Cross-Relational Preference Learning for Better LLM Instruction Following

Runsheng Li[1,2], **Kai Sun**[1,2*], **Bin Shi**[1,2] and **Bo Dong**[2,3]
[1]School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
[2]Shaanxi Provincial Key Laboratory of Big Data Knowledge Engineering, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
[3]School of Distance Education, Xi'an Jiaotong University, Xi'an, Shaanxi 710049, China
lirunsheng@stu.xjtu.edu.cn, {sunkai, shibin, dong.bo}@xjtu.edu.cn

Abstract

Large Language Models (LLMs) still exhibit limited capability in following complex instructions. While existing approaches often rely on preference learning to enhance this ability, they typically overlook the relationships between the permissible response spaces of different instructions, which restricts a model to align with subtle and diverse constraint variations. To address this, we propose Cross-Relational Preference Learning (CRPL), a novel framework for constructing preference data that explicitly models inter-instruction relationships through two key techniques: *Cross-Relationship Perturbation* and *Cross-Region Pair Sampling*. This enables the generation of more diverse preference data that captures a wide spectrum of constraint variations. Additionally, we introduce an atomic constraint-based verification mechanism to rigorously assess response satisfaction, ensuring high-quality preference pair construction. Extensive experiments across multiple preference learning methods (e.g., DPO, KTO), LLM backbones and four instruction-following benchmarks demonstrate that our approach achieves substantial improvements over prior baselines and exhibits strong generalization.

1 Introduction

Instruction following is a core capability of large language models (LLMs), requiring precise understanding of user instructions and the generation of high-quality responses that adhere to instruction constraints [Lou *et al.*, 2024; Moon *et al.*, 2025]. Although advanced LLMs [Grattafiori *et al.*, 2024; DeepSeek-AI *et al.*, 2025; Yang *et al.*, 2025] excel at handling simple instructions, their performance remains limited when faced with complex instructions involving multiple constraints [Pyatkin *et al.*, 2025; Pham *et al.*, 2024].

To enhance LLM performance on complex instructions, existing methods employ reinforcement learning to align outputs with constraints [He *et al.*, 2024; Ren *et al.*, 2025]. A central challenge lies in constructing high-quality positive and negative samples for preference optimization. Early approaches focused on instruction-level optimization, synthesizing complex instructions and using verifiers to perform rejection sampling—where passing and failing responses form preference pairs [An *et al.*, 2025; Dong *et al.*, 2025].

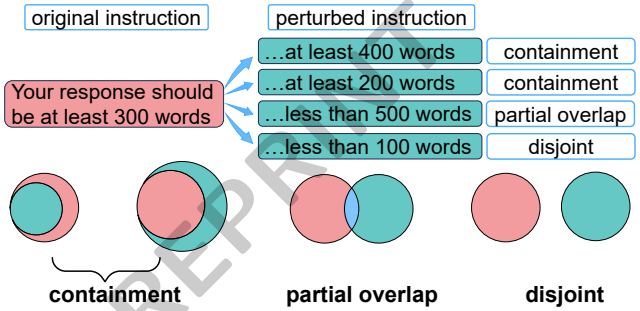

Figure 1: Relationships between the permissible response spaces of an original instruction and its perturbed variant. Colors denote: red — original instruction's response space; green — perturbed instruction's response space; blue — intersection of the two spaces.

Recent works have shifted toward constraint-level optimization, aiming to enhance the model's understanding of individual constraints within instructions. Specifically, several methods generate positive responses by correcting only the violated portions of negative responses, thereby refining the model's comprehension of specific constraints [Cheng *et al.*, 2025; Jiang *et al.*, 2025]. Alternatively, some methods introduce random perturbations, such as deletion, modification, or addition, to specific constraints in the instructions. The model then learns from the differences between responses generated before and after perturbation [Zhang *et al.*, 2025; Huang *et al.*, 2025].

Despite promising results, existing approaches suffer from a critical limitation: they fail to adequately account for the relationships between the permissible response spaces of different instructions when constructing preference pairs. We classify these relationships into three fundamental types: containment, partial overlap, and disjoint, as illustrated in Figure 1. Existing methods either neglect these relationships [Sun *et al.*, 2024; Dong *et al.*, 2025] or focus solely on the disjoint case [Huang *et al.*, 2025; Zhang *et al.*, 2025]. Consequently, the resulting preference data lack diversity, restricting the

*Corresponding author.

model's ability to learn from comprehensive variants of constraints. To address this limitation, we propose that training should incorporate more diverse preference pairs that explicitly capture these inter-space relationships, analogous to how humans leverage relational concepts such as similarity and opposition for comparative learning.

