AIR: Complex Instruction Generation via Automatic Iterative Refinement

Wei Liu[1*], **Yancheng He**[1*], **Yu Li**[2*], **Hui Huang**[1*✉], **Chengwei Hu**[1], **Jiaheng Liu**[3], **Shilong Li**[1], **Wenbo Su**[1], **Bo Zheng**[1]

[1]Alibaba Group, [2]Beijing Jiaotong University, [3]Nanjing University

liuweihit2023@163.com, hh456524@alibaba-inc.com

Abstract

With the development of large language models, their ability to follow simple instructions has significantly improved. However, adhering to complex instructions remains a major challenge. Current approaches to generating complex instructions are often irrelevant to the current instruction requirements or suffer from limited scalability and diversity. Moreover, methods such as back-translation, while effective for simple instruction generation, fail to leverage the rich knowledge and formatting in human written documents. In this paper, we propose a novel **A**utomatic **I**terative **R**efinement (**AIR**) framework to generate complex instructions with constraints, which not only better reflects the requirements of real scenarios but also significantly enhances LLMs' ability to follow complex instructions. The AIR framework consists of two stages: 1) Generate an initial instruction from a document; 2) Iteratively refine instructions with LLM-as-judge guidance by comparing the model's output with the document to incorporate valuable constraints. Finally, we construct the AIR-10K dataset with 10K complex instructions and demonstrate that instructions generated with our approach significantly improve the model's ability to follow complex instructions, outperforming existing methods for instruction generation[1].

1 Introduction

Recent advancements in Large Language Models (LLMs) have shown impressive performance across a wide range of tasks (Zhao et al., 2023; Li et al., 2024a; He et al., 2024b). Driven by vast amounts of data and efficient training, most current LLMs are capable of effectively following user instructions and aligning to a certain extent with human preferences (Ouyang et al., 2022; Li et al., 2024b; Huang et al., 2025). However, despite these successes, they still face significant challenges when it comes to following complex instructions (Jiang et al., 2023; Wen et al., 2024).

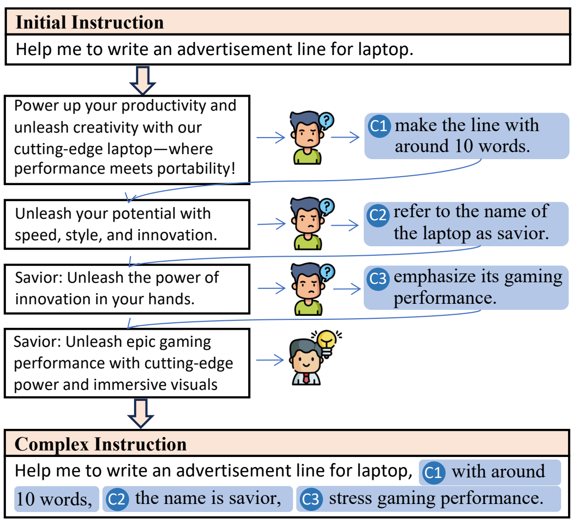

Initial Instruction
Help me to write an advertisement line for laptop.
Power up your productivity and unleash creativity with our cutting-edge laptop—where performance meets portability!
C1 make the line with around 10 words.
Unleash your potential with speed, style, and innovation.
C2 refer to the name of the laptop as savior.
Savior: Unleash the power of innovation in your hands.
C3 emphasize its gaming performance.
Savior: Unleash epic gaming performance with cutting-edge power and immersive visuals
Complex Instruction
Help me to write an advertisement line for laptop, C1 with around 10 words, C2 the name is savior, C3 stress gaming performance.

Figure 1: Illustration of how humans iteratively refine instructions to be more complex.

Existing datasets of complex instructions primarily originate from two sources: 1) Curated data from open-source datasets or human annotations (Zhou et al., 2024a; Zhang et al., 2024), which are resource-intensive and **lack scalability**, and 2) Transforming simple instructions into complex ones automatically using proprietary LLMs (Xu et al., 2023; Sun et al., 2024). While the automatic transformation improves scalability, the generated constraints are often recombinations of few-shot examples, resulting in **limited diversity**. Moreover, these constraints may have **low relevance** to the target output, failing to reflect real-world scenarios.

Recently, back-translation, which involves translating text from the target side back into the source side, has been proposed to generate scalable and diverse instructions from human-written corpora (Sennrich, 2015; Hoang et al., 2018; Zheng et al., 2024a; Li et al., 2023). However, these methods typically focus on generating **simple instruc-**

[1]Codes and data are available at https://github.com/WeiLiuAH/AIR-Automatic-Iterative-Refinement.

tions and have not fully explored the rich knowledge contained in the human corpus.

In this paper, we propose an **Automatic Iterative Refinement (AIR)** framework for generating high-quality complex instructions. Specifically, our approach is based on two key observations. First, human-written documents contain massive human preferences that can be converted into specific constraints, such as formatting conventions in legal documents. Second, humans often refine complex instructions iteratively based on feedback from model outputs. As illustrated in Figure 1, simple instructions are progressively adjusted and enriched to better align with human preferences. This iterative process plays a critical role in crafting effective complex instructions.

Therefore, our AIR framework incorporates document-based knowledge and LLM-as-judge to iteratively construct complex instructions. The framework consists of two key steps: 1) **Initial Instruction Generation**, where the model generates initial instructions based on the document content; 2) **Iterative Instruction Refinement**, where instructions are iteratively refined with LLM-as-judge guidance by comparing model outputs with the document, to identify and incorporate valuable constraints. This process enables the framework to generate more challenging instructions that align more closely with real-world scenarios.

In summary, our contributions are as follows:

- To better align with real-world scenarios, we propose the **AIR** framework, which iteratively refines complex instructions with LLM-as-judge guidance by comparing with the document.
- We introduce a novel instruction dataset, **AIR-10K**, generated using our framework. Experimental results demonstrate that our fine-tuned model significantly outperforms existing methods on instruction-following benchmarks.
- We provide a comprehensive experimental analysis to evaluate the individual components of our framework, validating the contribution of each step to the overall improvement.

2 Related Work

2.1 Instruction Generation

Instruction tuning is essential for aligning Large Language Models (LLMs) with user intentions (Ouyang et al., 2022; Cao et al., 2023). Initially, this involved collecting and cleaning existing data, such as open-source NLP datasets (Wang et al., 2023; Ding et al., 2023). With the importance of instruction quality recognized, manual annotation methods emerged (Wang et al., 2023; Zhou et al., 2024a). As larger datasets became necessary, approaches like Self-Instruct (Wang et al., 2022) used models to generate high-quality instructions (Guo et al., 2024). However, complex instructions are rare, leading to strategies for synthesizing them by extending simpler ones (Xu et al., 2023; Sun et al., 2024; He et al., 2024a). Nevertheless, existing methods struggle with scalability and diversity.

2.2 Back Translation

Back-translation, a process of translating text from the target language back into the source language, is mainly used for data augmentation in tasks like machine translation (Sennrich, 2015; Hoang et al., 2018). Li et al. (2023) first applied this to large-scale instruction generation using unlabeled data, with Suri (Pham et al., 2024) and Kun (Zheng et al., 2024a) extending it to long-form and Chinese instructions, respectively. Nguyen et al. (2024) enhanced this method by adding quality assessment to filter and revise data. Building on this, we further investigated methods to generate high-quality complex instruction datasets using back-translation.

3 Approach

Our approach mainly consists of two steps: 1) Initial Instruction Generation; 2) Iterative Instruction Refinement, as shown in Figure 2. In this section, we will introduce the two steps in detail.

3.1 Initial Instruction Generation (IIG)

Document Collection. Traditional instruction generation methods such as Self-Instruct (Wang et al., 2022) often suffer from limited diversity, as the generated instructions are generally recombinations of the provided few-shot examples. Inspired by Li et al. (2023), we generate initial instructions using back translation based on human-written documents.

To further enhance the diversity of the generated instructions, we implement a density-based sampling mechanism for documents, as shown in Algorithm 1. Specifically, we convert documents into vector representations using Sentence-Transformers[1], and perform sampling to maximize the density of samples in the representation space.

[1] `sentence-transformers/all-MiniLM-L6-v2`.

