et al., 2025; Liu et al., 2024a; Li et al., 2025). Self-instruct methods (Wang et al., 2023) generate instruction-response pairs, while Evol-Instruct (Xu et al., 2024) and Auto-Evol-Instruct (Zeng et al., 2024) refine data complexity iteratively. However, these methods remain static, failing to adapt as models improve. Recent approaches integrate model feedback into data generation (Anonymous, 2025a; Cao et al., 2025; Anonymous, 2025b; Li et al., 2024f), incorporating student model signals for adaptive synthesis. LLM2LLM (Lee et al., 2024) is an iterative data augmentation strategy that enhances low-data fine-tuning by using a teacher LLM to generate synthetic training data from incorrect student LLM predictions and I-SHEEP (Park et al., 2024) uses iterative self-enhancement paradigm.

2.2 Data Selection

Data selection is crucial for LLM fine-tuning, as high-quality and informative data directly impacts model performance (Zhou et al., 2023a; Xu et al., 2023). Early heuristic-based methods relied on surface-level statistics like item frequency (Raffel et al., 2020) and repetition count (Laurençon et al., 2022), but lacked adaptability to model evolution.

Recent work explores LLM-driven data selection, optimizing for quality, diversity, and complexity (Cao et al., 2024; Zhou et al., 2023b; Li et al., 2024d; Jia et al., 2024; Zhou et al., 2024; Li et al., 2024e,a; Du et al., 2023; Kung et al., 2023). Instruction-Following Difficulty (IFD) metric (Li et al., 2024c) enables models to self-select training instances, while other methods (Yu et al., 2024; Colombo et al., 2024; Lu et al., 2024; Zhao et al., 2024b) use LLM self-assessment for efficiency. Further advancements integrate LLM-based evaluation mechanisms. AlpaGasus (Chen et al., 2024) and LIFT (Xu et al., 2023) use structured prompts for data assessment, while DEITA (Liu et al., 2024b) introduces a multi-dimensional scoring system based on complexity and quality.

3 Methodology

This section delineates the methodological framework of Middo (illustrated in Figure 2), our proposed self-evolving data optimization system for LLM fine-tuning. The methodology is structured into four interconnected components, an overview of the closed-loop and iterative dynamic optimization pipeline, followed by introducing detailed expositions of its three core modules.

3.1 Middo Pipeline

As depicted in Figure 2, our Middo framework establishes an iterative data-model co-evolution loop driven by tri-axial signal analysis, along with three interconnected data optimization mechanisms, each targeting distinct dimensions of training sample selection: (1) *Loss patterns*, to identify samples with mismatched complexity (overly challenging) relative to the current model's capability through loss trajectory analysis. (2) *Embedding cluster dynamics*, to detect coverage gaps in the semantic space, ensuring balanced conceptual representation. (3) *Self-alignment scores*, for quality filtering to leverage the model's self-evaluation capacity to flag low-confidence or inconsistent responses through automated alignment scoring.

At each iteration, these parallel signal analyzers jointly select suboptimal samples, which are then regenerated through context-aware synthesis—preserving original semantic intent while enhancing pedagogical value. The refined dataset immediately feeds back into model training, creating a dynamic feedback loop where improved model capabilities inform subsequent optimization cycles. Notably, the optimized dataset remains similar data size, without extending large data synthesis, leading an efficient data optimization. This self-referential mechanism ensures continuous alignment between data characteristics and model evolution. Following sections systematically elaborate on the implementation of each signal-specific optimization module and their synergistic integration.

3.2 *Loss Patterns*: Complexity Optimization

Complexity Selection. Complexity reflects the "difficulty" or "compositionality" of data. A good dataset usually requires smooth complexity distribution of data for training. Therefore, we introduce *Loss Patterns*, which targets overly challenging samples by modifying them to maintain a balanced and learnable training set (Zhao et al., 2024b). During fine-tuning, the loss for a sample (X_i, Y_i) is computed as the likelihood of predicting successive tokens given the instruction X_i and its context. We denote the loss before and after training as $\mathcal{L}_{\text{pre}}(X_i, Y_i)$ and $\mathcal{L}_{\text{post}}(X_i, Y_i)$, respectively.

Intuitively, we consider both the loss before and after training to select the complex data. Specifically, we classify samples based on their loss evolution: samples with both low $\mathcal{L}_{\text{pre}}$ and $\mathcal{L}_{\text{post}}$ are considered easy, while those with high values in both metrics remain difficult, indicating excessive complexity. We then define a complex sample subset, $\mathcal{D}^{\text{hard}}$, as those where both losses exceed thresholds τ_{pre} and τ_{post}. For adaptive refinement, these thresholds are dynamically computed as $\tau_l = \mu + m\sigma$, with μ and σ being the mean and standard deviation of the loss values across the dataset.

Complexity Optimization. For complex data optimization, instead of discarding difficult samples, we transform them to be simpler with more manageable forms. Specifically, we replace samples in $\mathcal{D}^{\text{hard}}$ with their simplified counterparts, $\mathcal{D}^{\text{hard}'}$. This is achieved by an automatic process in which a LLM analyzes and summarizes the complex instructions (Zeng et al., 2024), then simplifies them step by step while preserving the core educational content. An example is shown as Figure 9 in Appendix. This iterative transformation process updates the dataset by replacing overly complex samples with refined versions that offer more effective learning material. As training continues, this adaptive approach ensures a continuous alignment between data complexity and model capability.

3.3 *Embedding Cluster Dynamics*: Diversity Optimization

Diversity Selection. Diversity is crucial for ensuring broad concept coverage and a uniform data

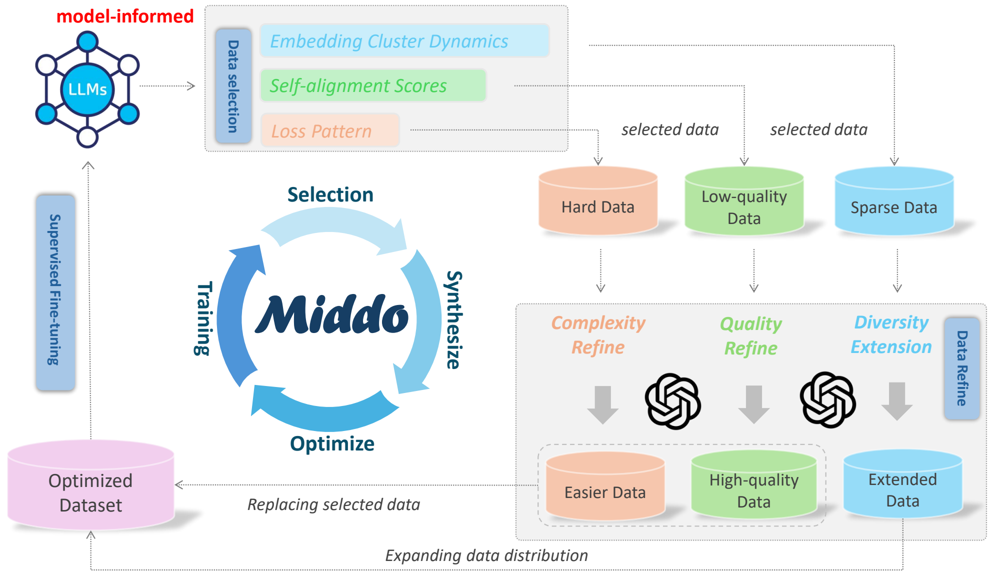


Figure 2: The Middo pipeline: a closed-loop, iterative dynamic optimization framework for LLM fine-tuning. It comprises three core modules that leverage model feedback: *Loss Patterns* identify overly complex samples, which are then simplified; *Self-alignment Scores* evaluate data quality, transforming low-quality samples into high-quality ones; and *Embedding Cluster Dynamics* detect sparse data points and expand the data distribution through targeted augmentation. Middo ensure the training set continually evolves to better align with the model's capabilities.