Based on these insights, we propose Cross-Relational Preference Learning (CRPL), a novel framework to enhance LLMs' adherence to intricate instructions. CRPL follows a perturbation paradigm, but it explicitly models the relationships between the response spaces of the original and perturbed instructions. This is achieved through two key designs:

(1) **Cross-Relationship Perturbation:** When modifying a specific constraint, we steer the perturbation direction according to a target relationship. For instance, to construct a containment relationship, we instruct the model that "*the perturbed valid-response set must be a proper superset of the original valid-response set*". This approach enables the model to generate comprehensive constraint variants that span all three relationship types.

(2) **Cross-Region Pair Sampling:** For a given relationship, we ensure that the sampled positive responses collectively cover all pertinent regions within the combined response space of the original and perturbed instructions. As illustrated for the partial-overlap relationship in Figure 1, the positive responses for the original and perturbed instructions span the disjoint regions (red and green) as well as their intersection (blue). This ensures that the constructed positive-negative pairs capture the full spectrum of the relationship, thereby providing diverse and representative data for subsequent preference optimization.

To ensure that positive responses are strictly drawn from specific regions of the combined response space, we introduce an atomic constraint-based verification mechanism. This mechanism constructs and applies a separate verification function for each atomic constraint, providing a rigorous guarantee on the quality of the resulting preference pairs. Unlike existing methods, they typically rely on an LLM to judge the entire instruction-response pair or to generate a single verification function (e.g., as in UltraIF [An *et al.*, 2025] and IOPO [Zhang *et al.*, 2025]), which often fail to adequately verify all constraints.

Our contributions can be summarized as follows:

- We propose Cross-Relational Preference Learning (CRPL), a novel framework that enhances the diversity of preference pairs by explicitly modeling the relationships between response spaces.
- We introduce an atomic constraint-based verification mechanism to rigorously assess whether sampled responses satisfy a given instruction, thereby ensuring the construction of high-quality preference data.
- Through extensive experiments across multiple preference learning methods (e.g., DPO, KTO), diverse LLM backbones and four instruction-following benchmarks, we demonstrate that our framework consistently improves instruction-following performance and exhibits strong generalization.

2 Related Work

2.1 Preference Learning

Preference learning is an important paradigm for enhancing the instruction-following capabilities of LLMs. Early work typically collected human preference data to train reward models and then optimized LLMs using reinforcement learning such as PPO [Schulman *et al.*, 2017] to ensure adherence to human preferences [Ouyang *et al.*, 2022]. Subsequently, preference learning methods such as DPO [Rafailov *et al.*, 2023], KTO [Ethayarajh *et al.*, 2024], and SimPO [Meng *et al.*, 2024] have been proposed to directly optimize the model's output distribution to better align with human preferences, without explicitly training a reward model.

Recently, some works have adapted the DPO algorithm to enhance models' ability to discern fine-grained distinctions between instructions. For example, [Jiang *et al.*, 2025] and [Huang *et al.*, 2025] propose token-level preference optimization methods to achieve fine-grained instruction alignment. IOPO [Zhang *et al.*, 2025] proposes an alignment method that simultaneously considers preferences over both inputs and outputs. In addition, some methods employ an iterative DPO training strategy to achieve continuous improvement in instruction-following capabilities [Cheng *et al.*, 2025; An *et al.*, 2025]. More recently, several works [Pyatkin *et al.*, 2025; Lambert *et al.*, 2025; Peng *et al.*, 2025] have leveraged Reinforcement Learning with Verifiable Rewards (RLVR) to enhance models' ability to follow complex instructions by utilizing verifiable reward signas.

2.2 Preference Data Construction for Instruction-Following

Constructing high-quality preference data is critical for preference learning [Lou *et al.*, 2024]. A simple yet effective approach is rejection sampling [Yuan *et al.*, 2023; Grattafiori *et al.*, 2024]. For instance, ULTRAIF [An *et al.*, 2025] trains an instruction composer to iteratively synthesize complex constrained instructions and evaluate responses to obtain preference data. AutoIF [Dong *et al.*, 2025] leverages LLMs to generate instructions and code to verify responses.