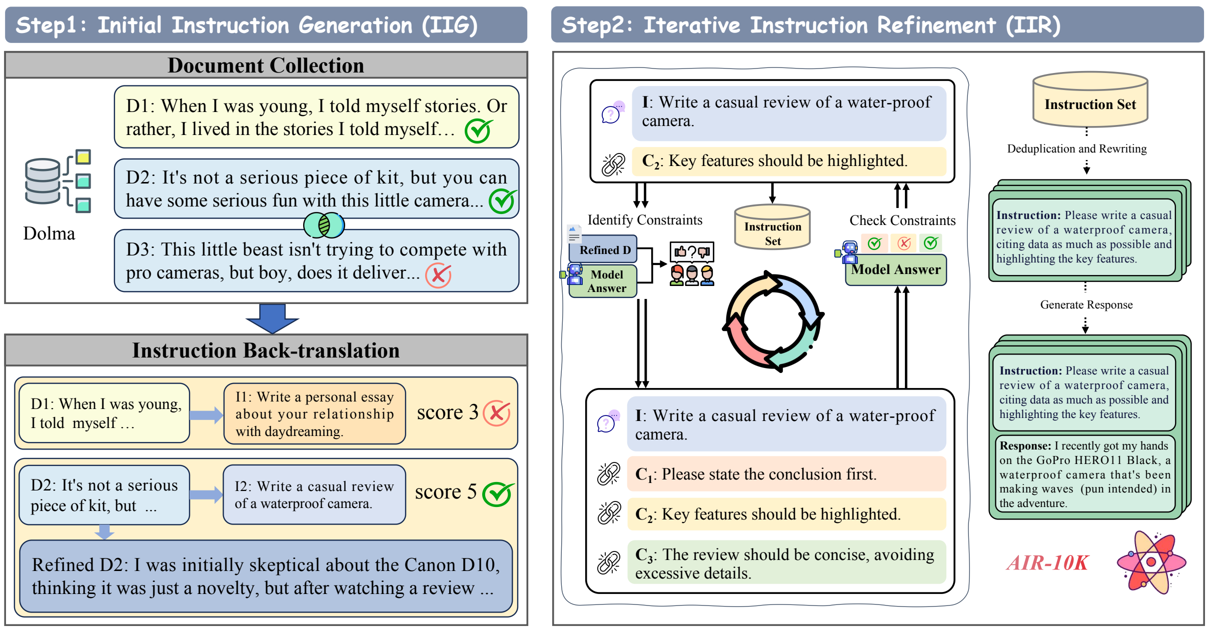

Step1: Initial Instruction Generation (IIG)
Document Collection
Dolma
D1: When I was young, I told myself stories. Or rather, I lived in the stories I told myself…
D2: It's not a serious piece of kit, but you can have some serious fun with this little camera...
D3: This little beast isn't trying to compete with pro cameras, but boy, does it deliver...
Instruction Back-translation
D1: When I was young, I told myself …
I1: Write a personal essay about your relationship with daydreaming.
score 3
D2: It's not a serious piece of kit, but …
I2: Write a casual review of a waterproof camera.
score 5
Refined D2: I was initially skeptical about the Canon D10, thinking it was just a novelty, but after watching a review …
Step2: Iterative Instruction Refinement (IIR)
I: Write a casual review of a water-proof camera.
C2: Key features should be highlighted.
Identify Constraints
Instruction Set
Check Constraints
Refined D
Model Answer
Model Answer
I: Write a casual review of a water-proof camera.
C1: Please state the conclusion first.
C2: Key features should be highlighted.
C3: The review should be concise, avoiding excessive details.
Instruction Set
Deduplication and Rewriting
Instruction: Please write a casual review of a waterproof camera, citing data as much as possible and highlighting the key features.
Generate Response
Instruction: Please write a casual review of a waterproof camera, citing data as much as possible and highlighting the key features.
Response: I recently got my hands on the GoPro HERO11 Black, a waterproof camera that's been making waves (pun intended) in the adventure.
AIR-10K

Figure 2: **AIR**: Automatic Iterative Refinement Framework.

Algorithm 1 Density-based Sampling

```
Input: Instruction Dataset $D$ with $m$ samples, number of samples to select $n$.
Output: Selected Dataset $D'$ with $n$ samples.
1: Derive the embeddings for each sample in $D$.
2: Randomly sample one data point $x$ from $D$.
3: Delete $x$ from $D$, add $x$ to $D'$.
4: for $i = 1, 2, \ldots, n-1$ do
5:     Calculate the cosine similarity score between $x$ and each sample from $D$.
6:     Select the least similar sample $x'$ from $D$.
7:     Let $x = x'$.
8:     Delete $x$ from $D$, add $x$ to $D'$.
9: end for
```

In this way, we effectively eliminate redundant documents, enhancing the diversity of instructions. Moreover, this approach ensures that the knowledge introduced during instruction fine-tuning is evenly distributed across various domains. This not only prevents the model from overfitting to a specific domain but also mitigates the risk of catastrophic forgetting of fundamental capabilities[2].

Moreover, to further ensure the quality of the document collection, we filter out documents based on the following criteria: 1) Length: Documents with fewer than 50 words or exceeding 2,048 words are removed. 2) Symbol-to-text ratio: Documents where the proportion of symbols exceeds that of textual content are excluded. 3) Redundancy: Documents containing repetitive paragraphs or excessive symbol repetitions are eliminated.

Instruction Back-translation Based on the sampled documents, we employ the back-translation method to construct initial instructions. Specifically, we utilize a guidance model to predict an instruction which can be accurately answered by (a portion of) the document[3]. This enables the generation of new instructions without relying on few-shot examples or pre-designed rules. Moreover, we can further ensure the diversity of the generated instructions by diversifying the documents.

However, although constructed from documents, instructions do not always align well with them in two key respects (Nguyen et al., 2024). First, the document is unstructured and does not follow the AI-assistant format. Second, it may contain content irrelevant to the instruction. Therefore, we introduce an additional refinement step to transform the document into response format and remove irrelevant content.

To further ensure the quality of the instructions, we introduce a scoring step to filter out low-quality data. Each instruction is assigned a score on a scale of 1 to 5 by the guidance model, with each point corresponding to a specific aspect. Only instructions with a score greater than (or equal to) 4 are retained for the next step[4].

[2]The effectiveness of this density-based sampling approach is demonstrated in Appendix **??**.

[3]Detailed prompt templates are presented in Appendix B.2.

[4]Instruction score criteria are presented in Appendix B.3.

Algorithm 2 Iterative Instruction Refinement

```
Input: Guidance model $M$, current model $m$, refined document $R$, initial instruction $I_0$.
Output: Constraint Sets $C_n$ and $C'_n$.
1: for $i = 1, 2, \ldots, n$ do
2:     Use $m$ to generate a response $A_i$ for the previous instruction $I_{i-1}$.
3:     Leverage $M$ as the judge, compare $A_i$ with $R$ to identify a new constraint $c_i$.
4:     Add $c_i$ to $C_n$.
5:     Add $c_i$ to $I_{i-1}$ to form a new instruction $I_i$.
6:     Use $m$ to generate a response $A'_i$ for $I_i$.
7:     Leverage $M$ as the judge, check whether $A'_i$ satisfies constraint $c_i$. If not, add $c_i$ to $C'_n$.
8: end for
```

3.2 Iterative Instruction Refinement (IIR)

To enhance a model's ability to follow complex instructions, it is crucial to construct complex instruction data that incorporates multiple constraints. Previous methods typically start with simple instructions and generate complex ones through rewriting or recombination (Xu et al., 2023). However, the constraints generated in this way often do not meet actual needs or lack diversity.

An effective sample for complex instruction fine-tuning should adhere to two key principles:

1. Whether the model's response originally misaligns with the constraint before it is added;
2. Whether the model's response still misaligns with the constraint after it is added.

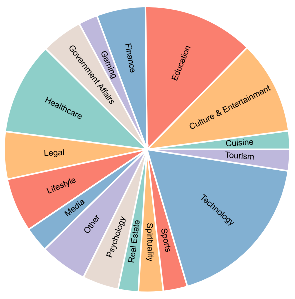
These constraints highlight the model's weaknesses in handling complex instructions and require further improvement. Conversely, if a constraint does not meet these principles, it indicates that the constraint falls within the model's current capabilities and does not require additional learning.

Therefore, we introduce constraint generation using LLM-as-judge guidance (Zheng et al., 2023), which mimics the human process of iteratively refining prompts to form complex instructions[5]. As shown in Algorithm 2, during the process of iteration, we obtain the constraints that the model fails to satisfy, which require further fine-tuning.

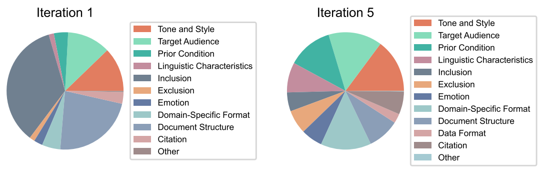
Throughout this process, as the number of constraints increases, the model's response also improves, making the identification of new constraints more challenging. To uncover constraints that better reflect human preferences, we use the refined document as the reference answer for the judgment process. Human-written documents inherently contain vast amounts of knowledge and formatting conventions that reflect human preferences. Therefore, the derived constraints will also align more closely with human preferences.

Finally, the constraint set is merged into a new instruction. Note that two constraint sets are derived: the first set C_n satisfies Principle 1, while the second set C'_n, which includes an additional checking step, satisfies both Principle 1 and 2.

While we leverage the refined document as the reference for the judgment process, it should not be used as the target for fine-tuning as in Nguyen et al. (2024), as the document is not refined with the constraints presented explicitly. Therefore, we leverage the guidance model to re-generate the response based on the combined instruction[6].



(a) Distribution of domains



(b) Distribution of constraint types in iteration 1 and 5

Figure 3: Data statistics of AIR-10K.

3.3 Data Statistics of AIR-10K

With our proposed framework, we constructed a high-quality complex instruction dataset, **AIR-10K**, based on openly available documents. We present the real-life scenario-specific domain distribution of AIR-10K in Figure 3(a). As can be seen, our dataset encompasses nearly 20 different domains in total, demonstrating a high degree of balance across diverse fields. Furthermore, we present the distribution of constraint types during iteration

[5]The analysis of potential biases in the LLM-as-Judge approach are presented in Appendix A.1.

[6]A detailed example illustrating the complete pipeline is provided in Appendix A.6.

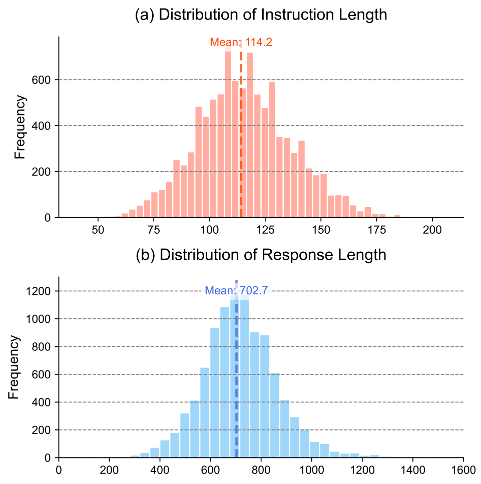


Figure 4: Length distribution of AIR-10K.

1 and 5 in Figure 3(b). It is evident that in iteration 1, *Inclusion* and *Document Structure* constraints dominate. However, after four rounds of constraint additions, by iteration 5, the proportions of each constraint type become more uniform[7].

We also analyze the length distributions of both instructions and responses. As shown in Figure 4(a) and 4(b), our instructions are of substantial information for capturing complex tasks.

4 Experiments

4.1 Set-up

Data. Following Nguyen et al. (2024), we utilize a subset of Dolma v1.7 (Soldaini et al., 2024) as the document source, which is derived from a collection of web pages and has undergone rigorous quality and content filtering to ensure data quality.

Models. We apply our method to two models, Llama-3-8B and Qwen2.5-7B, and we apply preliminary supervised fine-tuning for both models. The preliminary fine-tuning process is conducted on two general instruction datasets, namely ultrachat-200k (Ding et al., 2023) and tulu-330k (Lambert et al., 2024), respectively. For the guidance model to construct the data, we rely on a larger model with the same group to ensure data quality, namely Qwen-2.5-72B-Instruct for Qwen-2.5-7B, and Llama-3-70B-Instruct for Llama-3-8B. We set the maximum number of iterations to 5.