distribution. *Embedding Cluster Dynamics* identifies sparse data points that signal underrepresented regions in the dataset. We use the embedding cluster dynamics to select data as follows: We extract sentence embeddings from the model's (which is the initial model that we will train upon it) last hidden layer $\mathcal{H}^{(L)}$ using average pooling, then compute the cosine similarity between each data point and find the k-nearest neighbors for each data, $\mathcal{N}_k(X_i)$. A lower average cosine similarity among these neighbors $\mathcal{N}_k(X_i)$ indicates the data is positioned in a sparser region. Thus, the data points whose average cosine similarity score (diversity score) below a threshold τ_e are selected for optimization. Similarly, the threshold τ_e is also dynamically updated as iteration goes on.

Diversity Optimization. To enhance diversity-balanced distribution, we augment the sparse subset $\mathcal{D}^{\text{sparse}}$ by incorporating examples from their corresponding $\mathcal{N}_k(X_i)$ as demonstrations to generate new samples. This process generates an expanded set $\mathcal{D}^{\text{sparse}'}$, which is then integrated back into the dataset. An instance can be found in Appendix (Figure 10). By following a structured augmentation strategy that preserves semantic relevance, we ensure that the data distribution becomes both broader and more balanced, ultimately improving the model's generalization.

3.4 *Self-alignment Scores*: Quality Optimization

Quality Selection. High-quality data is essential for fine-tuning, as poor-quality samples can degrade performance (Zhou et al., 2023a). To reduce manual annotation costs, many approaches use the LLM-as-a-Judge paradigm (Chen et al., 2024; Xu et al., 2023). We enhance this by leveraging the fine-tuned model itself to assess data quality via *Self-alignment Scores*. Specifically, for each instruction-response pair (X_i, Y_i) in the dataset $\mathcal{D}$, the model computes scores $\mathcal{S}\pi^{\text{ins.}}(X_i)$ for instruction evaluation and $\mathcal{S}\pi^{\text{res.}}(X_i, Y_i)$ for instruction-response pair evaluation based on three key metrics π from AlignBench (Liu et al., 2024c): *Clarity*, *Completeness*, and *Factuality*. The final quality score $\mathcal{S}(X_i, Y_i)$ is obtained by averaging these individual scores. These samples with scores below a similar dynamic threshold τ_s are identified as low-quality, forming the seed dataset $\mathcal{D}^{\text{low}}$.

Quality Optimization. To refine $\mathcal{D}^{\text{low}}$, we use LLMs to automatically analyze and improve these samples via tailored evolution strategies (prompt templates and examples are provided in the Appendix Figure 11). This process converts low-quality samples into higher-quality versions, denoted as $\mathcal{D}^{\text{low}'}$. The dataset is then updated by replacing the original low-quality samples with $\mathcal{D}^{\text{low}'}$, maintaining the dataset size while progressively enhancing its overall quality.

In each iteration, after the above three data selection and refinement progress, the optimized dataset is then feed back for next round model training.

4 Experiment

4.1 Settings

Data Optimization Configurations. We conduct optimization on the Alpaca (Taori et al., 2023) and WizardLM (Xu et al., 2024) datasets. To ensure a fair comparison, we also completely rewrite the responses of Alpaca using GPT-4o-mini as seed data for the experiments. For *Embedding Cluster Dynamics*, we set the number of neighbors to $k = 2$. During experiments, each dataset undergoes three iterations of optimization. A detailed analysis of the number of neighbors and iteration numbers are provided in Appendix B and Appendix C. We generate data by GPT-4o-mini, setting both temperature and top_p to 1.0 to maximize diversity.

Training and Evaluation Settings. We finetune LLaMA-3.1-8B (Dubey et al., 2024) and Mistral-7B-v0.3 (Jiang et al., 2023) using LLaMA-Factory (Zheng et al., 2024), performing full-parameter SFT for one epoch. Evaluation is conducted using OpenCompass (Contributors, 2023), with vLLM (Kwon et al., 2023) for batch inference acceleration. All hyperparameters are detailed in Appendix A.5. We assess general knowledge on IFEval (Zhou et al., 2023c) and MMLU (Hendrycks et al., 2021a), mathematical problem-solving on GSM8K (Cobbe et al., 2021) and MATH (Hendrycks et al., 2021b), code generation on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), and commonsense reasoning on Hellaswag (Zellers et al., 2019) and GPQA (Rein et al., 2024).

4.2 Main Results

The evaluation results on all benchmarks over various data iterations and models are presented in Table 1. We can see that Middo consistently enhances model performance across all benchmarks, achieving an average accuracy increase of 7.15% over three iterations on the Alpaca dataset based on LLaMA-3.1-8B, all while preserving the original data scale. Moreover, when extending our experiments to Mistral-7B-v0.3, we observed an average improvement of 4.75%, further underscoring the robustness and adaptability of our framework across different model architectures.

Dynamic Iterative Improvement. On the Alpaca dataset, the average score increased progressively with each iteration. Across the MMLU, GSM8K, MATH, and MBPP benchmarks, we observed consistent, step-by-step improvements over multiple iterations. This showcases the versatility of our approach, which excels in general capabilities, mathematics, and coding. Notably, accuracy on GSM8K improved by 15.55%, and Hellaswag saw an 11.11% increase when evaluated on the LLaMA-3.1-8B model. For Mistral-7B-v0.3, we observed an 11.07% improvement on MMLU, a 12.59% increase on GSM8K, and a 10.6% gain on GPQA. These results underscore the effectiveness of our method in driving performance gains and highlight the cumulative benefit of our iterative optimization process.

Further Validation on 4o-mini Rewritten Data. Steady improvements observed on both the 4o-mini rewritten Alpaca dataset—averaging a 2.2% increase overall, with MMLU showing an impressive 11.87% boost—demonstrate that these gains are not merely a result of using 4o-mini data. Instead, they illustrate that our framework intrinsically enhances dataset quality and model performance. Importantly, we achieve these improvements without resorting to stronger variants such as GPT-4o (Hurst et al., 2024) or GPT-o1 (Jaech et al., 2024), reinforcing the robustness and general applicability of our method.

Initial Dataset Quality. Our experiments reveal that higher-quality datasets require fewer modifications to reach optimal performance. For instance, while the Alpaca dataset achieve peak performance at third iterations, the 4o-mini rewritten Alpaca required only two iterations, and the Wizard dataset reached its best performance in just one round of optimization.

Comparison with Other Works. We compare Middo with both existing data selection (Alpaca-clean(Ruebsamen, 2023), Superfiltering (Li et al., 2024b), Long (Zhao et al., 2024a), AlpaGasus (Chen et al., 2024)) and data augmentation (Alpaca-GPT4 (Peng et al., 2023), I-SHEEP (Anonymous, 2025b), WizardLM (Xu et al., 2024)) methods on the Alpaca dataset.

