ULTRAIF: Advancing Instruction Following from the Wild

Anonymous ACL submission

Abstract

Instruction-following made modern large language models (LLMs) helpful assistants. However, the key to taming LLMs on complex instructions remains mysterious, for that there are huge gaps between models trained by open-source community and those trained by leading companies. To bridge the gap, we propose a simple and scalable approach ULTRAIF for building LLMs that can follow complex instructions with open-source data. ULTRAIF first decomposes real-world user prompts into simpler queries, constraints, and corresponding evaluation questions for the constraints. Then, we train an *UltraComposer* to compose constraint-associated prompts with evaluation questions. This prompt composer allows us to synthesize complicated instructions as well as filter responses with evaluation questions. In our experiment, for the first time, we successfully align LLaMA-3.1-8B-Base to catch up with its instruct version on 5 instruction-following benchmarks without any benchmark information, using only 8B model as response generator and evaluator. The aligned model also achieved competitive scores on other benchmarks. Moreover, we also show that ULTRAIF could further improve LLaMA-3.1-8B-Instruct through self-alignment, motivating broader use cases for the method.

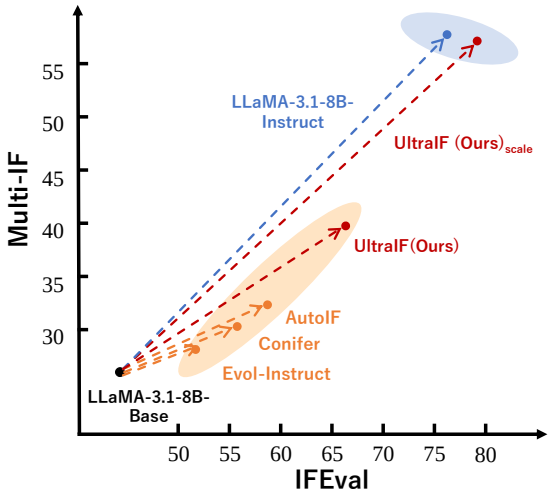

Figure 1: Instruction-following performance comparison of ULTRAIF against baselines.

1 Introduction

Large language models (Meta, 2024; OpenAI, 2024) have demonstrated remarkable capabilities, especially in following complex instructions. While modeling such ability is crucial, the technical details and the instruction datasets used in state-of-the-art LLMs remain mysterious. For example, LLaMA3 (Meta, 2024) reportedly leverages instruction-following data at the tens of millions scale but has not been open-sourced. This lack of transparency has resulted in a significant gap between research community and leading companies.

Recent efforts in aligning LLMs to follow instructions have focused on creating high-quality instruction-following data. On the one hand, Wei et al. (2021); Rajani et al. (2023); Jiang et al. (2023) involve human annotators in developing instructions and manually crafting corresponding responses. While effective, these methods are label-intensive, heavily reliant on human expertise, and face challenges in scalability and cost efficiency. On the other hand, Xu et al. (2023); Wang et al. (2023); Sun et al. (2024a); Dong et al. (2024) attempt to leverage LLMs to automatically construct high-quality instruction data. Specifically, Xu et al. (2023); Sun et al. (2024a) guide LLMs to generate constraints and evolve initial instructions into more complex forms. However, these LLMs-driven methods heavily rely on models' instruction-evolving capability and overemphasize instruction complexity, ultimately hindering the diversity of evolved instructions and the correctness of generated responses. To improve this, Wang et al. (2023); Dong et al. (2024) introduce handcrafted constraints inspired by human priors

to guide LLMs. For instance, Dong et al. (2024) introduces constraints that can be verified by code execution to ensure response correctness. However, these handcrafted constraints introduce rigidity, leading to homogeneous instructions and making it narrow in encompassing more complex or diverse instructions (e.g., *write in Shakespeare's tone*). As a result, scaling such instruction with correct responses remains a significant challenge, limiting the applicability of modeling the distribution of instructions from real-world users.

In this paper, we propose ULTRAIF, a simple and scalable method which synthesizes high-quality instruction-following data. The core idea of ULTRAIF is to decompose real-world user instructions for both constraints and evaluation questions, then train a composer model to synthesize diverse and complex instructions with verification questions. To achieve this, we first utilize LLM to decompose human instructions into simplified instructions and their associated constraints. For each constraint, the LLM further generates the corresponding evaluation question to verify whether the upcoming response meets the requirement. With these components, we train ***UltraComposer***, which takes the simplified instruction as input and outputs the original instruction along with its evaluation question. In this way, the composer learns to evolve instructions with verifiable constraints, and benefits from the generalization ability of LLMs rather than handcrafted rules. With the composer, ULTRAIF could make any instruction more complicated to synthesize a large-scale and diverse dataset. The evaluation questions further help with quality control in rejection sampling and preference learning (Rafailov et al., 2024; Chen et al., 2024).

Through comprehensive experiments, we demonstrate that ULTRAIF significantly enhances the instruction-following capabilities of LLMs with high scalability and cost efficiency. Our evaluation, conducted on the LLaMA-3.1-8B model across five instruction-following datasets, confirms ULTRAIF's strong alignment with general instructions. Notably, as shown in Figure 1, by scaling up the training data, we achieve a milestone, optimizing the LLaMA-3.1-8B-Base model to match the instruction-following ability of its instruct version. Additionally, we assess the generability of ULTRAIF by evaluating it on mathematical, reasoning, coding, and general conversation domains. Furthermore, we explore the potential of self-alignment in ULTRAIF by further optimizing the LLaMA-3.1-8B-Instruct model, and achieved sustaintial improvement.

The main contributions of our paper include:

- We introduce ULTRAIF, a simple and scalable approach that leverages real-world user instructions to train a composer model, *UltraComposer*, enabling the synthesis of complex and diverse instructions with correct responses.
- Our experiments demonstrate the strong performance of ULTRAIF in handling complex instructions, surpassing all baselines under the same data budget while retaining general capabilities in domains such as mathematics, coding, and conversational tasks.
- We reach a new milestone by optimizing the LLaMA-3.1-8B-Base model to match the instruction-following abilities of its Instruct counterpart with only 200k data, and showcase the self-alignment potential by further optimizing the LLaMA-3.1-8B-Instruct model on it own.

2 ULTRAIF

2.1 Overview

ULTRAIF synthesizes high-quality instruction-following datasets in two stages. As shown in Figure 2, ULTRAIF first constructs the ***UltraComposer*** by decomposing user instructions into simplified ones and constraints, along with corresponding evaluation questions (§2.2). This specialized composer facilitates the synthesis of instructions with more complex and diverse constraints, while the evaluation questions ensure the correctness and reliability of the generated responses. Then, the ***Generate-then-Evaluate*** process (§2.3) uses *UltraComposer* to incorporate constraints into instructions and assesses the generated responses using corresponding evaluation questions covering various quality levels.