Evaluation. We mainly conduct evaluations on two complex instruction-following benchmarks, **CFBench** (Zhang et al., 2024) and **FollowBench** (Jiang et al., 2023), where instructions consist of multiple constraints. We also conduct evaluations on a general instruction benchmark of **AlpacaEval2** (Dubois et al., 2024). Note that all benchmarks require GPT-4 for judgment, and we use GPT-4o-0806 [8] as the evaluator for all of them. We also conduct evaluations on fundamental capability benchmarks, including math, code, and knowledge tasks, and the results are presented in Appendix A.2 due to space limitations.

Baselines. We mainly compare our method with four groups of methods as follows:

1. **Human-crafted instruction data**: This includes ShareGPT[9], which is a collection of real human-AI conversations.
2. **Automatically crafted general instruction data**: This includes Self-Instruct (Wang et al., 2022), which leverages few-shot examples to self-generate simple instruction samples.
3. **Automatically rewritten complex instruction data**: This includes Evol-Instruct (Xu et al., 2023), ISHEEP (Liang et al., 2024), Muffin (Lou et al., 2023) and Conifer (Sun et al., 2024), which initiate with simple instructions and progressively construct more complex ones through rewriting or recombination.
4. **Automatically back-translated complex instruction data**: This includes Suri (Pham et al., 2024) and Crab (Qi et al., 2024), which curate the complex instructions and constraints by back-translating the pre-existing response. These methods are the closest to our work.

We also compare with the original back-translation and back-and-forth translation (Cao et al., 2023; Nguyen et al., 2024), where IIR is skipped and initial instructions are directly used.

For all constructed datasets, we sample 10k instruction-response pairs for supervised fine-tuning under the same hyper-parameters[10].

Note that, due to space limitations, some results and analysis are presented in Appendix A.

4.2 Main Results

As shown in Tables 1 and 2, our proposed method achieves the best performance on both complex and general instruction-following benchmarks, demonstrating its effectiveness. In contrast, automatically

[7]The constraint type definition and complete distributions across all iterations are detailed in Appendix A.7.

[8]`platform.openai.com/docs/models/gp#gpt-4o`

[9]`huggingface.co/datasets/anon8231489123/ShareGPT_Vicuna_unfiltered`

[10]Detailed hyper-parameters are presented in Appendix B.1.

Fine-tuned on Llama-3-8B-UltraChat							
Method	**CF-Bench**			**FollowBench**		**AlpacaEval2**	
	CSR	**ISR**	**PSR**	**HSR**	**SSR**	**LC.**	**Len**
Baseline	0.51	0.15	0.22	41.04	57.39	8.86	1,017
back-translation	0.40$_{-0.11}$	0.11$_{-0.04}$	0.15$_{-0.07}$	21.19$_{-19.85}$	33.92$_{-23.47}$	0.96$_{-7.90}$	2,966
back-and-forth	0.58$_{+0.07}$	0.20$_{+0.05}$	0.27$_{+0.05}$	44.65$_{+3.61}$	61.58$_{+4.19}$	10.06$_{+1.20}$	1,440
ShareGPT	0.62$_{+0.11}$	0.22$_{+0.07}$	0.32$_{+0.10}$	40.99$_{-0.05}$	58.59$_{+1.20}$	8.36$_{-0.50}$	1,052
Self-Instruct	0.34$_{-0.17}$	0.08$_{-0.07}$	0.10$_{-0.12}$	12.33$_{-28.71}$	26.92$_{-30.47}$	2.76$_{-6.10}$	384
Evol-Instruct	0.57$_{+0.06}$	0.22$_{+0.07}$	0.28$_{+0.06}$	43.58$_{+2.54}$	59.21$_{+1.82}$	7.15$_{-1.71}$	903
MUFFIN	0.50$_{-0.01}$	0.16$_{+0.01}$	0.22$_{+0.00}$	30.88$_{-10.16}$	48.48$_{-8.91}$	4.51$_{-4.35}$	791
Conifer	0.57$_{+0.06}$	0.22$_{+0.07}$	0.28$_{+0.06}$	47.06$_{+6.02}$	61.32$_{+3.93}$	12.81$_{+3.95}$	1,084
I-SHEEP	0.53$_{+0.02}$	0.17$_{+0.02}$	0.23$_{+0.01}$	34.26$_{-6.78}$	50.28$_{-7.11}$	5.41$_{-3.45}$	838
Suri	0.26$_{-0.25}$	0.05$_{-0.10}$	0.07$_{-0.15}$	3.19$_{-37.85}$	3.83$_{-53.56}$	0.60$_{-8.26}$	29
Crab	0.56$_{+0.05}$	0.18$_{+0.03}$	0.25$_{+0.03}$	39.92$_{-1.12}$	56.83$_{-0.56}$	9.05$_{+0.19}$	1,192
AIR	**0.61**$_{+0.10}$	**0.24**$_{+0.09}$	**0.31**$_{+0.09}$	**50.69**$_{+9.65}$	**63.89**$_{+6.50}$	**21.00**$_{+12.14}$	1,813
Fine-tuned on Qwen-2.5-7B-UltraChat							
Method	**CF-Bench**			**FollowBench**		**AlpacaEval2**	
	CSR	**ISR**	**PSR**	**HSR**	**SSR**	**LC.**	**Len**
Baseline	0.68	0.29	0.40	47.71	64.79	10.87	836
back-translation	0.42$_{-0.26}$	0.14$_{-0.15}$	0.18$_{-0.22}$	21.62$_{-26.09}$	34.86$_{-29.93}$	1.79$_{-9.08}$	3,266
back-and-forth	0.63$_{-0.05}$	0.24$_{-0.05}$	0.34$_{-0.06}$	45.33$_{-2.38}$	60.39$_{-4.40}$	12.59$_{+1.72}$	1,480
ShareGPT	0.69$_{+0.01}$	0.32$_{+0.03}$	0.41$_{+0.01}$	47.67$_{-0.04}$	64.46$_{-0.33}$	10.75$_{-0.12}$	1,028
Self-Instruct	0.39$_{-0.29}$	0.10$_{-0.19}$	0.14$_{-0.26}$	20.10$_{-27.61}$	35.47$_{-29.32}$	2.47$_{-8.40}$	557
Evol-Instruct	0.67$_{-0.01}$	0.30$_{+0.01}$	0.40$_{+0.00}$	46.67$_{-1.04}$	63.98$_{-0.81}$	8.81$_{-2.06}$	964
MUFFIN	0.61$_{-0.07}$	0.26$_{-0.03}$	0.34$_{-0.06}$	45.27$_{-2.44}$	62.45$_{-2.34}$	8.44$_{-2.43}$	880
Conifer	0.70$_{+0.02}$	0.34$_{+0.05}$	0.44$_{+0.04}$	51.65$_{+3.94}$	65.72$_{+0.93}$	19.39$_{+8.52}$	1,024
I-SHEEP	0.63$_{-0.05}$	0.25$_{-0.04}$	0.36$_{-0.04}$	41.96$_{-5.75}$	59.48$_{-5.31}$	6.43$_{-4.44}$	996
Suri	0.31$_{-0.37}$	0.07$_{-0.22}$	0.10$_{-0.30}$	4.55$_{-43.16}$	4.85$_{-59.94}$	0.94$_{-9.93}$	239
Crab	0.62$_{-0.06}$	0.24$_{-0.05}$	0.32$_{-0.08}$	41.48$_{-6.23}$	59.57$_{-5.22}$	9.68$_{-1.19}$	1,102
AIR	**0.76**$_{+0.08}$	**0.41**$_{+0.12}$	**0.51**$_{+0.11}$	**59.07**$_{+11.36}$	**71.35**$_{+6.56}$	**32.43**$_{+21.56}$	1,779

Table 1: Experiment results on Llama-3-8B and Qwen-2.5-7B, with both models fine-tuned with ultrachat-200k (Ding et al., 2023). Llama-3-70B-Instruct and Qwen-2.5-72B-Instruct are used as the guidance models respectively.

Fine-tuned on Llama-3-8B-Tulu					
Method	**CF-Bench**			**AlpacaEval2**	
	CSR	**ISR**	**PSR**	**LC.**	**Len**
Baseline	0.50	0.15	0.20	5.20	995
back-trans	0.27	0.07	0.08	1.09	2,263
back&forth	0.47	0.14	0.19	9.04	1,337
ShareGPT	0.61	0.21	0.29	9.00	1,080
Self-Instruct	0.30	0.07	0.09	2.63	378
Evol-Instruct	0.58	0.19	0.27	18.09	991
MUFFIN	0.46	0.15	0.18	5.21	760
Conifer	0.61	0.24	0.32	7.15	903
I-SHEEP	0.49	0.16	0.19	3.11	931
Suri	0.25	0.05	0.06	0.44	151
Crab	0.56	0.19	0.27	8.55	1,221
AIR	**0.68**	**0.28**	**0.38**	**22.00**	2,097

Table 2: Experiment results on Llama-3-8B, fine-tuned with tulu-330k (Lambert et al., 2024), with Llama-3-70B-Instruct as the guidance model.

crafted general instruction data significantly underperform, highlighting the importance of multiple constraints in effective instruction fine-tuning. Automatic rewritten instructions also underperform, as their constructed constraints do not align with real-world practice. Additionally, automatically back-translated instructions underperform as well. Despite the constraints being derived from documents, the documents (even after refinement) suffer from misalignment and should not be directly used as the target for fine-tuning.

4.3 Data Quality Evaluation

To evaluate our dataset's quality, we employed the Deita scorer (Liu et al., 2024), which utilizes an LLM to assess the complexity score for instructions and the quality score for both instructions and responses. As shown in Figure 5, our approach significantly outperforms human crafted instructions, automatically crafted general instructions, and automatically rewritten complex instructions in terms of both complexity and quality scores. Notably, our method shows marginal improvements over automatic back-translation approaches like Suri and

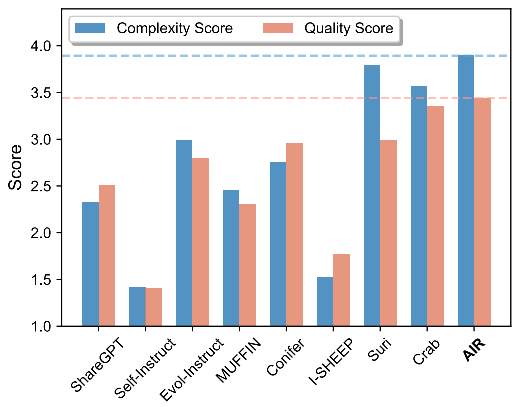
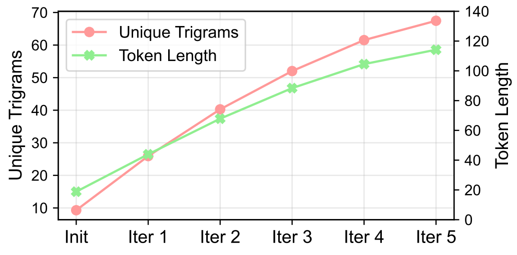
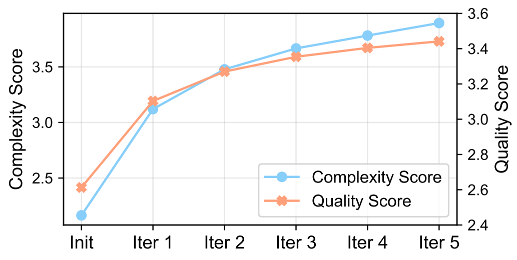


Figure 5: Comparison of averaged complexity and quality scores on different datasets.