Alternatively, some methods construct preference data by correcting negative responses to obtain corresponding positive responses. For example, From Complex to Simple (FCS) [He *et al.*, 2024] uses a student model to generate initial responses, which are then corrected by a teacher model to form preference pairs. SPAR [Cheng *et al.*, 2025] employs tree-search-based stepwise self-correction on instruction-violating responses to produce highly comparable preference pairs. BMC [Jiang *et al.*, 2025] generates highly correlated preference pairs by applying minimal modifications to negative responses. Other methods generate contrasting responses by perturbing the original instructions. IOPO [Zhang *et al.*, 2025] perturbs certain constraints in the original instruction to induce responses that oppose the original constraints, forming preference pairs. MUSC [Huang *et al.*, 2025] decomposes each complex instruction into atomic constraints and randomly removes some constraints to create negative instructions, with responses from the original and negative instructions forming preference pairs.

Our work contributes to the construction of high-quality preference data but differs from prior approaches by explicitly modeling the relationships between the response spaces of original and perturbed instructions. This approach enhances the diversity of preference pairs, enabling the model to learn from comprehensive constraint variations and thereby significantly improving its instruction-following capability.

3 Problem Definition

Complex Instruction Following is a conditional text generation problem subject to a set of specific constraints.

Notation. Let $\mathcal{X}$ denote the space of complex instructions. Given an input instruction $x_j \in \mathcal{X}$, we posit that x_j entails a set of distinct constraints $\mathcal{C}_j = \{c_1, c_2, \ldots, c_m\}$, where m is the number of constraints in x. Each constraint c_i represents a specific requirement (e.g., length limit, keyword inclusion, or formatting style) that must be satisfied.

Task Formulation. The goal is to generate a response y_j that follows all constraints in $\mathcal{C}_j$. Formally, we aim to learn a model parameterized by θ that approximates:

$$\begin{aligned} \hat{y} &= \arg\max_{y} P_\theta(y \mid x) \\ \text{s.t.} \quad & \mathbb{V}(y, c_i) = 1, \forall c_i \in \mathcal{C}_x \end{aligned} \tag{1}$$

where $\mathbb{V}(\cdot,\cdot)$ is a verification function such that $\mathbb{V}(y, c_i) = 1$ if the response y satisfies constraint c_i, and 0 otherwise.

4 Method

Our framework consists of two key steps: Cross-Relationship Perturbation and Cross-Region Pair Sampling. First, we guide a strong LLM to generate perturbations for a given instruction across three predefined relationship types (i.e., containment, partial overlap or disjoint). Second, using both the original and perturbed instructions, we perform Cross-Region Pair Sampling to derive a comprehensive set of preference pairs, supported by an atomic constraint-based verification. This ensures broad coverage of all relevant regions within the combined response space of the original and perturbed instructions. Figure 2 illustrates an overview of our framework.

4.1 Cross-Relationship Perturbation

Given an instruction, we first utilize an advanced LLM to decompose it into its constituent constraints. Our focus is specifically on hard constraints that can be verified through executable code. This decomposition yields a constraint set $\mathcal{C} = \{c_1, ..., c_m\}$, where m denotes the number of target constraints.

Next, for each instruction x, we iterate over its constraint set $\mathcal{C}$ and the three predefined relationships to generate perturbations. For example, for the containment relationship, we obtain the set $\mathcal{P}_x^{\text{con}} = \{\hat{x}_{c_1}, ..., \hat{x}_{c_m}\}$, where each $\hat{x}_{c_i}$ is a variant of x with constraint c_i perturbed such that the response spaces of $\hat{x}_{c_i}$ and x exhibit a containment relationship. This procedure is applied to all three relationships, resulting in three distinct perturbed instruction sets: $\mathcal{P}_x^{\text{con}}$, $\mathcal{P}_x^{\text{par}}$ and $\mathcal{P}_x^{\text{dis}}$, corresponding to containment, partial overlap and disjoint, respectively. Formally, the Cross-Relationship Perturbation process is described in Algorithm 1.