We use the optimal dataset obtained through Middo from Alpaca for comparison with other baselines. Additionally, to ensure a relatively fair comparison with data selection methods, we include a

Setting		General		Math		Code		Reasoning		Average
		MMLU	IFEval	GSM8K	MATH	HumanEval	MBPP	Hellaswag	GPQA	
Base Model: LLaMA-3.1-8B										
Alpaca	*init*	47.46	41.09	35.63	4.96	39.63	37.40	48.11	5.56	32.48
	iter1	50.13	45.77	43.67	10.62	40.24	39.20	56.37	13.64	37.45
	iter2	41.82	44.63	50.11	12.40	39.63	41.40	59.22	18.18	<u>38.42</u>
	iter3	51.32	43.20	51.18	12.92	39.63	41.80	58.78	16.67	**39.63**
Alpaca 4o-mini	*init*	32.82	44.04	57.09	17.78	51.22	45.20	53.70	24.24	40.76
	iter1	41.09	43.47	54.21	17.34	51.22	46.00	59.11	21.72	41.77
	iter2	44.69	47.96	57.62	18.50	52.44	45.40	57.37	19.70	**42.96**
	iter3	38.58	48.11	58.68	18.30	46.95	46.80	52.37	28.79	<u>42.32</u>
Wizard	*init*	46.12	46.14	53.30	12.72	40.24	48.00	53.05	12.12	38.96
	iter1	48.39	50.11	54.44	13.80	46.95	45.00	63.54	20.20	**42.80**
	iter2	48.86	49.48	55.12	13.90	48.78	45.20	58.63	18.18	<u>42.29</u>
	iter3	47.18	50.79	54.51	11.70	43.29	45.40	62.97	20.20	42.01
Base Model: Mistral-7B-v0.3										
Alpaca	*init*	27.66	43.22	22.21	3.88	29.27	28.80	44.17	0.51	24.97
	iter1	31.31	45.62	29.57	5.82	30.49	33.80	42.73	14.65	29.25
	iter2	26.87	49.46	31.69	6.84	31.71	31.00	53.95	5.56	<u>29.64</u>
	iter3	38.73	44.01	34.80	6.64	26.22	31.40	44.86	11.11	**29.72**
Alpaca 4o-mini	*init*	31.56	43.14	44.88	9.64	42.07	37.80	46.25	21.21	<u>34.56</u>
	iter1	31.33	47.93	45.19	8.72	37.20	41.32	41.32	19.70	34.09
	iter2	28.83	47.92	48.90	11.34	35.37	38.40	42.63	27.27	**35.08**
	iter3	28.96	50.78	48.60	10.10	32.32	39.00	32.95	20.20	32.86
Wizard	*init*	40.71	50.95	44.96	8.10	35.98	35.60	53.98	9.09	34.92
	iter1	41.39	51.18	44.43	9.44	37.80	38.60	59.01	17.17	<u>37.38</u>
	iter2	33.87	51.71	47.08	9.26	39.02	38.40	66.18	19.7	**38.15**
	iter3	33.18	50.79	41.51	9.70	36.29	37.40	54.97	16.20	35.01

Table 1: Main Results: performance of models in the optimization step for different datasets and the performance trained on optimized dataset. The *init* means the performance of the model before the iterative optimization step. The best performance for the average is highlighted in **bold** and the second best is <u>underlined</u>.

Method	Size	General		Math		Code		Reasoning		Average
		MMLU	IFEval	GSM8K	MATH	HumanEval	MBPP	Hellaswag	GPQA	
Alpaca	52.0k	47.46	41.09	35.63	4.96	39.63	37.40	48.11	5.56	32.48
Data Selection										
Alpaca-clean	51.7k	47.21	43.92	43.90	4.20	29.27	43.40	60.17	5.56	34.70
Superfiltering	7.8k	39.96	37.80	44.50	5.38	40.85	44.00	42.38	27.27	35.27
Superfiltering GPT4	7.8k	37.71	34.35	53.68	11.00	9.15	45.60	57.81	2.53	31.48
Long	1.0k	25.51	14.75	56.33	16.56	13.41	45.60	25.83	0.00*	24.75
AlpaGasus	9.2k	33.98	48.82	43.82	6.06	35.98	42.40	44.50	18.18	34.22
Data Augmentation										
I-SHEEP	8.4k	23.61	29.61	43.14	8.28	32.32	32.60	41.83	0.00*	26.42
Alpaca-GPT4	5.2k	51.94	38.68	50.87	10.28	17.07	43.60	63.02	0.51	34.50
WizardLM	70.0k	46.12	46.14	53.30	12.72	40.24	48.00	53.05	12.12	38.96
Middo Optimize Only	8.8k	43.47	40.78	65.20	15.58	51.83	47.60	58.65	17.68	<u>42.60</u>
Middo	63.1k	44.69	47.96	57.62	18.50	52.44	45.40	57.37	19.70	**42.96**

Table 2: Results of Middo compared to other baseline methods. The best and second best results are highlighted in **bold** and <u>underlined</u>, respectively. The size of the dataset is the number of examples used for training. Our method outperforms all baselines in the average score. *Note that 0.00 indicates that the method did not solve any examples.

dataset that only uses the optimized data without incorporating any unoptimized samples, referred as “Middo Optimize Only”, to isolate the effect of the optimization process and make a direct comparison with data selection approaches.

Results in Table 2 shows our method achieves

iteration	Ablations	IFEval	MATH	HumanEval	Hellaswag	Average
iter1	w	45.77	10.62	40.24	56.37	**38.25**
	w/o loss	42.49	10.11	39.02	59.53	37.79
	w/o neighbor	39.01	10.82	42.07	57.86	37.45
	w/o score	43.48	10.20	36.59	48.40	34.67
iter2	w	44.63	12.40	39.63	59.22	**38.97**
	w/o loss	42.28	9.92	42.68	58.21	38.27
	w/o neighbor	46.75	10.26	34.76	46.66	34.61
	w/o score	44.18	11.76	39.02	51.38	36.58
iter3	w	44.24	12.92	39.63	59.25	**39.01**
	w/o loss	43.18	12.42	36.59	55.30	36.87
	w/o neighbor	40.12	12.46	34.15	56.83	35.89
	w/o score	45.17	7.92	40.85	54.67	37.15

Table 3: Ablation study on the development set. We report the performance of the model with different ablations. The ablations include removing the *loss patterns*, *embedding cluster dynamics* and *self-alignment scores* separately. The best performance is highlighted in **bold**.

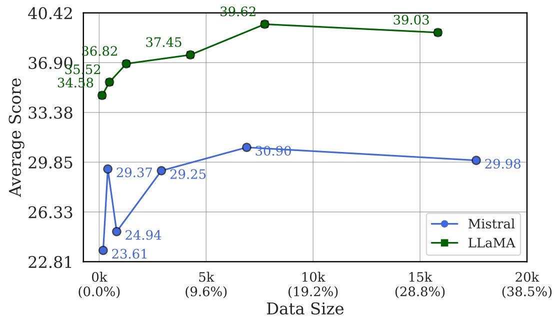


Figure 3: Performance comparison of Middo on the Alpaca dataset with varying refined data sizes. The x-axis represents the number and percentage of data selected for refinement, while the y-axis shows the average accuracy across three iterations. To ensure fairness, we guarantee that the data after refinement is the same.