2.2 UltraComposer

Previous studies (Xu et al., 2023; Sun et al., 2024a) that rely solely on LLMs are limited by the models' instruction-evolving ability, which restricts the diversity of synthetic instructions and compromises response accuracy. While Wang et al. (2023); Dong et al. (2024) address response correctness through handcrafted constraints, this approach further limits instruction diversity. In contrast, ULTRAIF focuses on generating diverse, complex instructions

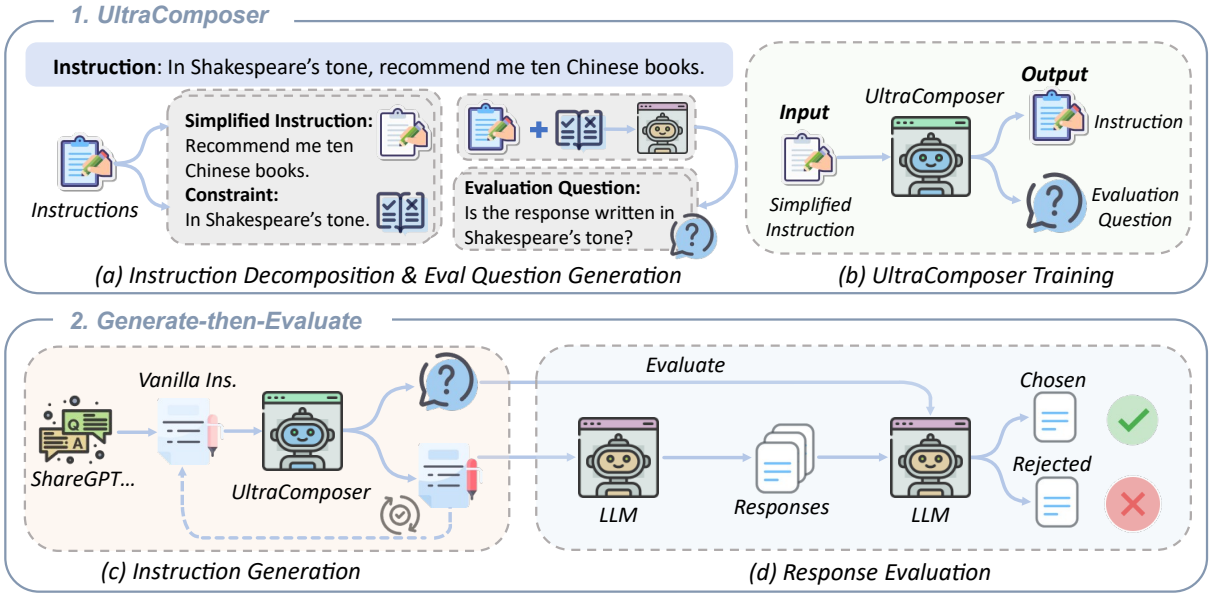

Figure 2: The framework of ULTRAIF. Specifically, ULTRAIF begins by training the ***UltraComposer***, which decomposes real-world user instructions and evaluation questions. For (a), the given instruction can be decomposed into several pairs, such as the numeric constraint 'ten books' and content constraint 'Chinese books'. Next, ULTRAIF adopts a ***Generate-then-Evaluate*** process, where the composer iteratively adds multiple constraints to each collected instruction and then applies the evaluation questions for rejection sampling.

with correct responses. To achieve this, we propose the *UltraComposer*, a specialized model to synthesize diverse instructions and generate corresponding evaluation questions. Building this composer model involves three key steps: instruction decomposition, evaluation question generation, and *UltraComposer* training.

Instruction Decomposition The decomposition process leverages LLMs to decompose complex instructions into different components. These components consist of a set of simplified instructions paired with constraints that represent the underlying requirements of the original instruction. For example, as shown in Figure 2 (a), the instruction (X) *"In Shakespeare's tone, recommend me ten Chinese books."* can be decomposed into the simplified instruction (x_1) *"Recommend me ten Chinese books."* and the paired constraint (c_1) *"In Shakespeare's tone."*, etc. This step is essential for disentangling intricate objectives into more structured elements, extending beyond basic format or content constraints (Dong et al., 2024; Wang et al., 2023), and forming a foundation to model the distribution of real-world user instructions effectively.

Evaluation Question Generation While Xu et al. (2023); Wang et al. (2023) focus on improving the complexity of instructions, omitting the quality of generated responses often leads to low-quality samples. Inspired by Qin et al. (2024), we utilize LLM to generate evaluation questions for each constraint. Given the example, the evaluation question (q_1) would be *"Is the response written in Shakespeare's tone?"*. These questions are designed to assess the generated responses for adherence to the constraints. This mechanism not only addresses the limitation of checking only programmatically verifiable constraints (Dong et al., 2024), but also improves the reliability of the response and its alignment with the original instruction.

$$X \rightarrow \{(x_1, c_1, q_1), \ \dots, (x_n, c_i, q_i)\}, \quad i \in \mathbb{N} \quad (1)$$

UltraComposer Training With decomposed instructions and evaluation questions, we train *UltraComposer* to take a simplified query (x_i) as input and generate the original instruction (X) with its evaluation question (q_i), denoted as Eq.2, as shown in Figure 2 (b). This enables *UltraComposer* to complicate instructions in a single step. Additionally, it enhances constraint diversity by incorporating not only the LLM's inherent knowledge but also distributions observed in real-world scenarios.

$$UltraComposer(x_i) \rightarrow (X, \ q_i), \quad i \in \mathbb{N} \quad (2)$$

2.3 Generate-then-Evaluate

With *UltraComposer*, ULTRAIF efficiently produces high-quality instruction-following data

through a Generate-then-Evaluate process, encompassing both instruction generation and response evaluation to support both Supervised Fine-tuning and Preference Learning strategies.

Instrucion Generation *UltraComposer* adapts the augmentation process fully automated and aligns with human preferences. This step starts by collecting user instructions from existing datasets (Chiang et al., 2023; Teknium, 2023; Rajani et al., 2023), and then use the Composer to augment these instructions. As shown in Eq.3, this process can be conducted iteratively, enabling the generation of more complex and realistic instructions ($\bar{x}$) with multiple constraints, paired with corresponding evaluation questions ($\bar{q}$).

$$UltraComposer(x^{(n)}) \rightarrow (\bar{x}^{(n)},\ \bar{q}^{(n)}),\ \ n \in \mathbb{N}$$
$$x^{(n+1)} = \bar{x}^{(n)}, \quad \bar{q}^{(n+1)} = \bar{q}^{(n+1)} \cup \bar{q}^{(n)} \tag{3}$$

Response Evaluation Next, we prompt LLMs to generate K responses for each augmented instruction. As 'LLM-as-judge' paradigm is prevalent (Zheng et al., 2023), human can be replaced by LLMs to assess the quality of response, so the quality of generated responses is assessed by evaluation questions. This results in a dataset $\mathcal{D}_{data}$ comprising $(\bar{x},\ \bar{q}, y_{chosen}, y_{rejected})$. Ideally, this process requires only three to four calls to LLMs, significantly reducing the computational cost, achieves greater efficiency and incurs minimal costs when constraining large-scale datasets compared to previous research (Xu et al., 2023; Dong et al., 2024).