Algorithm 1 Cross-Relationship Perturbation

Input: Complex instruction set $\mathcal{X}$
Output: Perturbed instruction set $\mathcal{P}_x^{\text{con}}, \mathcal{P}_x^{\text{par}}, \mathcal{P}_x^{\text{dis}}$.

```
Let P_x^con = P_x^par = P_x^dis = ∅.
for x_j in X do
    C_j ← Decompose(x_j)
end for
for x_j in X do
    for r in {con, par, dis} do
        P_{x_j}^r = ∅
        for c_i in C_j do
            ĉ_i, x̂_{c_i} ← Perturb(c_i, x_j)
            P_{x_j}^r = P_{x_j}^r ∪ x̂_{c_i}
        end for
        P_x^r = P_x^r ∪ P_{x_j}^r
    end for
end for
return P_x^con, P_x^par, P_x^dis
```

4.2 Cross-Region Pair Sampling

The combined response space of the original instruction x and its perturbed variant $\hat{x}_{c_i}$ can be partitioned into two distinct regions, determined by which instructions a given response satisfies: (1) **Both-Satisfying**, where responses satisfy both the original and the perturbed instruction; (2) **Single-Satisfying**, where responses satisfy only one of the two instructions.

To ensure preference data captures the full spectrum of a relationship, we perform systematic sampling across these regions. We illustrate this process using the partial overlap relationship as an example. Here, the combined response space divides into three exclusive sub-regions: (1) responses satisfying only the original instruction; (2) responses satisfying only the perturbed instruction; (3) responses satisfying both instructions. This sampling process is formalized as follows:

$$\begin{aligned} \mathcal{O}^{\text{original}} = &\{(x, y^+, y^-) \,|\, y^+ \in \mathcal{S}_1 \,\&\, y^- \in \mathcal{S}_2\} \\ &\cup \{(x, y^+, y^-) \,|\, y^+ \in \mathcal{S}_3 \,\&\, y^- \in \mathcal{S}_2\} \end{aligned} \tag{2}$$

$$\begin{aligned} \mathcal{O}^{\text{perturbed}} = &\{(\hat{x}_{c_i}, y^+, y^-) \,|\, y^+ \in \mathcal{S}_2 \,\&\, y^- \in \mathcal{S}_1\} \\ &\cup \{(\hat{x}_{c_i}, y^+, y^-) \,|\, y^+ \in \mathcal{S}_3 \,\&\, y^- \in \mathcal{S}_1\} \end{aligned} \tag{3}$$

where $\mathcal{S}_1$, $\mathcal{S}_2$ and $\mathcal{S}_3$ denote the response sets corresponding to the three sub-regions defined above; $\mathcal{O}^{\text{original}}$ and $\mathcal{O}^{\text{perturbed}}$ represent the sets of preference pairs for the original instruction and and the perturbed instruction, respectively.

Overall, the preference pair composition adopts a contrastive perspective to enhance learning efficiency [Yan *et al.*, 2024; Zhang *et al.*, 2025]. When a positive response is drawn from an exclusive region (e.g., $\mathcal{S}_1$ or $\mathcal{S}_2$), the corresponding negative is sampled from the opposite exclusive region. This cross-region pairing provides strong contrastive signals, helping the model more effectively distinguish between correct and incorrect constraint satisfaction.