(a) Diversity: unique trigrams and token length



(b) Complexity and quality score

Figure 6: Variation of quality indicators across iterations. *Init* represents initial instructions generated by IIG.

Crab, despite their use of high-quality seed datasets (e.g., Alpaca GPT4 for Crab) and advanced models (e.g., GPT-4-turbo for Suri). These results validate the effectiveness of our data generation strategy.

To investigate the effect of iterative refinement, we analyze the variation of average unique trigrams and token lengths across iterations in Figure 6(a). The results demonstrate consistent increases in both instruction length and unique trigrams, indicating that newly added constraints are diverse rather than mere repetition. Furthermore, Figure 6(b) displays the evolution of complexity and quality scores throughout the iterations, showing steady improvement of data quality as the iterations progress.

Method	**FollowBench** HSR	**FollowBench** SSR	**AlpacaEval2** LC.	**AlpacaEval2** Len
Results on Llama-3-8B-UltraChat				
Baseline	41.04	57.39	8.86	1,017
w/o judge	47.15	62.62	19.07	1,706
judge w/o doc	51.24	63.81	20.00	1,717
judge w/ doc	**52.34**	**64.09**	19.74	1,408
w/ check	50.69	63.89	**21.00**	1,813
Results on Llama-3-8B-Tulu				
Baseline	34.91	51.76	5.20	995
w/o judge	47.59	63.60	18.32	2,067
judge w/o doc	50.62	63.69	17.02	2,842
judge w/ doc	**54.16**	**67.52**	20.45	1,639
w/ check	51.35	66.09	**21.09**	2,049

Table 3: Experiment results on Llama-3-8B models with constraints from different judgment strategies.

4.4 Judgment Strategy for Better Constraint

In this section, we investigate the optimal judgment strategy for constraint generation. When humans adjust prompts based on the output, they typically have a pre-expected response as the reference in mind, and constraints are issued to guide the response closer to the reference. Therefore, we compare three judgment settings: 1) No judgment, directly curate constraints; 2) Judge without document as the reference. Instead, use the guidance models' response as the reference; 3) Judge with the refined document as the reference.

As shown in Table 3, the judgment process is essential for uncovering valuable constraints to improve the complex instruction following ability. LLM-judge can curate constraints that reflect the insufficiency of the model which requires further tuning. Moreover, using the document as a reference is also essential due to the limited judgment ability of the model, and human-written references aid in more targeted constraint construction.

On the other hand, the additional checking step does not improve complex instruction-following ability, as the checking step would result in fewer constraints. However, we observe improved performance on general-instruction following, indicating there exists a trade-off between general and complex instruction following abilities.

4.5 Influence of Iterative Judge

In this section, we investigate the effectiveness of our iterative judging approach by examining model performance across different iterations. As shown

Iteration	FollowBench		AlpacaEval2		Token Numbers	
	HSR	SSR	LC.	Len	Input	Output
1	49.75	64.78	21.63	1,994	1,882	1,869
2	53.82	67.55	21.01	1,829	3,351	2,562
3	**54.46**	67.54	20.69	1,722	4,844	3,275
4	53.97	67.09	**22.50**	1,672	6,361	4,008
5	53.30	**67.91**	20.78	1,599	7,902	4,761

Table 4: Experiment results and computational costs for Llama-3-8B-Tulu models across multiple iterations.

Method	FollowBench		AlpacaEval2		Uni-Trigrams
	HSR	SSR	LC.	Len	
1-shot	50.17	63.82	18.49	1,566	41.72
5-iteration	**53.30**	**67.91**	**20.78**	1,599	67.45

Table 5: Comparative analysis between 1-shot and 5-iteration generation for Llama-3-8B-Tulu.

in Table 4, we evaluate models trained on data from different iterations and compute the average number of input and output tokens to quantify the computational cost associated with each iteration.

Specifically, we observe consistent improvements on FollowBench and AlpacaEval2 through the first two iterations. This suggests that the iterative judging process effectively identifies and incorporates increasingly sophisticated constraints that are valuable for complex instruction following. However, improvements tend to plateau after the third iteration. This could be attributed to the fact that the most critical and fundamental constraints have already been discovered in earlier iterations.

Moreover, as shown in Figure 6, data quality improves steadily across iterations. Despite the increased computational costs reflected in Table 4, these iterations generate more valuable constraints that directly enhance model performance.

We also experimented with a 1-shot approach that generates multiple constraints simultaneously. As shown in Table 5, this approach is less effective because it lacks a gradual process of uncovering more challenging constraints with higher diversity (measured by unique trigrams).

4.6 Influence of Sampling Strategy

As explained in Section 3.1, we conducted density-based sampling to ensure the diversity of instruction data. To verify the effectiveness of our approach in enhancing diversity and improving fine-tuning results, we conducted experiments using three different sampling methods for selecting 1K samples:

1. **Random 1K**: Randomly selecting 1K samples.
2. **Density 1K**: Selecting 1K samples using our proposed density-based sampling method.

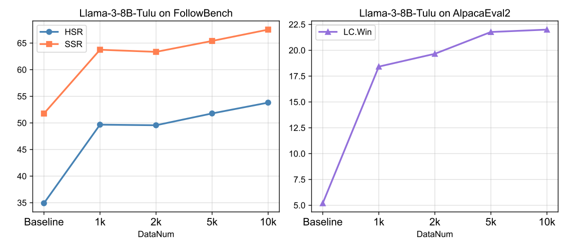


Figure 7: The variation of performance on FollowBench and AlpacaEval2 with the variation of data number.

3. **InsTag (Lu et al., 2023) 1K**: Using GPT-4 to label each instruction with semantic and complexity tags, then selecting instructions to ensure diverse representation.

As shown in Table 6, our density-based method significantly outperforms random selection on both complex and general instruction-following benchmarks. On the other hand, despite InsTag's expensive tagging process, it underperforms compared to our approach. Moreover, we also calculated the average unique trigrams in instructions sampled by different methods, finding that our method could select samples with higher unique trigrams, indicating better diversity.

Method	FollowBench		AlpacaEval2		Unique Trigrams
	HSR	SSR	LC.	Len	
Baseline	41.04	57.39	8.86	1,017	-
Random 1K	41.71	57.66	12.26	1,313	45.13
Density 1K	**45.85**	**60.21**	**17.15**	1,611	66.81
InsTag 1K	44.11	59.89	13.03	1,251	58.09

Table 6: Experiment results on Llama-3-8B-UltraChat fine-tuned on different sampling methods.

4.7 Robustness to Document Quality

As discussed in Section 3.2, our framework primarily utilizes documents to extract constraints rather than as direct fine-tuning targets. During fine-tuning, we use outputs from the guidance model as targets, which insulates the process from document quality issues. Moreover, we implemented multiple safeguards in our data production pipeline, which provides inherent robustness against document noise.

To empirically validate our method's robustness to document quality, we conducted an additional experiment comparing two document sets: (1) a "low-quality" subset comprising the 1,000 lowest-scoring documents from AIR-10k (evaluated by GPT-4 across helpfulness, completeness, and harmlessness dimensions), and (2) a randomly sampled set of 1,000 documents from the same corpus.

Table 7 presents the performance comparison of Llama-3-8B-UltraChat models fine-tuned on instructions derived from these two document sets. The results demonstrate negligible performance differences across all evaluation benchmarks. This empirical evidence confirms that our method maintains consistent performance regardless of input document quality, validating its robustness.

Additionally, we provide the number of samples at each stage of the iterative process. As shown in Table 8, despite starting with a large number of initial documents, only 15% of them are retained for iterative instruction refinement. Furthermore, nearly 50% of the documents are filtered out during the iterations, leaving only the highest-quality samples for final training. This demonstrates how our carefully designed strategies effectively maintain sample quality across multiple iterations.

Data	CFBench			FollowBench		AlpacaEval2	
	CSR	ISR	PSR	HSR	SSR	LC.	Len
random 1k	0.61	0.23	0.33	45.85	60.21	17.15	1,611
low-quality 1k	0.60	0.22	0.30	44.00	59.83	16.10	1,685

Table 7: Comparison of model performance when fine-tuned on instructions from different document sets.

Stage	Sample Number
Rule-based Filtering	300k
Density-based Sampling	60k
Instruction Scoring	20k
Iteration 1	17.3k
Iteration 2	15.2k
Iteration 3	13.7k
Iteration 4	12.7k
Iteration 5	11.9k

Table 8: Progressive reduction in sample size through filtering stages and five iterations.

4.8 Influence of Data Quantity

In this section, we investigate the impact of data quantity on AIR's performance. We present the results of models trained with varying amounts of data in Figure 7. As shown, performance on both general and complex instruction tasks improves with increasing data quantity. On the other hand, the model can achieve superior performance with only 1k training samples, and the performance gains become marginal as more data is added. Therefore, in practical applications, the optimal amount of fine-tuning data can be determined based on available computational resources.

Guid. Model	FollowBench		AlpacaEval2	
	HSR	SSR	LC.	Len
Baseline	47.71	64.79	10.87	836
14B	57.72	70.59	29.13	1,501
32B	**60.06**	**71.97**	26.39	1,309
72B	59.07	71.35	**32.43**	1,779

Table 9: Experiment results on Qwen-2.5-7B-UltraChat fine-tuned with different guidance model size.