3 Experiments

3.1 Experimental Setup

Datasets and Baselines To train *UltraComposer*, we decompose instructions from ShareGPT (Chiang et al., 2023) and generate corresponding evaluation questions by LLaMA-3.1-70B-Instruct. In our experiments, we collect human instructions from existing open-source datasets, including ShareGPT, OpenHermes2.5, and No Robots (Teknium, 2023; Rajani et al., 2023; Chiang et al., 2023), and employ *UltraComposer* to complicate instructions and then generate responses. For baselines, we reimplement existing methods using either public datasets (Sun et al., 2024a; Xu et al., 2023) or available implementations (Dong et al., 2024), and include a series of currently open and closed-source LLMs. More details are in Appendix A.1.

Evaluation We evaluate ULTRAIF on five instruction-following benchmarks, including IFEval (Zhou et al., 2023), Multi-IF (He et al., 2024), InfoBench (Qin et al., 2024), FollowBench (Jiang et al., 2023), and LiveBench (White et al., 2024). While IFEval and Multi-IF focus on testing verifiable instructions using functions, the others extend to more general instructions that need to be evaluated by LLMs. Additionally, we further test the general ability of ULTRAIF such as mathematical (Chen et al., 2021), reasoning (Suzgun et al., 2022), coding (Cobbe et al., 2021), and general interaction capabilities (Li et al., 2024). The details about benchmarks are provided in Appendix A.2.

Experimental Settings We fine-tune LLaMA-3.1-8B-Instruct to build our *UltraComposer*. Subsequently, we explore two settings to implement our training strategies listed in Appendix A.3, including Supervised Fine-tuning and Preference Learning. The implementation details are listed in Appendix A.4. And the prompts about instruction decomposition and evaluation question generation are provided in Appendix B.

- **Strong-to-Weak.** In this setting, knowledge is distilled from a larger model to a smaller one. For ULTRAIF, we leverage LLaMA-3.1-70B-Instruct for response generation and evaluation and then train LLaMA-3.1-8B-Base.
- **Self-Alignment.** We replace the supervision model with Llama-3.1-8B-Instruct.

3.2 Main Results

Table 1 shows the performance of ULTRAIF on five instruction-following benchmarks.

ULTRAIF Outperforms All Previous Methods In the strong-to-weak setting, ULTRAIF demonstrates performance that is comparable to or exceeds previous methods across all datasets. By fine-tuning on our generated data, ULTRAIF achieves substantial improvements, particularly on IFEval and Multi-IF. When compared to strong baselines like AutoIF (Dong et al., 2024), ULTRAIF achieves scores of 53.97 (Pr(S)) and 64.15 (Ins(S)) on IFEval and 81.91 (DRFR) on InfoBench, surpassing AutoIF by margins ranging from 1.29% to 6.84%. These results underscore ULTRAIF's capability to effectively follow instructions, even with lower training data, representing a significant advancement over state-of-the-art approaches.

Method	#Data	IFEval				Multi-IF			InfoBench	LiveBench	FollowBench
		Pr(S)	Pr(L)	Ins(S)	Ins(L)	Turn1	Turn2	Turn3	DRFR	Score	SSR
GPT-4†	-	76.90	79.30	83.60	85.40	81.50	70.50	60.90	89.40	69.40	78.60
LLaMA-3.1-8B-Instruct†	-	69.13	74.86	77.46	81.65	68.54	59.63	51.26	81.33	57.10	63.41
***Strong-to-Weak* (Supervisor: GPT-4o-mini)**											
SPaR (2024)‡	8k	54.71	58.59	64.86	68.70	55.37	36.22	27.23	78.61	50.80	59.37
***Strong-to-Weak* (Supervisor: LLaMA-3.1-70B-Instruct)**											
LLaMA-3.1-8B (ShareGPT)	10k	43.99	54.34	54.32	64.39	44.69	25.11	18.50	81.56	33.20	59.59
Evol-Instruct (2023)‡	10k	41.96	45.66	54.44	58.03	39.03	24.34	19.14	75.74	44.90	43.87
Conifer (2024a)‡	13k	46.40	51.02	58.51	62.59	44.91	25.83	17.95	75.73	45.60	52.42
AUTOIF (2024)‡	10k	47.13	56.93	57.55	67.02	47.63	27.53	20.53	80.62	40.50	**60.41**
ULTRAIF											
+ SFT	10k	53.97	58.59	64.15	68.82	52.55	29.34	22.29	81.91	42.20	59.50
+ Iterative DPO	8k	**58.22**	**65.25**	**68.11**	**74.22**	**58.14**	**35.65**	**26.55**	**83.56**	**49.50**	59.99
***Self-Alignment* (Supervisor: LLaMA-3.1-8B-Instruct)**											
ULTRAIF											
+ SFT	10k	55.82	58.78	66.18	69.54	55.59	36.72	28.07	77.78	46.60	55.88
+ Iterative DPO	8k	56.93	64.14	66.66	73.02	58.63	42.04	31.20	79.86	54.20	58.56
+ SFT *scale up*	175k	69.87	72.46	77.46	80.22	66.24	53.66	42.19	79.20	51.40	59.93
+ Iterative DPO	20k	**71.35**	**75.42**	**79.38**	**83.09**	**69.63**	**58.28**	**46.86**	**80.70**	**56.00**	**62.55**

Table 1: The main results on five instruction-following benchmarks. Results marked with † are sourced from the original benchmarks, and ‡ represents we reimplement the methods.

Iterative DPO Boosts Performance Effectively As shown in Table 1, the iterative DPO process substantially enhances alignment with complex instructions. Specifically, in comparison to SFT, iterative DPO achieves an average improvement of 5% in the strong-to-weak setting and 3.8% in the self-alignment setting on MultiIF. Furthermore, this process enables ULTRAIF to surpass state-of-the-art methods in three benchmarks that require LLM-based evaluation, with an improvement of 1.5% on InfoBench, 4.6% on LiveBench, and 2.62% on FollowBench, demonstrating the importance of ULTRAIF in handling diverse instructions.

Smaller Supervisor Yields Better Performance The self-alignment setting, which employs smaller model as supervisor, achieves superior performance relative to the strong-to-weak setting. This divergence is particularly evident during the SFT stage, wherein self-alignment outperforms strong-to-weak on IFEval, Multi-IF and LiveBench. While improvements introduced by DPO remain relatively incremetal, self-alignment still exhibits superior performance on two benchmarks. These results align with prior research by Hui et al. (2024); Zhang et al. (2025); Li et al. (2025), which demonstrates that self-generated responses more closely to the distribution of the base model.

ULTRAIF Achieves A New Milestone By scaling up the training data, ULTRAIF achieves a new milestone in instruction-following alignment. With 175k data in the SFT stage and 20k data in the DPO stage, ULTRAIF reaches impressive performance, with 71.35 (Pr(S)) and 79.38 (Ins(S)), while the LLaMA-3.1-8B-Instruct model only achieves 69.13 (Pr(S)) and 77.46 (Ins(S)), and comparable across the left benchmarks. This demonstrates that ULTRAIF, when optimized and trained on larger datasets, not only improves instruction-following capabilities but also comes closest to matching the performance of LLaMA-3.1-8B-Instruct, marking a significant leap forward in model performance.