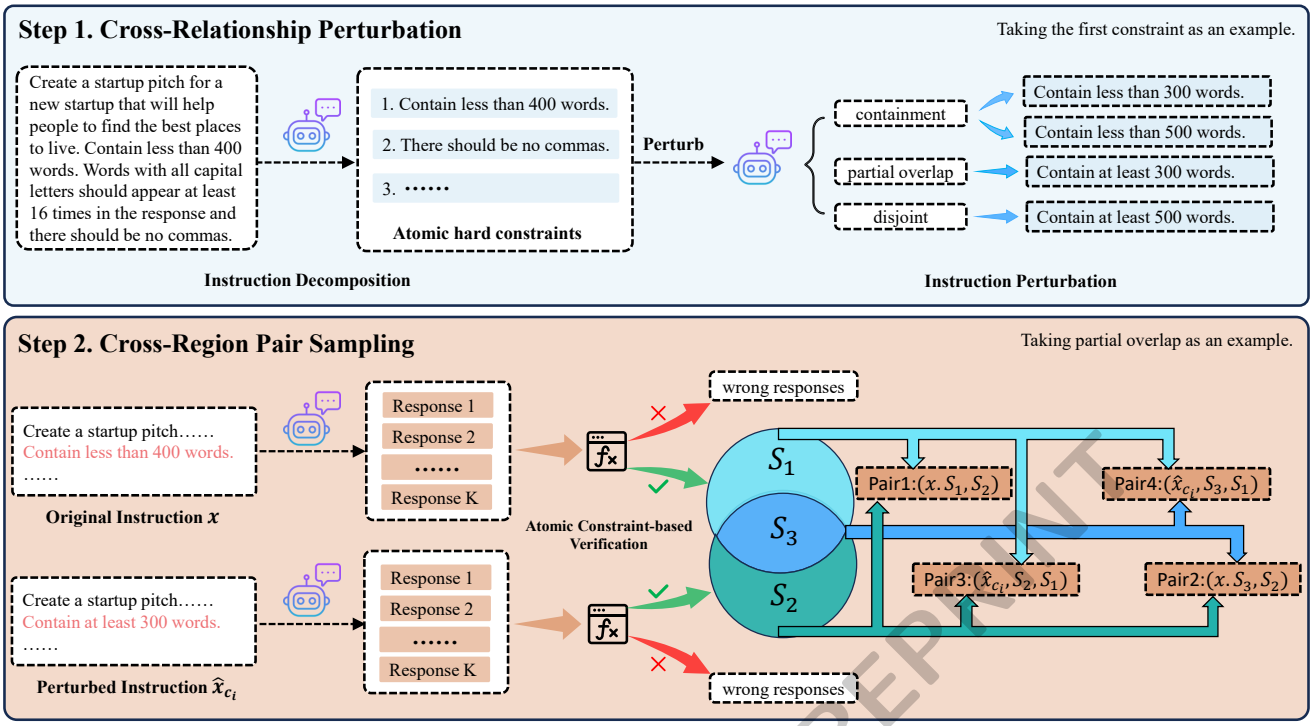

Figure 2: An overview of our Cross-Relational Preference Learning (CRPL) framework. (1) Cross-Relationship Perturbation: We decompose the original instruction into atomic constraints and generate perturbed instructions that exhibit specific response-space relationships (containment, partial overlap, or disjoint) with the original one. (2) Cross-Region Pair Sampling: Guarded by an atomic constraint-based verification mechanism, we sample positive and negative responses from distinct sub-regions of the combined response space to construct diverse, high-quality preference pairs.

Atomic Constraint-based Verification

To ensure that positive responses should be drawn strictly from designated regions within the combined response space, we introduce an atomic constraint-based verification mechanism. Specifically, we prompt an LLM to generate a dedicated verification function for each atomic constraint. To guarantee reliability, each function must execute successfully and exceed 80% accuracy on six LLM-generated test cases; otherwise, it is regenerated. This yields a set of verification functions $V = \{v_1, v_2, \ldots, v_m\}$, where each Boolean function v_i evaluates if a response satisfies constraint c_i. Formally, each sampled response y_k is verified against its corresponding constraints by computing:

$$a_k = \prod_{v_i \in V_j} v_i(y_k) \tag{4}$$

Responses where $a_k = 1$ are classified as positive. This approach provides a fine-grained and more strict evaluation of each response against individual constraints.

4.3 Training Objectives

Our framework serves as a general-purpose generator of preference pairs and is compatible with a range of preference optimization methods. To demonstrate this flexibility, we apply the sampled pairs to DPO [Rafailov *et al.*, 2023], KTO [Ethayarajh *et al.*, 2024], and an online variant of DPO [Cheng *et al.*, 2025].

5 Experiments

5.1 Setup

Datasets and Backbones We evaluate our method on two established instruction-following datasets: IFEval [Zhou *et al.*, 2023] and IFBench [Pyatkin *et al.*, 2025]. Following the evaluation settings of prior works [Dong *et al.*, 2025; Pyatkin *et al.*, 2025], for IFEval, we use 433 instructions for training and hold out the remaining 108 as a test set. For IFBench, we randomly sample 2000 instructions from its training split as our training set and use its official test set of 294 instructions for final evaluation. Additionally, we include 200 perturbed instructions generated by our Cross-Relationship Perturbation method as a separate testing set, which we name Perturbed-IF. Furthermore, to assess generalization beyond code-verifiable constraints, we also evaluate our method on the FollowBench dataset [Jiang *et al.*, 2024]. We conduct our experiments using two open-source LLMs as backbones: Qwen2.5-7B-Instruct [1] and Llama-3.1-8B-Instruct [2].