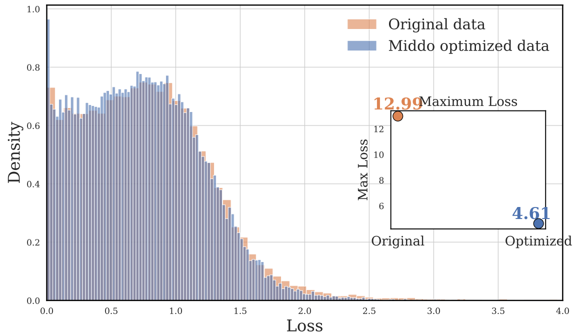


Figure 4: Loss distribution comparison before and after applying Middo. We tested LLaMA-3.1-8B on the Alpaca dataset over one optimization iteration. The density curve reflects the relative frequency of data points within specific loss intervals. The inset subfigure highlights the maximum loss reduction from 12.99 to 4.61 through our optimization.

the highest average score of 42.96, outperforming all other approaches. Notably, even when using only the optimized subset *Middo Optimize Only*, our method delivers a robust average score of 42.6. This demonstrates that data size is not the main factor influencing iterative improvement, but are inherent to the effectiveness of our dynamic data selection and optimization process.

5 Analysis

5.1 Ablation Studies

To assess the effectiveness of Middo and the contribution of each optimization pipeline, we conduct ablation experiments on the LLaMA-3.1-8B model over the Alpaca dataset. Specifically, we analyze the following ablations: **(a) w/o loss**: removes *Loss Patterns*. **(b) w/o neighbor**: excludes *Embedding Cluster Dynamics*. **(c) w/o score**: removes *Self-alignment Scores*.

The ablation results in Table 3 consistently show that removing any part of the framework leads to a decline in performance across multiple iterations, reinforcing that each component plays a significant role in the overall performance. This trend holds across the second (*iter2*) and third (*iter3*) iterations, where the removal of any pipeline consistently results in suboptimal performance, further highlighting the importance of balancing complexity, diversity, and quality in the optimization process. These findings underscore the necessity of the full framework for achieving optimal results.

5.2 Effect of Selected Data Scale

We give an investigation on the impact of the different scales of the selected and optimized data in this section by varying the thresholds for data selection. Results are illustrated in Figure 3. We observe that increasing the size of the refined data initially leads to an upward trend in performance; however, once the refined data exceeds a certain threshold, performance begins to decline. To maintain the potential for further iterative improvement, we set the refined data size at a moderate level that optimally balances the cost and benefit of the optimization process. In the first iteration, each component selects approximately 5% of the data for refinement. By controlling the parameter m, the amount of data refined can adaptively change as the model's capability increases. Detailed data sizes selected in each iteration are provided in Appendix F.

5.3 Data Analysis

For a deep understanding of how Middo transforms the dataset, we provide an analysis of its impact on data complexity, diversity, and quality.

Complexity. To quantify how Middo modulates dataset complexity, we analyze the loss distribution evolution through optimization cycles. As shown

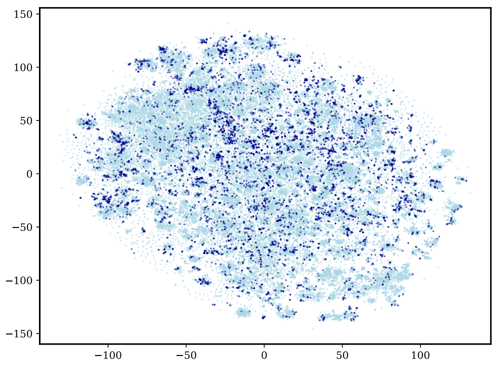


Figure 5: t-SNE visualization of the Alpaca dataset before and after applying Middo. The original dataset is shown in light blue, while the augmented data is in dark blue. The plot illustrates the distribution of data points in the latent space, highlighting the impact of Middo on dataset diversity.

in Figure 4, the original dataset exhibits a long-tailed distribution with extreme loss values up to 12.99. After applying Middo, the maximum loss decreases by 64.5% to 4.61, indicating successful mitigation of overly complex samples and the distribution mode shifts leftward, suggesting better alignment between data complexity and model capability. This transformation demonstrates our framework's ability to adaptively prune pathological samples while preserving pedagogically valuable challenges.

Diversity. To analyze the diversity of the dataset after applying Middo, we visualize the data distribution using t-SNE. As shown in Figure 5. This visualization reveals how the augmented data points are distributed relative to the original data. Notably, most of the augmented samples are located at the peripheries of the clusters, effectively filling in the sparsely populated regions. This distribution indicates that Middo is not merely adding redundant data but is instead enhancing the overall coverage of the latent space. By strategically augmenting the dataset at the cluster edges, Middo improves the diversity and ensures a more uniform distribution of data points, ultimately contributing to better model generalization.

Quality. The self-alignment score trajectories across different iterations are presented in Figure 6. The observed trend indicates a gradual increase in the average score as the iterations progress. This improvement signifies that the quality of the data is becoming more closely aligned with the model's evolving capabilities. Through the adversarial self-play mechanisms and iterative quality refinement, the model is able to assess and enhance the quality

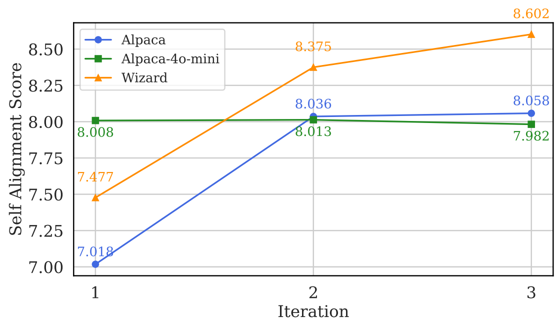


Figure 6: Self-alignment score evolution across iterations. The x-axis represents the number of iterations, while the y-axis shows the average self-alignment score.

of both the instructions and responses within the dataset. As the self-alignment scores increase, it reflects that the refined data is not only more accurate but also more consistent with the model's internal standards and expectations. This detailed evolution of the self-alignment scores provides critical insights into the dynamic process of dataset optimization, confirming that our approach effectively transforms low-quality samples into high-quality learning material over successive iterations.

6 Conclusion

In this paper, we present Middo, a model-informed dynamic data optimization framework that transforms LLM fine-tuning via closed-loop learning. Unlike traditional static methods, Middo establishes a self-evolving system that continuously adapts to the model's evolving capabilities. It employs three core mechanisms: complexity optimization refines overly complex samples using *loss patterns*, ensuring the training data remains appropriately challenging; diversity optimization enhances dataset diversity by analyzing *embedding cluster dynamics* and addressing gaps in the semantic space; and quality optimization leverages *self-alignment scores* to evaluate and improve the quality of training samples. Experiments on multiple benchmarks demonstrate that Middoconsistently boosts LLMs' performance, achieving an average accuracy improvement of 7.15% while maintaining the original data scale on LLaMA-3.1-8B. Ablation studies confirm the effectiveness of each component, underscoring the importance of balancing complexity, diversity, and quality. Middo's adaptability and model-awareness make it a powerful tool for sustainable LLM training, ushering in a new paradigm of dynamic human-AI co-evolution of data and models. Moreover, our approach paves the way for future research in adaptive training that continuously optimize learning efficiency.

Limitations

Despite its promising results, Middo has several limitations: (1) Middo relies on the model being fine-tuned itself for identifying data quality and complexity. This means that the approach requires a sufficiently capable base model, and the performance may be limited if the base model is not strong enough to generate meaningful diagnostics for data refinement. (2) Middo does not currently utilize Reinforcement Learning from Human Feedback (RLHF), which could further enhance data refinement, especially for complex or subjective tasks. (3) The closed-loop optimization system may lead to higher computational costs as the dataset grows or updates become more frequent, presenting scalability challenges. (4) Middo may propagate biases present in the initial training data, limiting fairness and generalization if the base model is trained on biased data. These limitations highlight areas for future improvement, such as integrating RLHF, optimizing for scalability, and addressing data biases.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anonymous. 2025a. Dataenvgym: Data generation agents in teacher environments with student feedback. In *The Thirteenth International Conference on Learning Representations*.