3.3 Cross-Domain Validation

Table 2 presents a comparative evaluation of ULTRAIF across four general domains against AutoIF and LLaMA-3.1-8B-Instruct. Although ULTRAIF exhibits slightly lower performance than AutoIF on math, it achieves substantial improvements on code and conversation. These gains are further amplified by scaling up the training data, and the application of the DPO stage consistently enhances performance across all evaluated domains. In particular, ULTRAIF contributes significantly to improving general capabilities, as evidenced by its performance on the comprehensive LiveBench benchmark (White et al., 2024) and the ArenaHard conversational benchmark (Li et al., 2024). ULTRAIF surpasses AutoIF by a statistically significant margin of 4.2% on LiveBench and achieves a substantial 15.4% improvement in conversational performance on ArenaHard. These results highlight the effectiveness of ULTRAIF in advancing the development of more versatile and general models.

Method	Code	Reasoning	Math	Conversation	General
	HumanEval	BBH	GSM8k	Arena Hard	LiveBench [All]
LLaMA-3.1-8B-Instruct	65.24	68.54	80.80	18.30	25.90
AutoIF (2024)	46.34	67.18	**51.50**	9.20	17.50
ULTRAIF + SFT	43.90	67.33	48.60	12.20	21.30
+ Iterative DPO	**47.56**	**68.03**	48.10	**16.00**	**21.70**
+ SFT *scale up*	52.44	67.26	66.70	16.00	22.80
+ Iterative DPO	**55.49**	**68.44**	**68.00**	**31.40**	**23.10**

Table 2: The general performance on mathematical, reasoning, coding, and conversational domains. We report Pass@1 on HumanEval, Acc on BBH and GSM8k, and Win Rate on Arena Hard.

Iteration	IFEval				Multi-IF			LiveBench
	Pr(S)	Pr(L)	Ins(S)	Ins(L)	Turn1	Turn2	Turn3	Score
Iter 1	$55.45_{+1.48}$	$61.55_{+2.96}$	$65.10_{+0.95}$	$70.74_{+1.92}$	$56.13_{+3.58}$	$32.11_{+2.77}$	$24.38_{+2.09}$	$42.20_{+0.00}$
Iter 2	$55.08_{+1.11}$	$62.66_{+4.07}$	$65.47_{+1.32}$	$71.82_{+3.00}$	$57.26_{+4.71}$	$34.92_{+5.58}$	$26.28_{+4.00}$	$47.20_{+5.00}$
Iter 3	$56.75_{+2.78}$	$63.03_{+4.44}$	$66.79_{+2.64}$	$72.42_{+3.60}$	$57.10_{+4.55}$	$34.87_{+5.53}$	$26.11_{+3.82}$	$45.70_{+3.50}$
Iter 3$_{w.NCA}$	$58.22_{+4.25}$	$65.25_{+6.66}$	$68.11_{+3.96}$	$74.22_{+5.40}$	$58.14_{+5.59}$	$35.65_{+6.31}$	$26.55_{+4.26}$	$49.50_{+7.30}$

Table 3: The performance compared to the SFT model across each iteration during the Iterative DPO process.

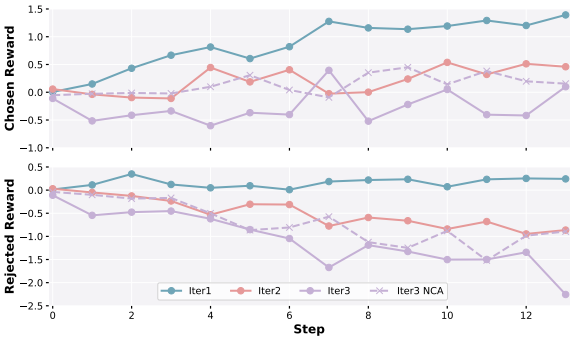

Figure 3: Reward trajectories for the chosen and rejected samples across the steps of each iteration during the Iterative DPO process.

Setting	c=1	c=2	c=3
Strong-to-Weak	91.76	86.41	79.44
Self-Alignment	92.46	89.57	85.79

Table 4: The pass rate of of ULTRAIF across increasing instruction complexity levels during SFT stage.

4 Analysis

4.1 Impact of the Iterative DPO Process

Given that *UltraComposer* enables the iterative incorporation of constraints into instructions, we adopt an Iterative DPO training strategy in which instruction complexity is gradually increased over successive training stages. Figure 3 illustrates the reward trajectories for the chosen and rejected samples. As training progresses, the reward margin between chosen and rejected samples widens. However, by the third iteration, the optimization objective begins to diverge, prioritizing the maximization of the reward margin over improving the absolute reward values. This shift leads to both chosen and rejected sample rewards falling below zero, ultimately degrading model performance on three benchmarks, as shown in Table 3. To address this, we replace the DPO objective with NCA (Chen et al., 2024), which stabilizes the training dynamics and results in more consistent and robust performance across benchmarks.

4.2 Analysis of Sampling Efficiency

We also evaluate the sampling efficiency of ULTRAIF by measuring the proportion of generated responses that satisfy the filter question criteria, as instruction complexity increases through iterative applications of *UltraComposer*. Table 4 presents the overall pass rates of ULTRAIF across varying constraint levels ($c = 1, 2, 3$). The pass rate decreases with increasing instruction complexity. Notably, the self-alignment setting consistently outperforms the strong-to-weak setting. These trends are consistent with findings from prior studies (Hui et al., 2024; Zhang et al., 2025; Li et al., 2025).

Furthermore, Appendix C.1 compares the data synthesis pass rates of ULTRAIF and AutoIF, demonstrating the superior scalability and cost-efficiency of ULTRAIF in dataset construction. To assess the reliability of LLM-based evaluation during filtering, we conduct a human evaluation study, where we randomly sample 75 examples from the SFT dataset and assign each to five human annotators. The resulting agreement shows that LLM evaluations achieve an accuracy of approximately 80%, which we consider sufficient for practical use.

Method	#Data	IFEval	Followbench	Multi-IF
		541	944	4501
		N-gram (N=13) Overlap Ratio		
AutoIF	10k	0.0000	0.0000	0.0000
UltraIF	10k	0.0000	0.0026	0.0000
UltraIF	175k	0.0000	0.0033	0.0000
		LLM Decontaminator (Rephr.)		
AutoIF	10k	0.0000	0.0032	0.0000
UltraIF	10k	0.0018	0.0117	0.0000
UltraIF	175k	0.0166	0.0360	0.0082

Table 5: Contamination analysis on SFT data generated by AutoIF and UltraIF on three benchmarks.