Evaluation Protocol Consider a test set $\mathcal{D} = \{x_j\}_{j=1}^{N}$ containing N instructions. Each instruction x_j is associated with a specific set of constraints $\mathcal{C}_j$. For each instruction x_j, we denote the model's generated response as y_i. We evaluate performance using the following two accuracy metrics:

[1] https://huggingface.co/Qwen/Qwen2.5-7B-Instruct
[2] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

Method	Training size	IFEval		IFBench		Perturbed-IF		FollowBench		Avg.
		Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}	
Backbone: Qwen2.5-7B-Instruct										
Qwen2.5-7B-Instruct	-	71.2	78.7	26.5	29.0	42.5	65.0	50.9	58.2	52.8
Training strategy: DPO										
Rejection Sampling	12k	75.7	81.0	24.8	26.3	51.0	66.6	46.2	55.6	53.4
From Complex to Simple	12k	74.1	81.6	27.6	29.9	45.5	64.8	51.6	58.5	54.2
AutoIF	12k	74.1	81.0	31.3	35.2	43.5	66.8	52.5	59.6	55.5
$Ours_{IFEval}$	12k	**90.7**	**94.3**	32.0	34.6	**56.5**	**73.5**	55.4	61.8	**62.4**
$Ours_{IFBench}$	12k	82.2	86.6	**34.9**	**36.2**	54.0	72.8	**57.5**	**63.2**	60.9
Training strategy: KTO										
Rejection Sampling	12k	70.4	75.9	23.5	25.7	44.5	62.4	45.5	55.1	50.4
From Complex to Simple	12k	73.1	80.5	28.2	30.7	43.5	65.0	49.4	56.6	53.4
AutoIF	12k	72.2	80.5	29.9	31.6	42.5	65.0	52.5	59.8	54.3
$Ours_{IFEval}$	12k	**82.4**	**87.4**	**30.6**	31.9	**52.0**	**71.5**	**54.7**	**61.8**	**59.0**
$Ours_{IFBench}$	12k	78.7	83.6	30.4	**32.6**	49.5	68.8	51.5	58.1	56.7
Training strategy: Online DPO										
UltraIF	12k	67.1	73.9	24.4	26.2	43.5	60.8	45.1	54.9	49.5
SPAR	12k	72.2	80.3	**33.7**	**36.0**	42.0	65.3	29.6	40.1	49.9
$Ours_{IFEval}$	12k	**76.9**	**82.9**	33.3	34.6	**49.5**	**70.1**	**53.2**	59.5	**57.5**
$Ours_{IFBench}$	12k	75.0	81.8	28.5	30.1	48.0	68.6	53.1	**59.6**	55.6
Backbone: Llama3.1-8B-Instruct										
Llama3.1-8B-Instruct	-	80.6	85.8	22.5	24.4	49.0	70.0	51.5	58.4	55.3
Training strategy: DPO										
Rejection Sampling	12k	81.5	87.6	23.8	26.4	57.5	75.7	46.8	53.2	56.6
From Complex to Simple	12k	76.9	83.3	33.7	35.8	53.0	73.9	50.8	58.1	58.2
AutoIF	12k	84.3	89.7	35.7	**38.8**	55.0	75.2	49.3	57.8	60.7
$Ours_{IFEval}$	12k	88.0	92.5	**36.4**	37.9	65.0	81.6	55.1	60.4	64.6
$Ours_{IFBench}$	12k	**88.7**	**92.7**	34.4	37.5	**66.5**	**82.1**	**57.2**	**62.2**	**65.2**
Training strategy: KTO										
Rejection Sampling	12k	79.9	85.3	25.9	28.1	54.0	75.4	50.9	57.3	57.1
From Complex to Simple	12k	78.7	85.6	27.6	29.6	49.5	71.2	51.3	58.6	56.5
AutoIF	12k	80.6	86.8	31.7	**34.9**	61.5	78.7	54.6	**62.0**	61.4
$Ours_{IFEval}$	12k	**86.6**	**91.5**	**32.6**	34.6	**64.5**	**80.8**	54.7	61.4	**63.3**
$Ours_{IFBench}$	12k	**86.6**	91.4	31.1	34.1	56.5	77.2	**56.2**	61.6	61.8
Training strategy: Online DPO										
UltraIF	12k	83.8	88.2	26.0	28.3	**54.5**	74.6	47.8	55.0	57.3
SPAR	12k	79.9	86.8	**27.7**	**29.7**	52.0	73.7	43.7	52.3	55.7
$Ours_{IFEval}$	12k	**85.8**	**90.5**	27.6	28.3	54.0	**77.0**	52.5	**58.7**	**59.3**
$Ours_{IFBench}$	12k	84.7	89.2	26.0	27.3	52.0	74.3	**52.6**	58.3	58.1
Recent baselines trained with RLVR on R1-Distill-Qwen-7B and TULU-3-8B										
VerIF (R1-Distill-Qwen-7B)	22k	73.6	82.9	18.8	21.0	26.5	47.8	47.6	55.1	46.7
VerIF (TULU-3-8B)	22k	85.0	89.8	23.6	26.3	55.5	77.7	53.4	60.4	59.0