4.9 Influence of Guidance Model Size

In Table 9, we investigate the impact of guidance model size on AIR's performance. We performed experiments with Qwen-2.5-7B-UltraChat as the base model, while varying the guidance model size from 14B to 72B parameters. As shown, all guidance models with different sizes significantly improve instruction-following ability compared to the baseline, while larger models generally provide greater improvement. On the other hand, even the 14B guidance model demonstrates remarkable improvement. This scalability across different model sizes highlights the robustness and efficiency of our proposed approach.

5 Conclusion

This paper introduces the Automatic Iterative Refinement (AIR) framework, a novel approach for generating complex instructions that better align with real-world scenarios. We also construct a complex instruction dataset, AIR-10K, to facilitate the application of complex instruction following.

While previous methods for complex instruction following often introduce constraints without clear justification, it is crucial to understand what authentic complex instruction entails. In the future, we will conduct further research on the effectiveness and efficiency of complex instruction data.

Limitations

Our work has several limitations. 1) Although our evaluation includes multiple established benchmarks and metrics, including human evaluation could further improve its credibility. Due to time and resource limitations, we have to leave this as future work. 2) Despite meticulous preprocessing, the Dolma dataset remains relatively noisy. Incorporating more high-quality documents (for example, judicial documents made public) could provide more knowledge and formality to support constraint construction. 3) The iterative nature of our framework requires multiple rounds of model inference, resulting in higher computational demands. While our ablation studies demonstrate effectiveness even with smaller guidance models and fewer samples, the computational cost remains a challenge for researchers with limited resources.

Ethical Considerations

Our data construction framework primarily leverages proprietary models such as Llama-3-70B-Instruct, which have undergone extensive preference optimization to minimize the likelihood of generating instructions that raise ethical concerns. However, large-scale web corpora—our primary data sources—are uncensored and may contain harmful or toxic content. To address this, we recommend implementing more rigorous and meticulous filtering mechanisms to proactively identify and remove such instances if possible.

While the AIR framework mainly aims to enhance models' ability to follow complex instructions, it is important to note that some user constraints may conflict with system constraints set by developers. For example, users may request the generation of harmful or toxic content. Although our study does not specifically investigate conflicting constraints, there is a potential risk that the pipeline could prioritize user requests over developer-defined safeguards.

Acknowledgement

This work was supported in part by the Jiangsu Science and Technology Major Project (BG2024031) and Nanjing University AI & AI for Science Funding (2024300540).

References

Zhangqian Bi, Yao Wan, Zheng Wang, Hongyu Zhang, Batu Guan, Fangxin Lu, Zili Zhang, Yulei Sui, Hai Jin, and Xuanhua Shi. 2024. Iterative refinement of project-level code context for precise code generation with compiler feedback. *arXiv preprint arXiv:2403.16792*.

Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. 2023. Instruction mining: Instruction data selection for tuning large language models. *arXiv preprint arXiv:2307.06290*.

Guiming Hardy Chen, Shunian Chen, Ziche Liu, Feng Jiang, and Benyou Wang. 2024. Humans or llms as the judge? a study on judgement biases. *arXiv preprint arXiv:2402.10669*.

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.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Joseph L Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378.

Hongyi Guo, Yuanshun Yao, Wei Shen, Jiaheng Wei, Xiaoying Zhang, Zhaoran Wang, and Yang Liu. 2024. Human-instruction-free llm self-alignment with limited samples. *arXiv preprint arXiv:2401.06785*.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024a. Can large language models understand real-world complex instructions? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18188–18196.

Yancheng He, Shilong Li, Jiaheng Liu, Yingshui Tan, Weixun Wang, Hui Huang, Xingyuan Bu, Hangyu Guo, Chengwei Hu, Boren Zheng, Zhuoran Lin, Xuepeng Liu, Dekai Sun, Shirong Lin, Zhicheng Zheng, Xiaoyong Zhu, Wenbo Su, and Bo Zheng. 2024b. Chinese simpleqa: A chinese factuality evaluation for large language models. *Preprint*, arXiv:2411.07140.

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

Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *2nd Workshop on Neural Machine Translation and Generation*, pages 18–24. Association for Computational Linguistics.

Hui Huang, Jiaheng Liu, Yancheng He, Shilong Li, Bing Xu, Conghui Zhu, Muyun Yang, and Tiejun Zhao. 2025. Musc: Improving complex instruction following with multi-granularity self-contrastive training. *arXiv preprint arXiv:2502.11541*.

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*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. 2024. T\" ulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*.

Shilong Li, Yancheng He, Hangyu Guo, Xingyuan Bu, Ge Bai, Jie Liu, Jiaheng Liu, Xingwei Qu, Yangguang Li, Wanli Ouyang, et al. 2024a. Graphreader: Building graph-based agent to enhance long-context abilities of large language models. *arXiv preprint arXiv:2406.14550*.

Shilong Li, Yancheng He, Hui Huang, Xingyuan Bu, Jiaheng Liu, Hangyu Guo, Weixun Wang, Jihao Gu, Wenbo Su, and Bo Zheng. 2024b. 2d-dpo: Scaling direct preference optimization with 2-dimensional supervision. *arXiv preprint arXiv*.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. 2023. Self-alignment with instruction back-translation. *arXiv preprint arXiv:2308.06259*.

Yiming Liang, Ge Zhang, Xingwei Qu, Tianyu Zheng, Jiawei Guo, Xinrun Du, Zhenzhu Yang, Jiaheng Liu, Chenghua Lin, Lei Ma, Wenhao Huang, and Jiajun Zhang. 2024. I-sheep: Self-alignment of llm from scratch through an iterative self-enhancement paradigm. *CoRR*, abs/2408.08072.

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

Renze Lou, Kai Zhang, Jian Xie, Yuxuan Sun, Janice Ahn, Hanzi Xu, Yu Su, and Wenpeng Yin. 2023. Muffin: Curating multi-faceted instructions for improving instruction following. In *The Twelfth International Conference on Learning Representations*.

K. Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, and Chang Zhou. 2023. #instag: Instruction tagging for analyzing supervised fine-tuning of large language models. *ArXiv*, abs/2308.07074.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Thao Nguyen, Jeffrey Li, Sewoong Oh, Ludwig Schmidt, Jason Weston, Luke Zettlemoyer, and Xian Li. 2024. Better alignment with instruction back-and-forth translation. *arXiv preprint arXiv:2408.04614*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following for long-form text generation. *arXiv preprint arXiv:2406.19371*.

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2024. Constraint back-translation improves complex instruction following of large language models. *arXiv preprint arXiv:2410.24175*.

Rico Sennrich. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709*.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Evan Walsh, Luke Zettlemoyer, Noah Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. 2024. Dolma: an open corpus of three trillion tokens for language model pretraining research. In *Proceedings of the 62nd Annual Meeting* *of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15725–15788, Bangkok, Thailand. Association for Computational Linguistics.

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

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Chandu, David Wadden, Kelsey MacMillan, Noah A Smith, Iz Beltagy, et al. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Advances in Neural Information Processing Systems*, 36:74764–74786.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language models with self-generated instructions. *arXiv preprint arXiv:2212.10560*.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, et al. 2024. Benchmarking complex instruction-following with multiple constraints composition. *arXiv preprint arXiv:2407.03978*.

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*.

Qiancheng Xu, Yongqi Li, Heming Xia, and Wenjie Li. 2024. Enhancing tool retrieval with iterative feedback from large language models. *arXiv preprint arXiv:2406.17465*.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, et al. 2024. Justice or prejudice? quantifying biases in llm-as-a-judge. *arXiv preprint arXiv:2410.02736*.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. 2024. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Tianyu Zheng, Shuyue Guo, Xingwei Qu, Jiawei Guo, Xinrun Du, Qi Jia, Chenghua Lin, Wenhao Huang, Jie Fu, and Ge Zhang. 2024a. Kun: Answer polishment for chinese self-alignment with instruction back-translation. *arXiv preprint arXiv:2401.06477*.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024b. 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)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2024a. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024b. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE.

A Additional Experimental Results and Analysis

A.1 Analysis of LLM-as-Judge Bias

Previous studies such as Chen et al. (2024); Ye et al. (2024) have revealed that biases in LLM-as-Judge approaches can significantly influence judgment outcomes. Therefore, this section conducts an analysis of the potential impact of LLM-as-Judge bias during the constraint generation process.

For this purpose, we analyzed the top three constraints with the highest proportions identified during each iteration, along with their respective distribution percentages. As shown in the Table 10, as the iteration progresses, the variety of constraint types becomes increasingly diverse, including a wide range of constraint types such as content constraints, tone constraints, and emotional constraints, indicating that the LLM-as-Judge's bias did not lead to an overabundance of specific format-related constraints.

Iterations	Primary Constraint	Secondary Constraint	Tertiary Constraint
1	Inclusion (35%)	Document Structure (23%)	Tone and Style (12%)
2	Inclusion (27%)	Document Structure (23%)	Target Audience (15%)
3	Document Structure (20%)	Target Audience (19%)	Inclusion (18%)
4	Target Audience (18%)	Tone and Style (16%)	Document Structure (15%)
5	Target Audience (15%)	Tone and Style (15%)	Domain - Specific Format (14%)

Table 10: Distributions of top three constraints across iterations in the iterative constraint construction process.

While LLM-as-Judge approaches may exhibit certain biases in response selection or scoring tasks (e.g., favoring longer or more formatted answers regardless of instruction adherence), our implementation mitigates these concerns, as we mainly rely on LLM-as-Judge specifically to identify missing constraints from current responses with reference documents as ground truth, rather than for response selection or scoring. This targeted application substantially reduces the impact of potential biases.

Furthermore, as demonstrated in Section 4.9, experiments with smaller models as LLM-as-Judge resulted in only minimal performance degradation. This finding underscores the robustness of our methodology: even with less capable judge models, we can still construct effective constraints that enhance complex instruction-following capabilities.

A.2 Impact on Fundamental Capabilities

Method	MMLU	CQA	NQ	HumanEval	GSM8K	AVG
Results on Llama-3-8B-UltraChat						
Baseline	64.00	72.97	29.61	30.49	57.47	50.90
AIR	61.64	**73.63**	**30.54**	29.88	54.59	50.05
Results on Qwen-2.5-7B-UltraChat						
Baseline	73.64	82.39	25.68	52.20	81.65	63.11
AIR	73.35	**82.56**	**25.76**	**55.49**	**84.38**	**64.30**
Results on Llama-3-8B-Tulu						
Baseline	65.43	79.44	32.22	50.61	64.14	58.36
AIR	64.95	**79.92**	**34.62**	**50.85**	63.70	**58.80**

Table 11: Experiment results on fundamental capabilities.