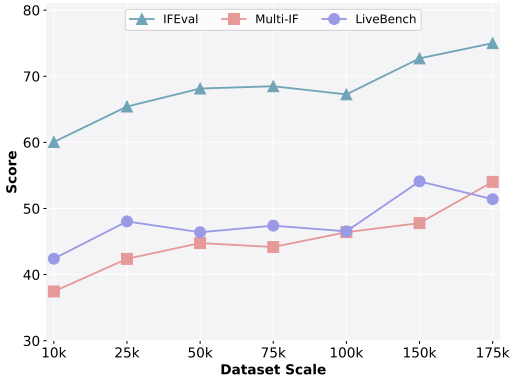

Figure 4: Scaling the training set on SFT stage.

4.3 Contamination Analysis

To ensure the integrity of our evaluation, we conduct a comprehensive contamination analysis of the training data generated by both AutoIF and UltraIF across three benchmarks. We use a traditional n-gram overlap method ($overlap\ ratio = \frac{\#matched_ngrams}{\#total_ngrams}$) to identify any exact or near-duplicate sequences between training and test sets and utilize the LLM-based contamination detection framework from LM-Sys (Yang et al., 2023), which leverages advanced LLMs to identify semantically rephrased versions of test instances in the training data. Table 5 reveals extremely low contamination rates across all configurations. Notably, both detection methods show that UltraIF exhibits contamination levels comparable to those of AutoIF. These findings confirm that the self-generated training data is clean and does not compromise the validity of our evaluation results.

4.4 Scalability of UltraIF

To validate the effieiency and effectiveness of UltraIF, we conduct scaling up experiments under the self-alignment setting. Figure 4 shows the impact of varying training data sizes during the SFT stage. With about 175k training samples, UltraIF demonstrates powerful performance compared to

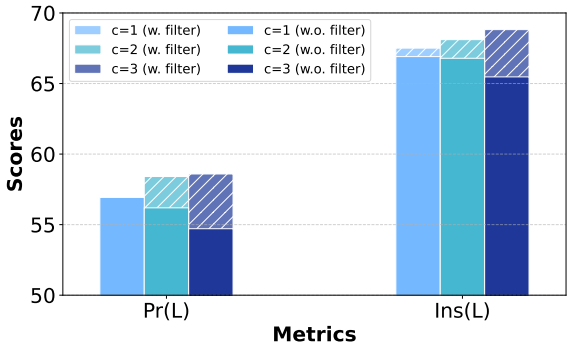

Figure 5: Ablations on the number of added constraints and the evaluation question filter.

baselines, highlighting its strong capacity to scale with increasing data volume. Moreover, we analyze the impact of multi-turn data in Appendix C.2, and incoporate such data in our scaling experiments.

4.5 Ablation Studies on UltraIF

The iterative augmentation capability of *UltraComposer* raises a critical question for the SFT stage: should simple or complex instructions be prioritized for training? Figure 5 presents the results of using varying levels of instruction complexity during the SFT stage. The results demonstrate that as instruction complexity increases, performance correspondingly improves, reaching the peak after three iterations with. Furthermore, we evaluate the effectiveness of our evaluation questions. Without filtering out low-quality responses, performance deteriorates significantly over 3.35%-5.36%. This mechanism becomes increasingly critical as instruction complexity grows, with the performance gap widening alongside the increasing complexity, underscoring the importance of this module in maintaining high-quality training data. We provide cases to illustrate the augmented instructions and evaluation questions in Appendix C.3.

4.6 Extension of Self Alignment

In our main experiments, we distill knowledge from the Instruct version model to enhance the vanilla model, demonstrating the effectiveness of UltraIF. However, the potential for UltraIF to independently enhance a strong model like Cheng et al. (2024) has not yet been explored. In this section, we conduct experiments to investigate the self-improvement capabilities of UltraIF. Under the self-alignment setting, we use data generated by LLaMA-3.1-8B-Instruct to enable the model to train itself. As shown in Figure 6, UltraIF significantly boosts the performance of the strong model

Method	#Data	IFEval				Multi-IF			LiveBench	FollowBench
		Pr(S)	Pr(L)	Ins(S)	Ins(L)	Turn1	Turn2	Turn3	Score	SSR
Qwen2-7B-Instruct	-	52.68	55.63	62.82	65.34	54.44	39.41	29.95	46.30	63.36
AutoIF(2024)†	10k	40.70	44.50	51.30	55.40	-	-	-	-	53.30
ULTRAIF + SFT	10k	44.17	47.31	54.19	57.55	47.56	25.38	18.13	35.20	54.16
+ Iterative DPO	8k	45.28	48.61	56.59	59.23	48.57	28.45	19.60	39.80	55.44

Table 6: The results on four instruction-following benchmarks with Qwen2-7B as the backbone model.

Figure 6: The performance of exploring the potentiality of ULTRAIF on self-alignment.

across different size of training data, even without a more powerful supervisor. Specifically, ULTRAIF improves performance on IFEval by 2.4%-5.9%, on Multi-IF by 3.74%-5.38%, further validating the effectiveness of our approach.

4.7 Generalizability of ULTRAIF

To assess the generalizability of ULTRAIF across different foundation models, we apply it to the Qwen2-7B base model (Yang et al., 2024). As shown in Table 6, ULTRAIF maintains strong performance when built on the Qwen2 architecture, demonstrating its adaptability to different model backbones. Notably, compared to AutoIF, ULTRAIF exhibits greater potential in aligning LLMs with instruction-following capabilites.

5 Related Work

5.1 Instruction Following

Instruction following is a core area for LLMs, aiming to improve understanding and execution of complex human instructions. Early work (Wei et al., 2021; Rajani et al., 2023; Jiang et al., 2023) use curated datasets of human-written instructions and responses. Recent methods automate this using LLMs, Xu et al. (2023) and Sun et al. (2024a) prompt LLMs to evolve or complicate instructions. However, this can yield low-quality data due to LLMs' limitations. To improve quality, Wang et al. (2023); Dong et al. (2024) add human priors like verifiable constraints, but this reduces instruction diversity. In contrast, ULTRAIF decomposes user instructions into constraints and evaluation questions, then trains *UltraComposer* to generate diverse, complex instructions with accurate responses, offering a robust approach to instruction data generation.

5.2 Perference Learning

Preference learning has emerged as a key method to improve instruction-following by refining models through feedback (Ouyang et al., 2022; Dong et al., 2024; Sun et al., 2024a; Gao et al., 2024). It typically enhances models finetuned on instruction data using reward signals from human or automated to guide learning. While RLHF with PPO is common, it depends on ranked responses, which are costly and labor-intensive. Recent work (Rafailov et al., 2024; Chen et al., 2024) addresses this via direct preference optimization, reducing reliance on human input. ULTRAIF supports this by generating evaluation questions that guide preference learning more efficiently. It complements direct optimization with a scalable, cost-effective approach to producing instruction-following data.