Anonymous. 2025b. I-SHEEP: Self-alignment of LLM from scratch through an iterative self-enhancement paradigm. In *Submitted to ACL Rolling Review - December 2024*. Under review.

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. *Preprint*, arXiv:2108.07732.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *International Conference on Learning Representations*.

Maosong Cao, Taolin Zhang, Mo Li, Chuyu Zhang, Yunxin Liu, Haodong Duan, Songyang Zhang, and Kai Chen. 2025. Condor: Enhance llm alignment with knowledge-driven data synthesis and refinement. *Preprint*, arXiv:2501.12273.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2024. Instruction mining: Instruction data selection for tuning large language models. In *First Conference on Language Modeling*.

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

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. *Preprint*, arXiv:2107.03374.

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

Pierre Colombo, Telmo Pessoa Pires, Malik Boudiaf, Dominic Culver, Rui Melo, Caio Corro, Andre F. T. Martins, Fabrizio Esposito, Vera Lúcia Raposo, Sofia Morgado, and Michael Desa. 2024. Saullm-7b: A pioneering large language model for law. *Preprint*, arXiv:2403.03883.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. `https://github.com/open-compass/opencompass`.

Haixing Dai, Zhengliang Liu, Wenxiong Liao, Xiaoke Huang, Yihan Cao, Zihao Wu, Lin Zhao, Shaochen Xu, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Lichao Sun, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. 2023. Auggpt: Leveraging chatgpt for text data augmentation. *Preprint*, arXiv:2302.13007.

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

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding back-translation at scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. *arXiv preprint arXiv:2412.16720*.

Qi Jia, Siyu Ren, Ziheng Qin, Fuzhao Xue, Jinjie Ni, and Yang You. 2024. Boosting llm via learning from data iteratively and selectively. *Preprint*, arXiv:2412.17365.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *Preprint*, arXiv:2310.06825.

Jaehun Jung, Peter West, Liwei Jiang, Faeze Brahman, Ximing Lu, Jillian Fisher, Taylor Sorensen, and Yejin Choi. 2024. Impossible distillation for paraphrasing and summarization: How to make high-quality lemonade out of small, low-quality model. 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 4439–4454, Mexico City, Mexico. Association for Computational Linguistics.

Po-Nien Kung, Fan Yin, Di Wu, Kai-Wei Chang, and Nanyun Peng. 2023. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 611–626, New York, NY, USA. Association for Computing Machinery.

Hugo Laurençon, Lucile Saulnier, Thomas Wang, Christopher Akiki, Albert Villanova del Moral, Teven Le Scao, Leandro Von Werra, Chenghao Mou, Eduardo González Ponferrada, Huu Nguyen, Jörg Frohberg, Mario Šaško, Quentin Lhoest, Angelina McMillan-Major, Gérard Dupont, Stella Biderman, Anna Rogers, Loubna Ben allal, Francesco De Toni, Giada Pistilli, Olivier Nguyen, Somaieh Nikpoor, Maraim Masoud, Pierre Colombo, Javier de la Rosa, Paulo Villegas, Tristan Thrush, Shayne Longpre, Sebastian Nagel, Leon Weber, Manuel Romero Muñoz, Jian Zhu, Daniel Van Strien, Zaid Alyafeai, Khalid Almubarak, Vu Minh Chien, Itziar Gonzalez-Dios, Aitor Soroa, Kyle Lo, Manan Dey, Pedro Ortiz Suarez, Aaron Gokaslan, Shamik Bose, David Ifeoluwa Adelani, Long Phan, Hieu Tran, Ian Yu, Suhas Pai, Jenny Chim, Violette Lepercq, Suzana Ilic, Margaret Mitchell, Sasha Luccioni, and Yacine Jernite. 2022. The bigscience ROOTS corpus: A 1.6TB composite multilingual dataset. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Nicholas Lee, Thanakul Wattanawong, Sehoon Kim, Karttikeya Mangalam, Sheng Shen, Gopala Anumanchipalli, Michael Mahoney, Kurt Keutzer, and Amir Gholami. 2024. LLM2LLM: Boosting LLMs with novel iterative data enhancement. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 6498–6526, Bangkok, Thailand. Association for Computational Linguistics.

Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, Yuxian Gu, Xin Cheng, Xun Wang, Si-Qing Chen, Li Dong, Wei Lu, Zhifang Sui, Benyou Wang, Wai Lam, and Furu Wei. 2025. Synthetic data (almost) from scratch: Generalized instruction tuning for language models.

Ming Li, Lichang Chen, Jiuhai Chen, Shwai He, Jiuxiang Gu, and Tianyi Zhou. 2024a. Selective reflection-tuning: Student-selected data recycling for LLM instruction-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 16189–16211, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yong Zhang, Shwai He, Zhitao Li, Hongyu Zhao, Jianzong Wang, Ning Cheng, and Tianyi Zhou. 2024b. 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, Bangkok, Thailand. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024c. 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.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. 2024d. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*.

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. 2024e. One-shot learning as instruction data prospector for large language models. *Preprint*, arXiv:2312.10302.

Zhuochun Li, Yuelyu Ji, Rui Meng, and Daqing He. 2024f. Learning from committee: Reasoning distillation from a mixture of teachers with peer-review. *Preprint*, arXiv:2410.03663.

Ruibo Liu, Jerry Wei, Fangyu Liu, Chenglei Si, Yanzhe Zhang, Jinmeng Rao, Steven Zheng, Daiyi Peng, Diyi Yang, Denny Zhou, and Andrew M. Dai. 2024a. Best practices and lessons learned on synthetic data. In *First Conference on Language Modeling*.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024b. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Andrew Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, Xiaohan Zhang, Lichao Sun, Xiaotao Gu, Hongning Wang, Jing Zhang, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024c. AlignBench: Benchmarking Chinese alignment of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11621–11640, Bangkok, Thailand. Association for Computational Linguistics.

Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2024. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.

Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *Preprint*, arXiv:2306.02707.

Chansung Park, Juyong Jiang, Fan Wang, Sayak Paul, and Jing Tang. 2024. Llamaduo: Llmops pipeline for seamless migration from service llms to small-scale local llms. *Preprint*, arXiv:2408.13467.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

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*.

Gene Ruebsamen. 2023. GitHub - gururise/AlpacaDataCleaned: Alpaca dataset from Stanford, cleaned and curated — github.com. `https://github.com/gururise/AlpacaDataCleaned`.

Shivchander Sudalairaj, Abhishek Bhandwaldar, Aldo Pareja, Kai Xu, David D. Cox, and Akash Srivastava. 2024. Lab: Large-scale alignment for chatbots. *Preprint*, arXiv:2403.01081.

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`.

Ruida WANG, Wangchunshu Zhou, and Mrinmaya Sachan. 2023. Let's synthesize step by step: Iterative dataset synthesis with large language models by extrapolating errors from small models. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6382–6388, Hong Kong, China. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

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*.

Yang Xu, Yongqiang Yao, Yufan Huang, Mengnan Qi, Maoquan Wang, Bin Gu, and Neel Sundaresan. 2023. Rethinking the instruction quality: Lift is what you need. *Preprint*, arXiv:2312.11508.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2025. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In *The Thirteenth International Conference on Learning Representations*.