Previous methods have shown LLMs may suffer from capability degradation during alignment (Ouyang et al., 2022). To evaluate this concern, we tested our AIR method on MMLU (Hendrycks et al., 2021), CommonsenseQA (CQA) (Talmor et al., 2019), Natural Questions (NQ) (Kwiatkowski et al., 2019), HumanEval (Chen et al., 2021), and GSM8K (Cobbe et al., 2021). In Table 11, our method does not have a negative impact on fundamental capabilities. For Qwen-2.5-7B-UltraChat and Llama-3-8B-Tulu, our method even improves the average performance by 1.19 and 0.44 points, respectively. This indicates that instructions constructed from documents with evenly sampled distributions also exhibit even distribution, which would not lead to catastrophic forgetting of fundamental capabilities.

A.3 Comparison with Related Iterative Refinement Paradigms

Method	CFBench			FollowBench		AlpacaEval2	
	CSR	ISR	PSR	HSR	SSR	LC.	Len
Results on Llama-3-8B-UltraChat							
Self-Refine	0.58	0.22	0.30	46.82	61.62	18.91	1,706
AIR	**0.61**	**0.24**	**0.31**	**50.69**	**63.89**	**21.00**	1,813
Results on Qwen-2.5-7B-UltraChat							
Self-Refine	0.71	0.38	0.48	54.99	67.33	28.27	2,093
AIR	**0.76**	**0.41**	**0.51**	**59.07**	**71.35**	**32.43**	1,779

Table 12: Performance comparison between AIR and Self-Refine methods.

This section presents a comprehensive comparison between our proposed AIR method and existing iterative refinement paradigms. While prior work has explored iterative refinement in various contexts, our approach differs fundamentally in both objective and methodology. Existing methods typically focus on improving response quality through multiple iterations in specific domains such as code generation (Madaan et al., 2023; Bi et al., 2024), tool retrieval (Xu et al., 2024), and task planning (Zhou et al., 2024b), where the refined output serves as the final result. In contrast, our work targets instruction construction, where the refined instructions are utilized to enhance the model's capability in following complex instructions. Additionally, these tasks often include external feedback (e.g., code executor), while we primarily rely on LLM-as-Judge as feedback during iteration.

To validate the effectiveness of our approach, we conducted external experiments comparing AIR with the Self-Refine baseline across multiple benchmarks, as shown in Table 12. The results demonstrate that AIR consistently outperforms Self-Refine across all evaluation metrics on both models, indicating the superiority of our proposed iterative refinement strategy.

A.4 Comprehensive Human Preference Study

To complement our automated evaluation metrics, we conducted a blind, pairwise human evaluation to compare instructions generated by our AIR framework against those from strong and representative baselines.

For this study, we selected three diverse and highly relevant baselines for comparison:

1. **ShareGPT**: A widely-used dataset composed of real, human-crafted conversations.
2. **Conifer**: A state-of-the-art method for generating complex instructions via automatic rewriting.
3. **Crab**: A strong baseline that, similar to our work, utilizes back-translation from existing documents.

The evaluation was conducted as a blind pairwise comparison. For each baseline, we randomly sampled 100 instructions. To ensure a fair and controlled comparison, each instruction pair (e.g., one from AIR, one from Crab) was generated from the identical source document. We recruited three expert annotators, all holding Master's degrees and proficient in NLP, who were blind to the origin of the instructions. Annotators were asked to select a winner for each pair based on three criteria: *Instruction Complexity*, *Instruction Clarity*, and *Overall Preference*.

The final results, determined by majority vote, are presented in Table 13. The findings reveal an overwhelming preference for instructions generated by our AIR framework across all comparisons and criteria.

To ensure the reliability of our human evaluation, we also calculated the inter-annotator agreement using Fleiss' Kappa (Fleiss, 1971). As shown in Table 14, the high agreement scores confirm the consistency and validity of our annotators' judgments.

Comparison	Evaluation Criterion	AIR Wins (%)	Baseline Wins (%)	Tie (%)
AIR vs. ShareGPT	Instruction Complexity	66.0	30.0	4.0
	Instruction Clarity	58.0	33.0	9.0
	Overall Preference	60.0	29.0	11.0
AIR vs. Conifer	Instruction Complexity	69.0	21.0	10.0
	Instruction Clarity	67.0	23.0	10.0
	Overall Preference	68.0	23.0	9.0
AIR vs. Crab	Instruction Complexity	70.0	23.0	7.0
	Instruction Clarity	67.0	27.0	6.0
	Overall Preference	67.0	21.0	12.0

Table 13: Pairwise human evaluation results comparing AIR against three baseline methods across three criteria. Results are based on the majority vote of three expert annotators.

Comparison	Fleiss' Kappa (%)
AIR vs. ShareGPT	82.82
AIR vs. Conifer	83.34
AIR vs. Crab	80.47

Table 14: Inter-annotator agreement (Fleiss' Kappa) for the human preference study.

This study provides compelling human-centric evidence that AIR generates instructions that are not only more complex but also significantly clearer and more preferred than those from both human-crafted datasets and other state-of-the-art automated methods.

A.5 Controlled Comparison of Instruction Generation Methods

The main paper evaluates AIR-10K against existing public datasets, a common paradigm that reflects how these resources are used in practice. However, this "dataset-vs-dataset" approach can be influenced by confounding variables, such as the quality of the source corpus and the capability of the guidance models used to generate each dataset.

To isolate the direct impact of the instruction generation *methodology* itself, we conducted an additional set of experiments under a strictly controlled environment. In this setting, we re-generated datasets for all baseline methods using the identical source corpus (**Dolma v1.7**) and guidance model (**Llama-3-70B-Instruct**) employed in our AIR framework. This ensures a direct "apples-to-apples" comparison of the underlying methodologies, removing the influence of external factors.

The results of this controlled comparison are presented in Table 15. Our analysis reveals several key insights:

1. **Robust Superiority of AIR**: Under these strictly controlled conditions, our AIR method continues to significantly outperform all baseline methodologies across all complex instruction-following (CFBench, FollowBench) and general-purpose (AlpacaEval2) benchmarks. This provides strong evidence that the performance gains are intrinsic to our Automatic Iterative Refinement framework.
2. **Sensitivity to Foundational Components**: The experiment underscores the high sensitivity of instruction generation methods to the quality of their source data and guidance model. For instance, methods like Suri and Self-Instruct exhibited substantial performance improvements when provided with our high-quality setup. Conversely, methods such as Conifer and Crab, which may have originally benefited from highly curated private data or more powerful proprietary models, saw a performance decrease. Meanwhile, robust methods like Evol-Instruct and ShareGPT performed comparably, suggesting their original configurations were already strong.

In summary, this controlled analysis provides stronger evidence for the effectiveness of the AIR framework, demonstrating that its advantages are attributable to the methodology itself rather than the initial choice of data or models.

Method	Setting	CFBench			FollowBench		AlpacaEval2	
		CSR	ISR	PSR	HSR	SSR	LC.	Len
Baseline	-	0.51	0.15	0.22	41.04	57.39	8.86	1,017
back-translation	Original	0.40	0.11	0.15	21.19	33.92	0.96	2,966
back-and-forth	Original	0.58	0.20	0.27	44.65	61.58	10.06	1,440
ShareGPT	Original	0.62	0.22	0.32	40.99	58.59	8.36	1,052
	Re-run	0.59	0.20	0.31	44.50	59.20	12.70	1,570
Self-Instruct	Original	0.34	0.08	0.10	12.33	26.92	2.76	384
	Re-run	0.42	0.11	0.16	15.00	32.00	8.50	2,384
Evol-Instruct	Original	0.57	0.22	0.28	43.58	59.21	7.15	903
	Re-run	0.55	0.21	0.26	45.20	59.80	8.80	890
MUFFIN	Original	0.50	0.16	0.22	30.88	48.48	4.51	791
	Re-run	0.48	0.15	0.20	29.50	46.20	4.01	990
Conifer	Original	0.57	0.22	0.28	47.06	61.32	12.81	1,084
	Re-run	0.52	0.18	0.23	43.20	58.50	11.50	850
I-SHEEP	Original	0.53	0.17	0.23	34.26	50.28	5.41	838
	Re-run	0.55	0.18	0.25	36.00	52.50	7.20	1,070
Suri	Original	0.26	0.05	0.07	3.19	3.83	0.60	29
	Re-run	0.53	0.17	0.23	43.00	58.08	9.50	1,980
Crab	Original	0.56	0.18	0.25	39.92	56.83	9.05	1,192
	Re-run	0.54	0.17	0.23	38.50	55.50	8.80	1,105
AIR	Original	**0.61**	**0.24**	**0.31**	**50.69**	**63.89**	**21.00**	1,813

Table 15: Performance comparison on Llama-3-8B-UltraChat under controlled experimental settings. The "Re-run" setting indicates that a method's dataset was regenerated using our standardized corpus (Dolma v1.7) and guidance model (Llama-3-70B-Instruct).

A.6 Case Study for Complete Pipeline

This section presents a detailed end-to-end demonstration of our pipeline in Figure 9. The case study provides a thorough walkthrough of each stage in our instruction generation and refinement process.

A.7 Constraint Type Taxonomy and Distribution Analysis

This section provides a detailed classification of constraint types, as defined in Table 16. Additionally, we present a comprehensive analysis of constraint type distribution patterns observed across five iterative refinement rounds, as visualized in Figure 8.