6 Conclusion

In this paper, we propose ULTRAIF, a scalable and effective approach for synthesizing high-quality instruction-following data. By decomposing human instructions into simplified queries, constraints, and corresponding evaluation questions, we train *UltraComposer* that enables the efficient generation of constraint-aware instructions. Across two different settings, ULTRAIF demonstrates strong performance across five instruction-following benchmarks and four general benchmarks. Extensive experiments conducted on LLaMA-3.1-8B-Instruct further highlight ULTRAIF's potential for self-alignment. Most importantly, we are the first to optimize the LLaMA-3.1-8B-Base model to match the instruction-following capabilities of its Instruct counterpart, underscoing the effectiveness and potential of our approach.

Limitations

Due to limitations in time and computational resources, ULTRAIF has not yet been evaluated on a wider range of backbone models or on models with larger parameter scales. Nevertheless, the current experimental results provide sufficient evidence of its generalizability across different foundational architectures. Additionally, since the full pipeline depends on LLMs for supervision, the data generation process may involve limited controllability and introduces potential risks related to consistency and reliability. To minimize these risks, we apply responses filtering to ensure the quality and stability of the generated data.

References

Huayu Chen, Guande He, Lifan Yuan, Ganqu Cui, Hang Su, and Jun Zhu. 2024. Noise contrastive alignment of language models with explicit rewards. *arXiv preprint arXiv:2402.05369*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Spar: Self-play with tree-search refinement to improve instruction-following in large language models. *arXiv preprint arXiv:2412.11605*.

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

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*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*.

Bofei Gao, Feifan Song, Yibo Miao, Zefan Cai, Zhe Yang, Liang Chen, Helan Hu, Runxin Xu, Qingxiu Dong, Ce Zheng, Shanghaoran Quan, Wen Xiao, Ge Zhang, Daoguang Zan, Keming Lu, Bowen Yu, Dayiheng Liu, Zeyu Cui, Jian Yang, and 6 others. 2024. Towards a unified view of preference learning for large language models: A survey. *arXiv preprint arXiv:2409.02795*.

Yun He, Di Jin, Chaoqi Wang, Chloe Bi, Karishma Mandyam, Hejia Zhang, Chen Zhu, Ning Li, Tengyu Xu, Hongjiang Lv, and 1 others. 2024. Multi-if: Benchmarking llms on multi-turn and multilingual instructions following. *arXiv preprint arXiv:2410.15553*.

Tingfeng Hui, Lulu Zhao, Guanting Dong, Yaqi Zhang, Hua Zhou, and Sen Su. 2024. Smaller language models are better instruction evolvers. *arXiv preprint arXiv:2412.11231*.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2023. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*.

Julian Katz-Samuels, Zheng Li, Hyokun Yun, Priyanka Nigam, Yi Xu, Vaclav Petricek, Bing Yin, and Trishul Chilimbi. 2024. Evolutionary contrastive distillation for language model alignment. *arXiv preprint arXiv:2410.07513*.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. 2024. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. 2025. Small models struggle to learn from strong reasoners. *arXiv preprint arXiv:2502.12143*.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Meta. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. 2023. No robots. https://huggingface.co/datasets/HuggingFaceH4/no_robots.

Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3505–3506.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024a. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*.

Yuchong Sun, Che Liu, Kun Zhou, Jinwen Huang, Ruihua Song, Xin Zhao, Fuzheng Zhang, Di Zhang, and Kun Gai. 2024b. Parrot: Enhancing multi-turn instruction following for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9729–9750, Bangkok, Thailand. Association for Computational Linguistics.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

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

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, and 1 others. 2024. Livebench: A challenging, contamination-free llm benchmark. *arXiv preprint arXiv:2406.19314*.

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, and 43 others. 2024. Qwen2 technical report. *Preprint*, arXiv:2407.10671.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Dylan Zhang, Qirun Dai, and Hao Peng. 2025. The best instruction-tuning data are those that fit. *arXiv preprint arXiv:2502.04194*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

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

A Experimental Setup

A.1 Datasets and Baselines

A.1.1 Datasets

ShareGPT [1] is an open-source and multi-turn conversation dataset that AutoIF utilizing the official open-source code[2].

OpenHermes (Teknium, 2023) is a large-scale, diverse, and high-quality compilation consisting of around 1M synthetically generated instruction and chat samples. We randomly select a subset of 150K instructions from OpenHermes-2.5 for scaling experiment.

No Robots (Rajani et al., 2023) is a high-quality dataset of 10k instructions and demonstrations created by skilled human annotators. We use all instructions from No Robots for scaling experiment.

A.1.2 Baselines

Conifer (Sun et al., 2024a) curates a novel instruction tuning dataset which aims to enhance how LLMs, particularly open-source models, follow complex instructions involving multiple, intricate constraints. We generate responses for the public data using LLaMA-3.1-70B-Instruct[3].

Evol-Instruct (Xu et al., 2023) automatically mass-produces high-complexity training data by generating diverse instructions with varying difficulty levels. We sample 10k data and generate responses[4].

SPaR (Cheng et al., 2024) proposes a self-play framework to enhance instruction-following abilities for LLMs, where an LLM refines its own responses via tree-search. We reimplement SPaR utilizing its official open-source dataset[5].

A.2 Evaluation Benchmarks

IFEval (Zhou et al., 2023) is an easy-to-produce benchmark designed to evaluate the instruction-following capability of LLMs. IFEval constructs around 500 prompts that contain 25 types of verifiable instructions. We use both loose and strict accuracy metrics at prompt and instruction levels in our evaluation.

Multi-IF (He et al., 2024) is a benchmark designed to assess LLMs' proficiency in following multi-turn and multilingual instructions. Based on IFEval, Multi-IF contains 4,501 multilingual conversations, where each has three turns. We report the average accuracy across all languages for each of the three rounds in the experiment.

InfoBench (Qin et al., 2024) is a benchmark comprising 500 diverse instructions and 2,250 decomposed questions across multiple constraint categories, adopting a new metric Decomposed Requirements Following Ratio (DRFR) for evaluating LLM's ability to follow instructions. We use GPT-4-1106-preview as the evaluator in our assessments.

FollowBench (Jiang et al., 2023) is a multi-level fine-grained constraints following benchmark for LLMs. FollowBench incorporates five distinct fine-grained constraint types (Content, Situation, Style, Format, and Example) and underscores multi-level mechanism when building instruction prompts. In our experiment, we prompt the GPT-4-1106-preview to assess whether LLM's outputs have satisfied each individual constraint.

LiveBench (White et al., 2024) is a LLM benchmark that contains a wide variety of challenging tasks (math, coding, reasoning, language, instruction-following, and data analysis) and automatically scores answers according to objective ground-truth values. When assessing the instruction-following skills of LLMs, we employ the instruction-following subset, while the entire dataset is utilized to gauge their overall capabilities.

GSM8K (Cobbe et al., 2021) comprises 8.5K high-quality, multilingual grade school math word problems, specifically designed to assess the multi-step mathematical reasoning proficiency of language models. We report the overall accuracy in the experiment.

HumanEval (Chen et al., 2021) consists of 164 programming problems with function signatures, docstrings, bodies, and unit tests, averaging 7.7 tests per problem. It is utilized to evaluate the coding abilities of LLMs. HumanEval assesses the capability of LLMs in program synthesis from docstrings, testing language comprehension, reasoning, algorithms, and elementary mathematics skills. We report Pass@1 on HumanEval in the experiment.