Table 1: Performance comparison of our method against prior baselines across the four test sets. For each backbone and training strategy, the best and second-best accuracy scores are highlighted in **bold** and underlined, respectively. $Ours_{IFEval}$ and $Ours_{IFBench}$ indicate models trained on the IFEval and IFBench training sets, respectively.

Model	Original		Containment-1		Containment-2		Partial Overlap		Disjoint	
	Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}	Acc_{ins}	Acc_{con}
Qwen2.5-7B-Instruct	71.2	78.7	53.7	71.0	58.8	73.2	50.0	67.9	51.2	70.1
Rejection Sampling	75.7	81.0	57.5	75.0	60.0	75.6	55.0	72.8	62.5	73.7
From Complex to Simple	84.1	81.6	62.5	76.7	60.0	72.6	50.0	69.1	53.7	70.1
AutoIF	84.1	81.0	51.2	72.7	57.5	73.2	50.0	70.4	51.2	71.9
Ours	90.7	94.3	68.8	80.7	67.5	80.4	62.5	79.6	68.8	80.2

Table 2: Performance on original instructions and four types of perturbed instructions. Containment-1 and Containment-2 represent two subtypes of containment relationships, namely containing and contained.

- **Constraint-level Accuracy (Acc_{con})** This metric computes the proportion of satisfied constraints relative to the total number of constraints across all instructions:

$$\text{Acc}_{\text{con}} = \frac{\sum_{j=1}^{N} \sum_{c \in \mathcal{C}_j} \mathbb{V}(y_j, c)}{\sum_{j=1}^{N} |\mathcal{C}_j|} \tag{5}$$

  where $\mathbb{V}(\cdot, \cdot)$ denotes the verification function.

- **Instruction-level Accuracy (Acc_{ins})** This metric computes the proportion of instructions for which all constraints are satisfied across the test set:

$$\text{Acc}_{\text{ins}} = \frac{1}{N} \sum_{j=1}^{N} \left(\prod_{c \in \mathcal{C}_j} \mathbb{V}(y_j, c) \right) \tag{6}$$

  where $\prod$ is a logical AND operation, equaling 1 only if the response satisfies all constraints for instruction x_j.

Baselines We categorize the preference optimization baselines into two types: online and offline. Offline baselines include **Rejection Sampling** [Yuan *et al.*, 2023], which constructs preference pairs by sampling and verifying multiple responses per instruction; **From Complex to Simple** [He *et al.*, 2024], which forms preference pairs through teacher correction of student outputs; and **Autoif** [Dong *et al.*, 2025], which employs LLMs to automatically generate instructions and verification code. We compare these baselines with our method across both DPO and KTO training strategies.

Online baselines include **UltraIF** [An *et al.*, 2025], which trains an instruction composer to iteratively synthesize constrained complex instructions, followed by sampling and evaluation for online DPO training; and **SPAR** [Cheng *et al.*, 2025], which constructs preference data for online DPO by performing tree-search–based self-corrections on instruction-violating responses. We compare these baselines with our method using the online DPO framework adopted in SPAR.

We also compare with recent baselines that employ different training strategies. For instance, **VerIF** [Peng *et al.*, 2025] combines rule-based code verification with LLM-based validation to perform RLVR.

Implementation Details We employ DeepSeek v3.1 for instruction decomposition, perturbation, and the generation of verification functions. For offline training, DeepSeek v3.1 is also used for response sampling. In online training, responses are sampled from the optimized model at each iteration. All experiments are run on a single A100 GPU using the OpenRLHF framework [Hu *et al.*, 2025].

5.2 Main Results

The performance comparison is presented in Table 1. We highlight three key observations:

Performance on Offline Training Our framework significantly enhances instruction-following performance across both DPO and KTO training methods, as well as all four test sets. For instance, under the DPO training strategy with the Qwen2.5-7B-Instruct backbone, our models trained on either IFEval or IFBench consistently outperform prior baselines. Specifically, $\text{Our}_{\text{IFEval}}$ surpasses AutoIF by 6.9% in average accuracy across the four datasets. Similar performance gains are observed with the KTO training strategy. These results demonstrate the superiority and generalization of our method.