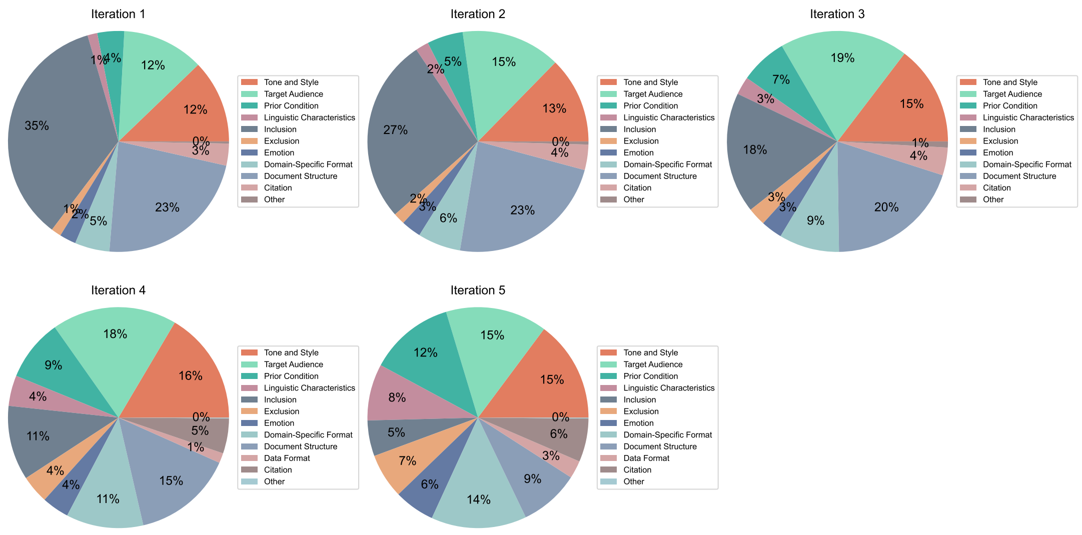
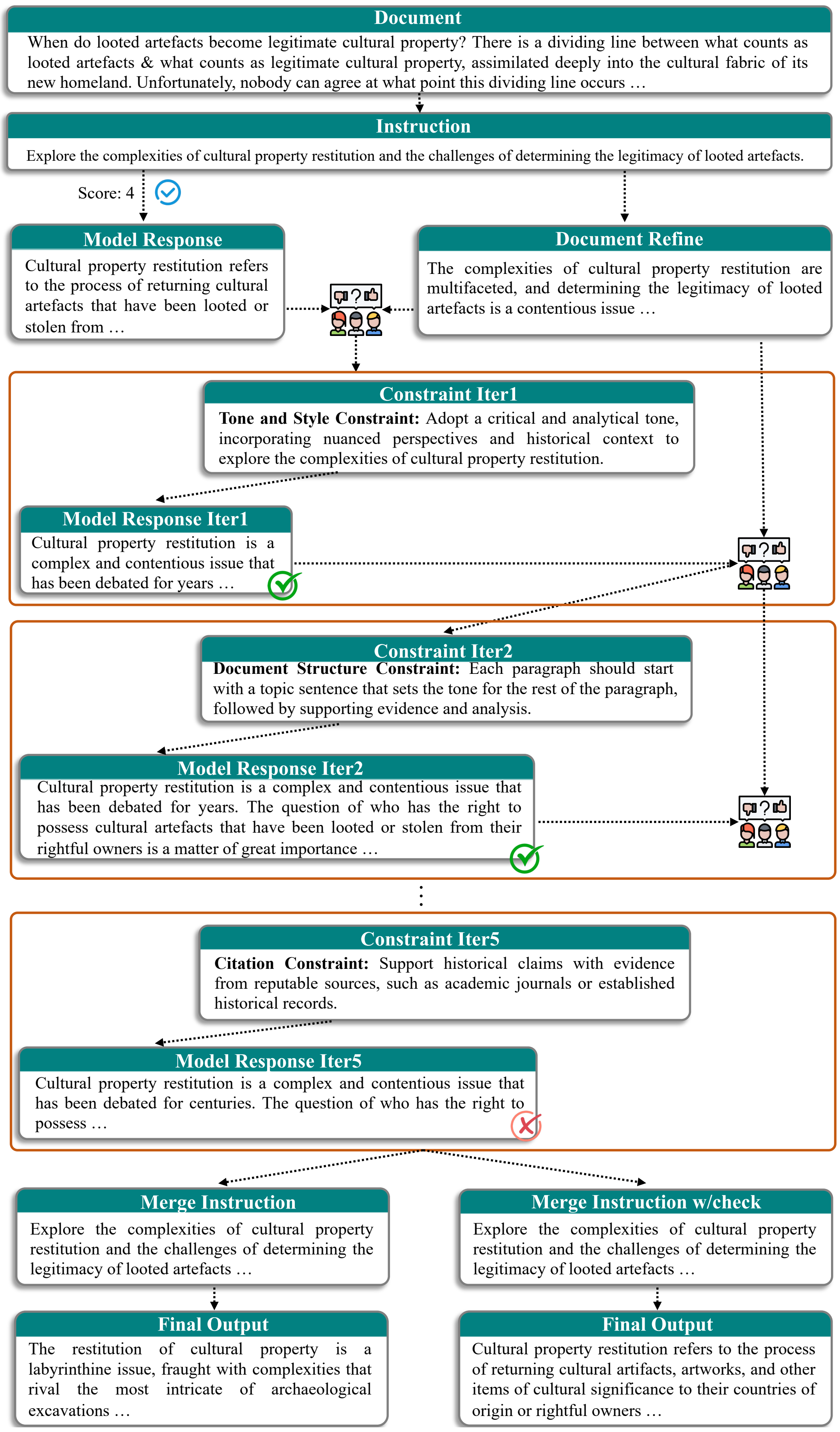


Figure 8: Distribution of constraint types across all iterations.

Constraint Type	Description
Data Format	The generated content should conform to specific data structure formats, such as JSON, Markdown, Table, CSV, etc.
Document Structure	The generated content should follow specific document organization patterns, including Numbered lists (1, 2, 3 or I, II, III), Bullet points (•, -, *), Custom templates with predefined sections, Tables, Headers, etc.
Domain-Specific Format	Content must follow strict format rules for different industries
Inclusion	Identify and list the specific elements or information that should be included in the generated content
Exclusion	Identify and list the specific elements or information that should not be included in the generated content
Citation	The generated content should include citations to sources, providing reliable sources and literature support; follow specific citation formats or reference styles
Prior Condition	When a specific intention is met, a particular process should be followed to perform an operation or output specific content
Target Audience	The generated content should target a specific audience, which affects the terminology used, the level of detail provided, and the complexity of the content
Tone and Style	The generated content should adopt a specific tone and style, such as formal, polite, academic, concise, literary, romantic, or sci-fi
Emotion	The generated content should express a specific emotion or mood, such as ensuring the content is positive, inspiring, or empathetic
Linguistic Characteristics	Use specific linguistic features, such as metaphors, personification, and other rhetorical devices
Multilingual	The generated content should be written in a specific language, such as English, Mandarin, or Spanish

Table 16: Types of Constraints Used in Dataset Generation.



Document
When do looted artefacts become legitimate cultural property? There is a dividing line between what counts as looted artefacts & what counts as legitimate cultural property, assimilated deeply into the cultural fabric of its new homeland. Unfortunately, nobody can agree at what point this dividing line occurs …
Instruction
Explore the complexities of cultural property restitution and the challenges of determining the legitimacy of looted artefacts.
Score: 4
Model Response
Cultural property restitution refers to the process of returning cultural artefacts that have been looted or stolen from …
Document Refine
The complexities of cultural property restitution are multifaceted, and determining the legitimacy of looted artefacts is a contentious issue …
Constraint Iter1
Tone and Style Constraint: Adopt a critical and analytical tone, incorporating nuanced perspectives and historical context to explore the complexities of cultural property restitution.
Model Response Iter1
Cultural property restitution is a complex and contentious issue that has been debated for years …
Constraint Iter2
Document Structure Constraint: Each paragraph should start with a topic sentence that sets the tone for the rest of the paragraph, followed by supporting evidence and analysis.
Model Response Iter2
Cultural property restitution is a complex and contentious issue that has been debated for years. The question of who has the right to possess cultural artefacts that have been looted or stolen from their rightful owners is a matter of great importance …
Constraint Iter5
Citation Constraint: Support historical claims with evidence from reputable sources, such as academic journals or established historical records.
Model Response Iter5
Cultural property restitution is a complex and contentious issue that has been debated for centuries. The question of who has the right to possess …
Merge Instruction
Explore the complexities of cultural property restitution and the challenges of determining the legitimacy of looted artefacts …
Merge Instruction w/check
Explore the complexities of cultural property restitution and the challenges of determining the legitimacy of looted artefacts …
Final Output
The restitution of cultural property is a labyrinthine issue, fraught with complexities that rival the most intricate of archaeological excavations …
Final Output
Cultural property restitution refers to the process of returning cultural artifacts, artworks, and other items of cultural significance to their countries of origin or rightful owners …

Figure 9: End-to-End Pipeline Implementation Example.

B Implementation Details

B.1 Model Training Hyper-parameters

This section details our model training configuration based on the LlamaFactory (Zheng et al., 2024b) framework. We employed Supervised Fine-Tuning (SFT) with hype-rparameters as outlined in Table 17.

Configuration	**Llama-3-8B**	**Qwen-2.5-7B**
max length	4096	4096
learning rate	1e-5	1e-5
scheduler	cosine decay	cosine decay
training epochs	3	3
batch size	64	64
flash-attn	fa2	fa2
numerical precision	bf16	bf16
ZeRO optimizer	stage 2	stage 2

Table 17: Hyper-parameters for Supervised Fine-Tuning.

B.2 Prompt Templates

This section presents the prompts used in our data generation pipeline. For Initial Instruction Generation, the prompts serve different purposes from initial instruction generation through back-translation (Figure 10) to document refining (Figure 11) and instruction scoring (Figure 13). For Iterative Instruction Refinement, the prompts serve different purposes from constraint generation (Figure 12), constraint verification (Figure 14), and finally combines all elements into refined instructions (Figure 15).