Zhaojian Yu, Xin Zhang, Ning Shang, Yangyu Huang, Can Xu, Yishujie Zhao, Wenxiang Hu, and Qiufeng Yin. 2024. Wavecoder: Widespread and versatile enhancement for code large language models by instruction tuning. *Preprint*, arXiv:2312.14187.

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*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Weihao Zeng, Can Xu, Yingxiu Zhao, Jian-Guang Lou, and Weizhu Chen. 2024. Automatic instruction evolving for large language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6998–7018, Miami, Florida, USA. Association for Computational Linguistics.

Hao Zhao, Maksym Andriushchenko, Francesco Croce, and Nicolas Flammarion. 2024a. Long is more for alignment: a simple but tough-to-beat baseline for instruction fine-tuning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Minghao Li, Fei Huang, Nevin L. Zhang, and Yongbin Li. 2024b. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16776–16789, Torino, Italia. ELRA and ICCL.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. 2024. Llamafactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, Bangkok, Thailand. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini 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 *Thirty-seventh Conference on Neural Information Processing Systems*.

Daquan Zhou, Kai Wang, Jianyang Gu, Xiangyu Peng, Dongze Lian, Yifan Zhang, Yang You, and Jiashi Feng. 2023b. Dataset quantization. *Preprint*, arXiv:2308.10524.

Haotian Zhou, Tingkai Liu, Qianli Ma, Yufeng Zhang, Jianbo Yuan, Pengfei Liu, Yang You, and Hongxia Yang. 2024. Davir: Data selection via implicit reward for large language models. *Preprint*, arXiv:2310.13008.

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

A Experimental Details

A.1 Instruction Fine-tune Dataset

We evaluate Middo on three general instruction fine-tuning datasets.

- **Alpaca** (Taori et al., 2023): consists of 52,002 instruction-response pairs generated by Stanford University using the self-instruct (Wang et al., 2023) method based on OpenAI's text-davinci-003. This dataset is designed for fine-tuning dialogue models similar to ChatGPT to achieve efficient instruction-following capabilities.

- **Alpaca-4o-mini**: to evaluate performance on a higher-quality response dataset, we generated responses for all Alpaca instructions using GPT-4o mini, creating the Alpaca-4o-mini dataset.

- **WizardLM** (Xu et al., 2024): 70K data generated based on Evol-Instruct, which aims to generate more complex instruction data through a recursive evolutionary approach in order to improve the model's reasoning and instruction comprehension.

A.2 Models

We primarily conducted experiments on LLaMA 3.1-8B, and additionally performed extra experiments on Mistral 7B-v0.3.

- **LLaMA 3.1-8B** (Dubey et al., 2024): LLaMA 3.1-8B is a large language model released by Meta, featuring 8 billion (8B) parameters. It is part of the LLaMA (Large Language Model Meta AI) series, focusing on efficient reasoning and text generation capabilities. LLaMA 3.1-8B excels in code generation, language understanding, and conversational tasks, optimizing inference speed and training efficiency, making it suitable for research, commercial applications, and AI studies.

- **Mistral 7B-v0.3** (Jiang et al., 2023): Mistral 7B-v0.3 is an open-source language model developed by Mistral AI, featuring 7 billion parameters. It is optimized based on the Transformer architecture, emphasizing efficiency and multitasking capabilities. Compared to earlier versions, this model shows improvements in coding, mathematics, and reasoning tasks, making it suitable for chatbots, programming assistance, and natural language processing tasks. Mistral 7B-v0.3 incorporates feedback from the open-source community to enhance inference efficiency, delivering high performance with reduced computational resources.

A.3 Benchmarks

We assess model performance on general knowledge, mathematical problem-solving, code generation and commonsense reasoning benchmarks.

- **IFEval (Instruction Following Evaluation)** (Zhou et al., 2023c): a benchmark dataset designed to assess the instruction-following capabilities of large models. It encompasses various tasks, including general knowledge question answering, commonsense reasoning, and mathematical reasoning, aiming to measure the understanding and accuracy of language models when executing complex instructions.

- **MMLU (Massive Multitask Language Understanding)** (Hendrycks et al., 2021a): a large-scale, multi-task language understanding benchmark that covers 57 subjects, testing models on their knowledge and reasoning abilities across fields such as history, law, mathematics, and medicine. It serves as a significant indicator of general artificial intelligence knowledge levels.

- **GSM8K (Grade School Math 8K)** (Cobbe et al., 2021): a dataset specifically created for solving mathematical problems, containing approximately 8,500 elementary school math questions that primarily focus on basic arithmetic, logical reasoning, and text comprehension skills. This dataset is used to evaluate models' mathematical computation and reasoning abilities.

- **MATH** (Hendrycks et al., 2021b): consists of math competition problems from middle school and college levels, covering areas such as algebra, geometry, number theory, and calculus. This dataset is more challenging than GSM8K and is primarily used to assess models' performance on advanced mathematical reasoning tasks.

- **HumanEval** (Chen et al., 2021): a dataset for evaluating code generation capabilities, featuring a series of Python programming problems, each with a clear function signature and test cases. This dataset is commonly used to measure AI performance in automated code generation and programming tasks.

- **MBPP (Mostly Basic Programming Problems)** (Austin et al., 2021): a benchmark dataset for code generation, containing 1,000 basic programming questions that cover data structures, algorithms, and logical reasoning. It is suitable for assessing AI capabilities in fundamental programming tasks.

- **Hellaswag** (Zellers et al., 2019): a benchmark dataset for commonsense reasoning, consisting of a series of incomplete sentences that require models to select the most reasonable ending. This dataset tests models' contextual understanding and reasoning abilities by designing misleading options.

- **GPQA (Graduate-Level Google-Proof Q&A)** (Rein et al., 2024): a challenging dataset designed to evaluate the capabilities of LLMs and scalable oversight mechanisms. Let me provide more details about it.

A.4 Baselines

We compare Middo with both existing data selection and data augmentation methods on the Alpaca dataset.

Data Selection Methods.

- **Alpaca-clean** (Ruebsamen, 2023): a cleaned version of the Alpaca dataset that removes low-quality samples and duplicates, aiming to improve the overall quality of the dataset.
- **Superfiltering** (Li et al., 2024b): using smaller, weaker language models (such as GPT-2) as data filters to compute IFD allows for the selection of high-quality instruction tuning data.
- **Long** (Zhao et al., 2024a): directly select the 1,000 samples with the longest responses as training data.
- **AlpaGasus** (Chen et al., 2024): utilize powerful LLMs (such as ChatGPT) to automatically assess the sample quality in the Alpaca dataset and filter out high-quality data to enhance model training effectiveness.

Data Augmentation Methods.

- **Alpaca-GPT4** (Peng et al., 2023): a data augmentation method that uses GPT-4 to generate additional training data for the Alpaca dataset.
- **I-SHEEP** (Anonymous, 2025b): a data augmentation method that uses a self-supervised learning approach to generate additional training data for the Alpaca dataset.
- **WizardLM** (Xu et al., 2024): 70K data generated based on Evol-Instruct, which aims to generate more complex instruction data through a recursive evolutionary approach in order to improve the model's reasoning and instruction comprehension.