BBH (Suzgun et al., 2022) is a clean, challenging and tractable subset benchmark filtered from Big Bench, which includes 23 types of difficult tasks and 6,511 evaluation examples in total. BBH primarily assesses the models' reasoning capacities and problem-solving skills comprehensively. In the experiment we report the accuracy metrics.

[1] https://huggingface.co/datasets/shibing624/sharegpt_gpt4

[2] https://github.com/QwenLM/AutoIF

[3] https://huggingface.co/datasets/ConiferLM/Conifer

[4] https://huggingface.co/datasets/WizardLMTeam/WizardLM_evol_instruct_70k

[5] https://huggingface.co/datasets/CCCCCC/SPaR

Arena Hard (Li et al., 2024) is an automatic LLM benchmark consisting of 500 challenging challenging user queries, which is curated to evaluate the comprehensive performance of LLMs in user dialogue scenarios. In the experiment, we adopt GPT-4-1106-preview as the judge model and report the win rate of our models against the baseline model (GPT-4-0314).

A.3 Training Strategies

ULTRAIF offers flexible training strategies for aligning model with instruction following capabilities. To thoroughly evaluate the effectiveness, we provide two approaches:

Supervised Finetuing (SFT). Given the dataset $\mathcal{D}_{data}$, we apply standard Supervised Finetuning (SFT) objective on vanilla model π with parameters θ, as shown in Eq. 4:

$$\mathcal{L}_{SFT}(\pi_\theta) = \sum_{(\bar{x},y_c)\in\mathcal{D}_{data}} \log\ \pi_\theta(y_c|\bar{x}) \quad (4)$$

where $\bar{I}$ represents the augmented instruction, and r_c denotes the corresponding chosen response.

SFT + Iterative Online DPO. As ULTRAIF is equipped with evaluation questions, it facilitates quality control by enabling the generation of pairwise responses with varying quality levels. This property makes it particularly suitable for the application of Direct Perference Optimization (DPO, Rafailov et al. (2024)) to refine the fine-tuned model, π_{ref}. The DPO objective is formulated as Eq. 5:

$$\mathcal{L}_{DPO}(\pi_\theta, \pi_{ref}) = -\mathbb{E}_{(\bar{x},y_c,y_r)\in\mathcal{D}_{data}} \log\ \sigma(\beta\ \cdot \Delta)$$
$$\Delta = (\log \frac{\pi_\theta(y_c|\bar{x})}{\pi_{ref}(y_c|\bar{x})} - \log \frac{\pi_\theta(y_r|\bar{x})}{\pi_{ref}(y_r|\bar{x})}) \quad (5)$$

where β is a scaling hyperparameter, σ denotes the sigmoid function, and π_θ is initialized from π_{ref} and further optimized during the DPO stage.

In the context of ULTRAIF, the *UltraComposer* enables an iterative augmentation of instructions, transitioning from simpler to more complex tasks. This allows the DPO process to be formulated as an iterative curriculum. At each iteration, the model π_{ref} is replaced with the latest optimized model from the previous stage. Concurrently, more challenging instruction-following datasets are generated and utilized for further training. This iterative approach ensures continuous improvement in model performance and adaptability across increasingly complex scenarios.

Moreover, during the iterative process, as observed by (Chen et al., 2024), the DPO objective primarily focuses on optimizing the margin between the chosen and rejected samples, rather than directly maximizing the probability of chosen samples and minimizing that of the rejected ones. To address this, we employ the Noise Contrastive Estimation (NCA, Chen et al. (2024)) loss in the final iteration, and the objective is defined in Eq. 6:

$$\mathcal{L}_{NCA}(\pi_\theta, \pi_{ref}) = -\mathbb{E}_{(\bar{x},y_c,y_r)\in\mathcal{D}_{data}} \Big[\log\ \sigma(\beta \log \frac{\pi_\theta(y_c|\bar{x})}{\pi_{ref}(y_c|\bar{x})}) + \frac{1}{2} \sum_{y\in\{y_c,y_r\}} \log\ \sigma(-\beta \log \frac{\pi_\theta(y|\bar{x})}{\pi_{ref}(y|\bar{x})}) \Big] \quad (6)$$

A.4 Implementation Details

Our experiments are conducted on 8×A100 GPUs (80GB) using mixed precision with bf16, DeepSpeed ZeRO Stage 3 (Rasley et al., 2020), and FlashAttention 2 (Dao, 2023).

In the SFT stage, we perform full fine-tuning with a learning rate of 1e-5. The maximum token length is set to 8192 and variable-length packing is enabled. We use AdamW (Loshchilov, 2017) as the optimizer with a warmup ratio of 0.03 and employ a LinearLR scheduler at the beginning, transitioning to CosineAnnealingLR towards the end.

In the DPO stage, the configuration is similar, with the only difference being a lower learning rate of 5e-7. Additionally, the beta parameter of DPO loss is set to 0.1.

B Prompts of ULTRAIF

To train our ***UltraComposer***, we prompt LLM to perform **Instruction Decomposition** and **Eval Question Generation**.

We use the following prompt template to decompose human instructions:

Prompt Template of Instruction Decomposition

You are an expert in extracting instruction constraints from a given query.
Definition of Constraint: The smallest unit of restriction or requirement that the instruction imposes on the task.

Query: {query}

- If the query is not a question, or is simple or straightforward without any constraints, please only respond with the following JSON, indicating that no constraints are present.

```
{
    "Complex": False
}
```

- If constraints are present, follow these steps:
  1. Identify the Basic Query: Clearly understand the primary goal of the query, stripping away any constraints. The Basic Query should be the essential task without any added conditions or restrictions.
  2. Extract and Categorize Constraints: Identify and classify constraints based on the following types:
     - Content Constraints:
       * Specific Terms or Symbols: Mandatory use of certain terms or symbols with their exact placement (e.g., must include the word 'beautiful').
       * Required Elements or Concepts: Mandates for including specific elements or concepts in responses, reflecting a scenario or object (e.g., highlights the Great Wall).
       * Thematic Directives: Instructions related to thematic content, perspective, or tone, emphasizing response significance (e.g., write a poem about London).
     - Numerical Constraints:
       * Constraints on quantities related to the content, such as the number of points, sentences, paragraphs, response length, or examples (e.g., within a single paragraph with three sentences).
     - Stylistic Constraints:
       * Desired tone and style for the response (e.g., formal, informal, conversational).
       * Specific language or terminology to be used or avoided (e.g., encyclopedic style).
     - Format Constraints:
       * Required structure or format for the response (e.g., list, JSON, bullet points, Java language).
       * Presentation styles or formatting requirements (e.g., electronic medical record format).
     - Linguistic Constraints:
       * Language use in specific contexts, such as discourse, dialects, sociolects, and language policies (e.g., in English).
       * Sentence structure, including phrases, constituents, and the use of imperatives (e.g., with nouns and verbs).
       * Internal structure of words, including roots, affixes, and morphological changes (e.g., lowercase, single-rhyme).
  3. Response Format:
     - Do not consider details that are part of the content itself, such as those used in descriptions, scenarios, or examples, unless they directly impose a restriction of the response.
     - The Basic Query should represent the queryâĂŹs core goal, free from any constraints.