B.3 Instruction Score Criteria

This section presents the detailed score criteria of the instruction quality through representative examples. As illustrated in Figure 16, we provide a diverse set of instructions spanning the entire quality spectrum (scores 1-5). Each score category is exemplified by five carefully selected cases, where score 1 represents basic quality and score 5 demonstrates exceptional quality.

```
Please generate a single instruction that would lead to the given text as a response.
- The instruction should not be a question. Instead, it should be a more general task.
- The instruction should not cover all details of the response. Instead, it should be concise and
  only focus on the main aspect.

Please generate your instruction based on the text.
Text: {document}
Instruction:
```

Figure 10: Prompt for generating initial instructions through back-translation.

```
You are a professional editor. Given an instruction and an original response, your task is to improve the response while ensuring it aligns well with the instruction.

The improvement should focus on:
-  Better alignment with the instruction
-  Enhanced clarity and coherence
-  Aligns with AI assistant response style
-  Maintaining the core message while improving expression.

Now, this is your task. Please directly present your modifications, without using ANY headings or prefixes.

Instruction: {instruction}
Original Response: {document}
Enhanced Response:
```

Figure 11: Prompt for refining document content.

```
Based on the provided instruction, I obtained Output1 and Output2 from two different models. Please analyze both outputs carefully to identify the MOST CRITICAL constraint type that Output2 needs to improve to match Output1's quality.

Available Constraint Types:
{constraints_type}

Task Requirements:
1. [Analysis] Compare Output1 and Output2 to identify differences
2. [Selection] Choose the SINGLE most critical constraint type where Output2 shows the biggest gap
3. [Constraint] Create ONE specific constraint that:
- Addresses ONLY the selected constraint type
- Exists in Output1 but is missing in Output2
- Is written in a clear and concise sentence (10-20 words)
- Avoids references to "Output1" or "Output2"
4. If no significant differences match the available types, specify "None"

Required Response Format:
**Analysis**: [Brief analysis]
**Selected Type**: [Single most critical type]
**Constraint**: [ONE specific constraint]

Context:
#Instruction#
{instruction}

#Output1#
{document_refine}

#Output2#
{model_response}

#Your Response#
```

Figure 12: Prompt for generating constraints based on judge.

Review the user's instruction using the additive 5-point scoring system described below. Points are accumulated based on the satisfaction of each criterion:

Award 1 point for containing a basic question or task.
Add 1 point if the instruction can be addressed using the language model's existing knowledge base without requiring external resources or current event information.
Add 1 point if the instruction does NOT require analyzing specific texts, documents, or specific person's perspective.
Add 1 point if the instruction effectively communicates both the core question and key preferences, demonstrating clear intent while being self-contained.
Add 1 point if the instruction pertains to general topics or advice that are widely applicable and within the common knowledge base, rather than requiring specialized or niche information about specific individuals or events.

After examining the instruction:
- Briefly justify your total score, up to 100 words.
- Conclude with the score using the format: "Score: <total points>/5"

Example 1:
Instruction: What was the impact of Gary Gilmour's career and his life in the years following his cricketing career?
Answer: The instruction poses a basic question about Gary Gilmour's impact after his cricketing career (1 point). It can be answered using the language model's existing knowledge (1 point). It doesn't require analyzing specific texts, documents, or a specific person's perspective (1 point). The question is clear, self-contained, and demonstrates clear intent (1 point). However, since it involves information about a specific individual, which requires specialized or niche knowledge, the last point is not awarded.
Score: 4/5

Example 2:
Instruction: What's the most helpful advice you have for students who are awaiting their college admission decision?
Answer: The instruction asks for the most helpful advice for students awaiting their college admission decisions, which is a basic question (1 point). It can be answered using the language model's existing knowledge (1 point). It does not require analyzing specific texts, documents, or a specific person's perspective (1 point). The question is clear, self-contained, and demonstrates clear intent (1 point). It pertains to a general topic that is widely applicable and within the common knowledge base (1 point).
Score: 5/5

…

This is your task:
Instruction: {instruction}
Answer:

Figure 13: Prompt for scoring initial instructions.

I want you to act as a quality evaluator. You need to evaluate the model answer by combining [User Instructions], [Model Answer], and [Evaluation Criteria] and score with 0-3.

Specifically, [Model Answer] is the response to [User Instructions], and [Evaluation Criteria] defines the points that the model answer should satisfy and needs to be evaluated. You need to strictly score the [Model Answer] according to each evaluation point in [Evaluation Criteria].

Scoring Rules:
- Score 0: Does not meet the evaluation criteria
- Score 1: Meets the evaluation criteria with acceptable response
- Score 2: Meets the evaluation criteria with high quality and comprehensive response
- Score 3: Meets the evaluation criteria with exceptional and flawless response

Output format: 1. Strictly output one line at a time according to the order of evaluation points in [Evaluation Criteria], with lines separated by "\n\n";
2. Each line first outputs the corresponding content in [Evaluation Criteria], then uses "\t" to separate, and outputs the corresponding score (0-3) after it;
3. Please output your evaluation directly without any other content;
4. Note that if a criteria states like "do not include X", the score should be 0 if the answer includes X.

[User Instructions]: {instruction}

[Model Answer]: {model_response}

[Evaluation Criteria]: {constraints}

[Your Evaluation]:

Figure 14: Prompt for verifying model responses against constraints.

You are a skilled writing specialist who excels at blending different elements into cohesive, natural-sounding instructions.

Fusion guidelines:
- Consolidates overlapping constraints and resolves any conflicts
- Craft a cohesive instruction that naturally integrates ALL appropriate constraints
- AVOID expanding constraints

{few_shot}

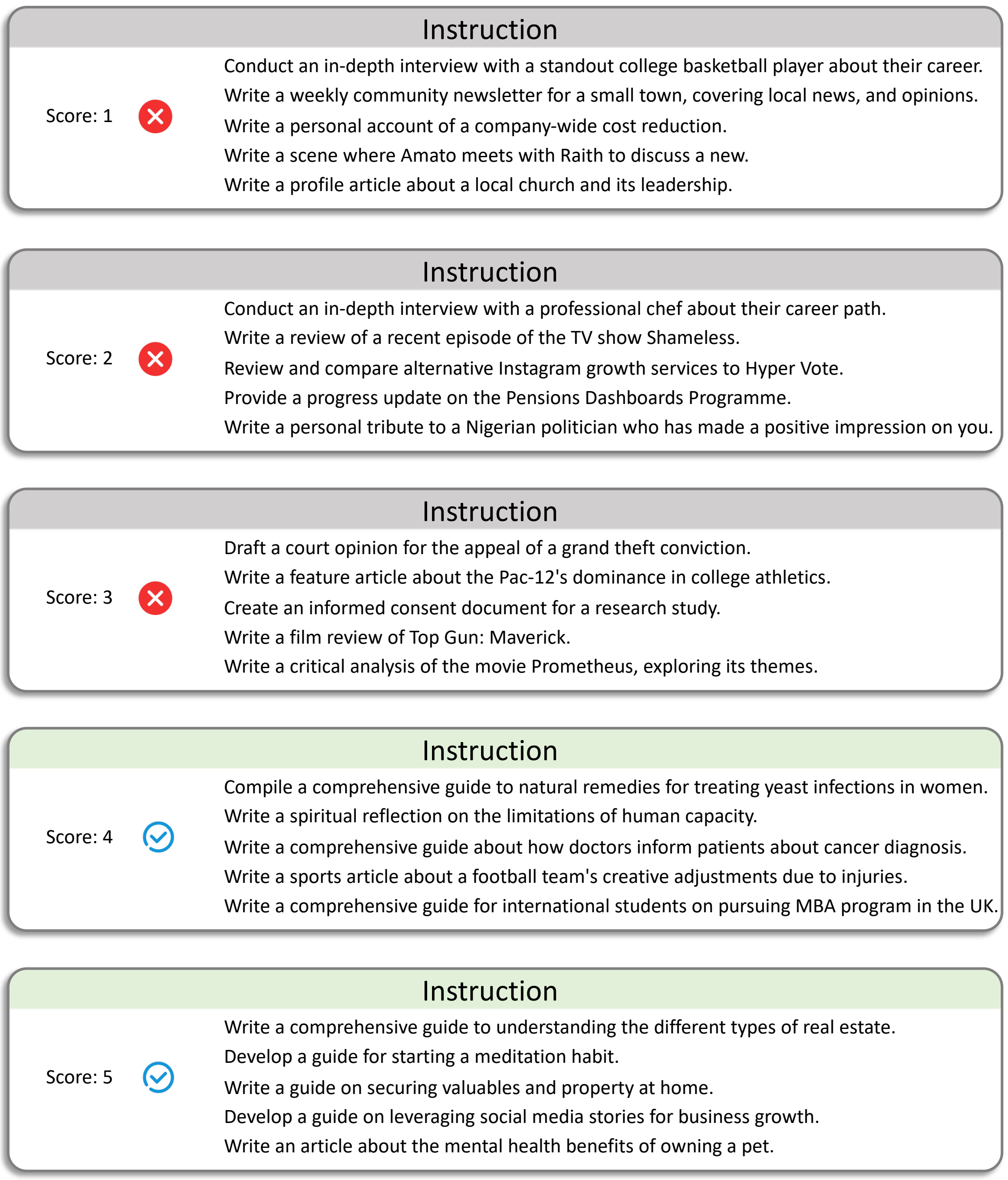
Now it's your turn. Please merge the following input and constraints, do not output anything else, including response to the merged instruction:

[Original Input]
{instruction}

[Original Constraints]
{constraints}

[Merged Instruction]

Figure 15: Prompt for combining instructions with constraints.



Instruction

Score: 1 ✗

Conduct an in-depth interview with a standout college basketball player about their career.
Write a weekly community newsletter for a small town, covering local news, and opinions.
Write a personal account of a company-wide cost reduction.
Write a scene where Amato meets with Raith to discuss a new.
Write a profile article about a local church and its leadership.

Instruction

Score: 2 ✗

Conduct an in-depth interview with a professional chef about their career path.
Write a review of a recent episode of the TV show Shameless.
Review and compare alternative Instagram growth services to Hyper Vote.
Provide a progress update on the Pensions Dashboards Programme.
Write a personal tribute to a Nigerian politician who has made a positive impression on you.

Instruction

Score: 3 ✗

Draft a court opinion for the appeal of a grand theft conviction.
Write a feature article about the Pac-12's dominance in college athletics.
Create an informed consent document for a research study.
Write a film review of Top Gun: Maverick.
Write a critical analysis of the movie Prometheus, exploring its themes.

Instruction

Score: 4 ✓

Compile a comprehensive guide to natural remedies for treating yeast infections in women.
Write a spiritual reflection on the limitations of human capacity.
Write a comprehensive guide about how doctors inform patients about cancer diagnosis.
Write a sports article about a football team's creative adjustments due to injuries.
Write a comprehensive guide for international students on pursuing MBA program in the UK.

Instruction

Score: 5 ✓

Write a comprehensive guide to understanding the different types of real estate.
Develop a guide for starting a meditation habit.
Write a guide on securing valuables and property at home.
Develop a guide on leveraging social media stories for business growth.
Write an article about the mental health benefits of owning a pet.

Figure 16: Examples of instructions at different score levels (1-5), where each score level is illustrated with five representative cases. Score 1 represents the lowest quality while score 5 represents the highest quality.