A.5 Hyperparameters

Fine-tune. For LLaMA-3.1-8B, we follow the Alpaca GitHub repository[1], setting the batch size to 32, the learning rate to 2×10^{-5}, and the warmup ratio to 0.03. For Mistral-7B-v0.3, we adjust the learning rate to 1×10^{-5}, as per official recommendations[2]. All the hyperparameters are detailed in Table 4.

[1] https://github.com/tatsu-lab/stanford_alpaca

[2] https://docs.mistral.ai/capabilities/finetuning

Hyperparameter	**Value**
LLaMA-3.1-8B	
Learning Rate	2×10^{-5}
Number of Epochs	1
Number of Devices	8
Per-device Batch Size	4
Gradient Accumulation Steps	8
Learning Rate Scheduler	cosine
Warmup Ratio	0.03
Max Sequence Length	4096
Mistral-7B-v0.3	
Learning Rate	1×10^{-5}
Number of Epochs	1
Number of Devices	8
Per-device Batch Size	4
Gradient Accumulation Steps	8
Learning Rate Scheduler	cosine
Warmup Ratio	0.03
Max Sequence Length	4096

Table 4: Hyperparameters used for fine-tuning.

Data Synthetic. We use the OpenAI API to generate data by GPT-4o-mini, setting both temperature and top_p to 1.0 to guarantee diversity.

Hyperparameter	**Value**
Pass@n	$n = 1$
Presence Penalty	0.0
Frequency Penalty	0.0
Repetition Penalty	1.0
Temperature	0.0
Top_p	1.0
Top_k	-1
Min_p	0.0
Max Tokens	4096
Min Tokens	0

Table 5: Hyperparameters used for evaluation.

Evaluation. All benchmarks are conducted in zero-shot and we conducted the tests using the default configuration of OpenCompass. All the hyperparameters are detailed in Table 5.

All experiments are conducted on 8 × NVIDIA Tesla A100 GPUs about 50 GPU hours.

B The Impact of Neighbor Number

We also explore how the number of neighbors k used in the *Embedding Cluster Dynamics* affects

k	IFEval	GSM8K	MATH	HumanEval	MBPP	Hellaswag	ARC-c	Average
1	43.59	38.74	9.20	35.98	39.8	48.59	17.17	33.3
2	51.56	43.21	10.72	40.85	41.00	57.47	12.12	**35.72**
3	40.82	40.49	9.50	32.32	39.20	59.72	8.59	32.95

Table 6: Impact of the number of neighbors (k) in the Embedding Cluster Dynamics on Middo performance. The table shows the performance across various benchmarks for different values of k, indicating that $k = 2$ yields the best overall average score.

the overall performance of Middo. By varying the number of neighbors, we analyze its impact on dataset diversity and model performance. Table 6presents the results of this analysis. We find that the optimal number of neighbors is $k = 2$, which achieves the best balance between diversity and performance. This setting ensures that the dataset is sufficiently expanded to enhance model generalization while avoiding excessive noise that may degrade performance.

C The Impact of Iterations

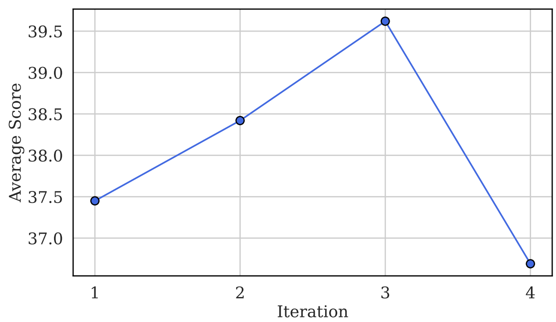


Figure 7: Performance trends on the Alpaca dataset across different optimization iterations. The model's performance peaks at three iterations and declines thereafter.

We tested the number of iterations on the Alpaca dataset as Figure 7 shows and found that the model's performance significantly declined after the third iteration. Therefore, we chose to optimize each dataset for three iterations. This optimal number is not necessarily fixed and may vary depending on the threshold of each iteration.

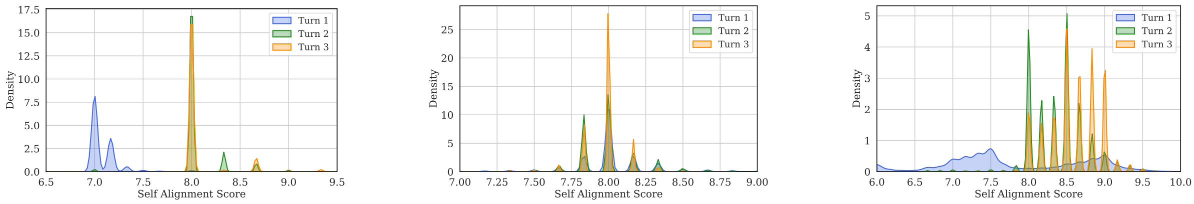
D Data Size Details

We selected approximately 5% of the data from each part in the first iteration and maintained the same threshold controlling hyperparameter m in subsequent rounds. We did not place excessive emphasis on the improvements brought about by differences in data volume, so our selection may not necessarily be optimal. The data size details are shown in Table 7.

Dataset	iteration	*loss*	*neighbor*	*self*	total
		LLaMA-3.1-8B			
Alpaca	*init*	$m = 1$	$m = -1$	$m = -1.5$	52,002
	iter1	1180	1924	1159	53,939
	iter2	299	1853	108	55,811
	iter3	242	1822	381	57,636
Alpaca 4o-mini	*init*	$m = 0$	$m = -1$	$m = -0.5$	52,002
	iter1	5684	8032	4145	60,865
	iter2	611	2291	876	63,184
	iter3	472	2127	661	65,324
Wizard	*init*	$m = 1$	$m = -1.5$	$m = -2$	70000
	iter1	3585	3585	2690	73,642
	iter2	959	3341	1016	76,993
	iter3	751	3414	420	80,419
		Mistral-7B-v0.3			
Alpaca	*init*	$m = 0.5$	$m = -2$	$m = -1$	52002
	iter1	2418	2111	2367	54,131
	iter2	1985	2091	932	56,268
	iter3	1788	1982	352	58,348
Alpaca 4o-mini	*init*	$m = 1$	$m = -2$	$m = -2.5$	52,002
	iter1	1407	7691	1499	59696
	iter2	1278	9116	1045	68874
	iter3	1346	2487	661	74036
Wizard	*init*	$m = 1$	$m = -1.5$	$m = -1.5$	70000
	iter1	5637	5709	5258	76429
	iter2	3558	5999	6310	82501
	iter3	3885	6229	3767	89178

Table 7: Data Size Details Across Iterative Refinement. For each dataset, the table lists the number of samples selected by the three components—*loss* (*Loss Patterns*), *neighbor* (*Embedding Cluster Dynamics*), and *self* (*Self-alignment Scores*). During each iteration, along with the total data size after refinement. The *init* row represents the original dataset size and the threshold controlling hyperparameter m corresponding to each component.

As the model's performance improves, the amount of data filtered out at the same threshold gradually decreases. We find that early training phases focus on complex, high-loss examples, whereas later stages prioritize diversity and data diversity is an ongoing necessity.



(a) Alpaca dataset. (b) Alpaca-4o-mini dataset. (c) WizardLM dataset.

Figure 8: Self-alignment score evolution across iterations. The x-axis represents the self-alignment scores, while the y-axis shows the density of data points.

E Self-alignment Scores

We provide detailed self-alignment score evolution across iterations on the Alpaca, Alpaca-4o-mini, and WizardLM datasets in Figure 8. These figures illustrate the dynamic evolution of self-alignment scores across iterations, highlighting the continuous improvement in dataset quality and alignment with model capabilities.