- Ensure that extracted constraints do not overlap with the Basic Query.
- Present each constraint as a dictionary within a list, where each dictionary contains:
  * 'constraint': The specific restriction or requirement.
  * 'simplified query': The query after removing this constraint, polished for coherence and correctness.
- Exclude any constraint types not present in the query.

```
{
    "Complex": True,
    "Basic Query": ...,
    "Content Constraints": [
        {
            "constraint": "...",
          "simplified query": "..."
        },
        {
            "constraint": "...",
          "simplified query": "..."
        },
    ],
    ...
}
```

Please only provide the response in JSON format.

We use the following prompt template to generate evaluation questions for instructions:

Prompt Template of Eval Question Generation

You are an expert in crafting questions to evaluate whether a response to a query adheres to specific constraints.
For the given constraint, please design a question that human evaluators can use to assess if the response meets the specified constraint. The question should focus solely on the given constraint and not other constraints present in the original query.
Specifically, if the given constraint is meaningless or is a part of the content itself, such as those used in descriptions, scenarios, or examples, you can respond with an empty string.
Query: {query}
Constraint: {constraint}
Please design a question for the specified constraint for the given query, and respond in the JSON format without explanation.

```
{
    "question": "string",
}
```

We use the following template to train ULTRAIF

Prompt Template of UltraComposer

Input:
[history]: ...
[initial query]: ...
Output:

```
{
    "augmented query": ..,
    "question": ...
}
```

For **Generate-then-Evaluate** process, we prompt LLM to perform **Response Generation** and **Response Evaluation**.

First we use the following prompt template to generate responses for the augmented instructions:

Prompt Template of Response Generation

You are an expert tasked with answering the given query. Please provide a clear and concise response directly, without introductory phrases such as 'What a great question,' 'Here is the answer,' or similar expressions. Focus solely on addressing the query.
Now please answer the given query while strictly following its inside constraints.
[Query] {query}

Then we use the following prompt template to evaluate the quality of those generated responses:

Prompt Template of Response Evaluation

You are an expert that is good at judging whether the response to a given query meets the specified evaluator questions.

Your task is to carefully examine the response to determine if it adheres to each requirement outlined in the evaluator questions.
[Query] {query}
[Response] {response}
[Evaluator Question] {question}
For each question, please provide a justification for your evaluation, explaining how the response does or does not satisfy the criteria and a score (`'YES'` or `'NO'`) indicating whether the answer satisfies each constraint. You should only respond in the following JSON format:

```
{
    "Question 1": {
        "explanation": "",
        "score": "YES" or "NO"
    },
    "Question 2": {
        "explanation": "",
        "score": "YES" or "NO"
    },
}
```

C Additional Experimental Results

C.1 Analysis of Sampling Efficiency

Moreover, Table 7 further compare the pass rates during dataset synthesis, where UltraIF demonstrates substantial improvements over AutoIF. Specifically, UltraIF achieves an SFT pass rate of 85% and a DPO pass rate of 60%, compared to only 20% and 26%, respectively, for AutoIF. This indicates that for generating an equivalent amount of data, UltraIF reduces costs by a factor of three to five. Furthermore, during the rejection sampling stage, while AutoIF necessitates rigorous function-based filtering for instruction synthesis and response generation, UltraIF achieves this with a single LLM call, making it far more scalable and cost-efficient.

Method	SFT Pass Rate	DPO Pass Rate
AutoIF	20%	26%
UltraIF	85%	60%

Table 7: The overall pass rate of data synthesis.

C.2 Analysis of Multi-Turn Data

Building on prior work that emphasizes enhancing multi-turn instruction-following capabilities (Sun et al., 2024b; He et al., 2024), our analysis reveals that incorporating multi-turn data during the SFT stage significantly improves UltraIF's performance across various benchmarks. As shown in Table 8, the inclusion of multi-turn data results in performance gains of 3.01% on Multi-IF, 1.18% on InfoBench, and 5.10% on LiveBench, compared to the baseline SFT model without such data. These improvements highlight the critical role of multi-turn interactions in training, allowing the model to better understand conversational context and dependencies, thereby enhancing its instruction-following capabilities. Therefore, we incorporate multi-turn data in our scaling experiments.

Method	Multi-IF	InfoBench	LiveBench
UltraIF + SFT	40.12	77.78	46.60
w. multi turn	43.13	79.86	54.20
Δ	+3.01	+1.18	+5.10

Table 8: The performance comparison of incorporating multi-turn data during the SFT stage.

C.3 Case Study

By modeling the distribution of real-world instructions, UltraIF supports effective instruction augmentation while minimizing inconsistencies between newly added constraints and the original instructions. Thus, UltraIF eliminates the need to verify whether the constraints are consistent with the original instructions (Dong et al., 2024; Katz-Samuels et al., 2024). Additionally, the evaluation questions take over a separate score-filtering stage. Consequently, UltraIF achieves greater efficiency and incurs minimal costs when constraining large-scale datasets.

Table 9 shows some examples of augmented instructions and evaluation questions generated by UltraIF. The original queries come from ShareGPT dataset.

Original Query	Augmented Instruction	Eval Question
Explain merkle tree in blockchain.	Explain merkle tree in blockchain **to a 10 years old**.	Is the explanation of a Merkle tree in the context of blockchain presented in a way that a 10-year-old can understand?
We are driving in a car. It is cold outside, windshield is frozen, and we are on a secluded road to a small village. This is scene from a horror movie, and it's just starting. Describe a scene in great detail.	We are driving in a car. It is cold outside, windshield is frozen, and we are on a secluded road to a small village. This is scene from a horror movie, and it's just starting. Describe a scene in great detail, and **write it in the style of a gothic horror author**.	Does the response evoke a dark, eerie, and ominous atmosphere, characteristic of gothic horror?
Design a html form with form tags.	Design a html form with form tags **for the following 3 user inputs: first_name, last_name, date_of_birth**.	Does the HTML form include form tags for exactly three user inputs: first_name, last_name, and date_of_birth?
I'm planning to visit Okinawa Japan from April 7th to April 10th. Do you have any recommendation on what to do while I'm there?	I'm planning to visit Okinawa Japan from April 7th to April 10th. Do you have any recommendation on what to do while I'm there? **I'd like to focus on nature, food, and local culture.**	Does the response recommend activities in Okinawa that focus on nature, food, and local culture?
What is the meaning of life?	What is the meaning of life? **Explain it in 5 paragraphs**.	Is the response to the question explained in exactly 5 paragraphs?
Write a homepage for translation business.	Write me a homepage for translation business **in wordpress**.	Is the homepage for the translation business designed using WordPress?

Table 9: Examples of ULTRAIF's data pair.