F Case Study

Complexity Optimization Prompt Template

You are a powerful LLM with the task to simplify the given #Prompt# for weaker open source LLMs (e.g. LLaMA, Qwen). Since the original #Prompt# is hard for them to handle, you need to rewrite it into a simpler version that these weaker LLMs can handle or learn from more easily. Follow the steps below carefully.
Step 1: Read the #Prompt# in detail. Suggest possible methods to make this prompt easier for weaker LLMs to handle or learn from, then list these methods as your #Methods List#.
Step 2: Create a comprehensive plan to simplify the #Prompt# using several methods from your #Methods List#. Present your plan in a clear, step-by-step format.
Step 3: Execute your plan to rewrite the #Prompt# into a simpler, more learnable version. You can change scenarios, contexts, or settings as needed. Your goal is to ensure that weaker LLMs learn from this prompt, rather than just memorizing an answer.
Step 4: Finally, review your rewritten version for any problems. Present only the #Final Rewritten Prompt#, without any additional explanation.
Please respond strictly in the following format:
Step 1 #Methods List#:
Step 2 #Plan#:
Step 3 #Rewritten Prompt#:
Step 4 #Final Rewritten Prompt#:

Overly Hard Data Example

Find the mode of the following set of numbers: 23, 16, 22, 19, 24, 21

Optimize Steps

Step 1 #Methods List#:
1. Simplify the language used in the prompt.
2. Break down the instructions into simpler steps.
3. Provide examples to clarify the concept of mode.
4. Use smaller numbers or more relatable contexts.
5. Ask the question in a more direct manner.

Step 2 #Plan#:
1. Change complex terminology like “mode” to a more descriptive explanation.
2. Use a familiar context, like favorite fruits or animals.
3. Clearly outline the steps needed to find the most common number, potentially using a smaller set.
4. Offer a simple example to illustrate the mode calculation.
5. Keep the structure straightforward and direct.

Step 3 #Rewritten Prompt#:
Find the most frequent number in this list: 3, 7, 2, 3, 5, 7. What number appears the most?’

Step 4 #Final Rewritten Prompt#:
Find the number that appears the most in this list: 3, 7, 2, 3, 5, 7. What is that number?

Optimized Easier Data

Find the number that appears the most in this list: 3, 7, 2, 3, 5, 7. What is that number?

Figure 9: Complexity optimization example.

Diversity Extension Prompt Template

You are a powerful LLM with the task to create brand new prompts for weaker open source LLMs (e.g. LLaMA, Qwen). You need to create a brand new complete prompt for them to learn in order to improve their knowledge and skills. Follow the steps below carefully.
Use #Hint Prompt 1# and #Hint Prompt 2# as guiding examples. Then read the #Core Prompt# in detail. Be inspired to suggest additional new prompts, and ultimately create only one completely original and diverse #New Prompt#.
Please respond strictly in the following format:
#New Prompt#:"

Sparse Data And Neighbors

#Hint Prompt 1#:
How long did Shakespeare live?

#Hint Prompt 2#:
How did the Industrial Revolution change society?

#Core Prompt#:
How did Julius Caesar die?

Extensioned Data

What were the key factors that led to the fall of the Roman Empire?

Figure 10: Diversity extension example.

Quality Optimization Prompt Template

You are a powerful LLM with the task to rewrite the given #Prompt# for weaker open source LLMs (e.g., LLaMA, Qwen). Since the original #Prompt# is of poor quality for them to learn, you need to rewrite it into a higher quality version that these weaker models can better learn from. Follow the steps below carefully.
Step 1: Read the #Prompt# in detail. Identify reasons for the poor quality of the #Prompt#. Suggest possible methods to improve the quality of the #Prompt#, then list these methods as your #Methods List#.
Step 2: Create a comprehensive plan to rewrite the #Prompt# using several methods from your #Methods List#. Present your plan in a clear, step-by-step format.
Step 3: Execute your plan to rewrite the #Prompt# into a higher quality version.
Step 4: Finally, review your rewritten version for any problems. Present only the #Final Rewritten Prompt#, without any additional explanation.
Please respond strictly in the following format:
Step 1 #Methods List#:
Step 2 #Plan#:
Step 3 #Rewritten Prompt#:
Step 4 #Final Rewritten Prompt#:

Low Quality Data Example

Given the following input, generate a job description for a product manager.
Noinput

Optimize Steps

Step 1 #Methods List#:
1. Clarify the input requirements to make them more explicit.
2. Use structured language with clear expectations for the output.
3. Specify the context in which the job description will be used.
4. Provide examples of desirable skills and experiences for the product manager role.
5. Ensure the language is professional and appropriate for a job description.

Step 2 #Plan#:
1. Add a clear request for a job description that outlines key responsibilities and qualifications for a product manager.
2. Specify the context, such as whether it's for a tech company, a startup, or a different industry.
3. Suggest including attributes and skills relevant to the product manager role, such as project management, communication, and analytics.
4. Format the request to clearly indicate the sections that should be included in the job description (e.g., summary, responsibilities, qualifications).

Step 3 #Rewritten Prompt#:
<note:same as next box>

Step 4 #Final Rewritten Prompt#:
<note:same as next box>

Optimized High Quality Data

Please create a detailed job description for a Product Manager position in a tech company. The description should include the following sections:
- Job Summary: A brief overview of the role and its importance within the company.
- Key Responsibilities: A list of primary duties that the Product Manager will be responsible for, such as managing product development, collaborating with cross-functional teams, and conducting market research.
- Required Qualifications: Skills and experiences necessary for the role, including knowledge of product management methodologies, strong analytical abilities, and effective communication skills.
- Preferred Attributes: Any additional qualities that would make a candidate stand out, such as a passion for technology or experience in agile environments.

Figure 11: Quality optimization example.

Self-Alignment Instruction Score Prompt Template

We would like to request your feedback on the {dimention} of the prompt displayed below.

Prompt:
Give three tips for staying healthy.

Please rate according to the {dimention} of the prompt to evaluate {explanations}. Each prompt is scored on a scale of 0 to 10, with higher scores indicating higher {dimention}. Try to avoid scoring a full 10. Give your rating number first, then give a explanation of your rating.

Self-Alignment Instruction Score Dimentions and Explanations

"factuality": "whether the information provided in the prompt is accurate and based on reliable facts and data",
"clarity": "whether the prompt is clear and understandable, and whether it uses concise language and structure",
"completeness": "whether the prompt provides sufficient information and details".

Instruction Clarity Score Example

7. The prompt is clear and understandable, but it could be more concise.

Figure 12: Self-Alignment instruction score example.

Self-Alignment Response Score Prompt Template

We would like to request your feedback on the {dimention} of the prompt displayed below.

Prompt:
What are the three primary colors?

Response:
The three primary colors are red, blue, and yellow.

Please rate according to the {dimention} of the response to evaluate {explain}. Each response is scored on a scale of 0 to 10, with higher scores indicating higher {dimention}. Try to avoid scoring a full 10. Give your rating number first, then give a explanation of your rating.

Self-Alignment Response Score Dimentions and Explanations

"factuality": "whether the information provided in the response is accurate and based on reliable facts and data",
"clarity": "whether the response is clear and understandable, and whether it uses concise language and structure",
"completeness": "whether the response provides sufficient information and details".

Response Clarity Score Example

8.5. The response is clear and understandable, but it could be more concise.

Figure 13: Self-Alignment response score example.