Method	Acc_{ins}	Acc_{con}
Ours	90.7	94.3
w/o Containment-1	85.9	90.7
w/o Containment-2	84.7	88.8
w/o Partial Overlap	84.3	88.8
w/o Disjoint	82.9	87.2

Table 3: Ablated results on different relationship types.

Method	Acc_{ins}	Acc_{con}
Ours	81.3	85.6
w/o Pair1	77.1	82.5
w/o Pair2	77.3	83.1
w/o Pair3	78.5	83.5
w/o Pair4	79.2	84.2

Table 4: Ablated results on cross-region pair sampling.

Performance on Online Training Under online training, without supervision from advanced LLMs, the performance of prior baselines is limited on the Qwen2.5-7B-Instruct backbone. Both UltraIF and SPAR exhibit lower average accuracy compared to the pretrained Qwen2.5-7B-Instruct. Nevertheless, our method maintains significant performance improvement in this setting. This can be contributed to our method's diverse sampling and atomic verification, which enable the acquisition of higher-quality preference pairs.

Performance across Different Backbones Our method demonstrates performance gains across both LLM backbones, further validating the generalization of our approach. We also conducted training on Qwen2.5-14B-Instruct [3] and Qwen3-8B [4], and the results are provided in the Appendix.

5.3 Further Analysis

Due to limited computational resources, we conduct further analysis only on Qwen2.5-7B-Instruct using offline training strategies, with IFEval as training set.

Performance on Different Data Sizes
To evaluate the quality of preference pairs generated by different sampling methods, we compare our method with prior baselines under both DPO and KTO as the training data size grows (Figure 3). As training data size increases, our method demonstrates a clear upward performance trend, while other methods show limited gains. These results indicate that our improvements stem not only from increased data quantity but also from higher data quality and more informative preference signals.

Performance on Different Relationship Types
A model with strong instruction-following ability should adhere consistently to a constraint across its variants. We compare our method with baselines—Rejection Sampling, From

[3] https://huggingface.co/Qwen/Qwen2.5-14B-Instruct
[4] https://huggingface.co/Qwen/Qwen3-8B

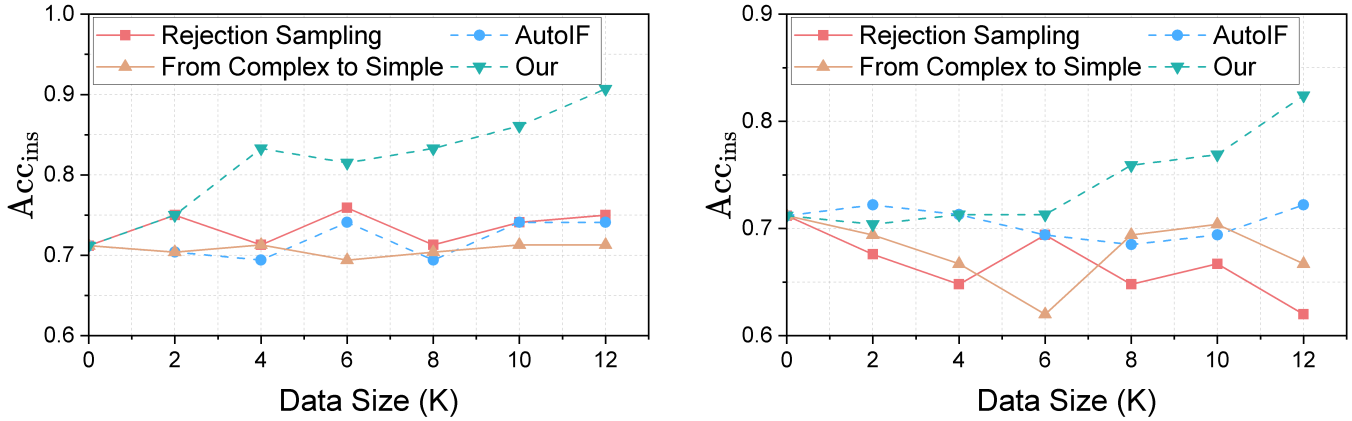

Figure 3: Performance scaling with training data size for DPO (left) and KTO (right) across different methods.

Method	TP	FP	TF	NF	Acc_{ins}	Acc_{con}
ACV (ours)	66	0	27	15	85.4	89.2
ILV	47	0	27	34	84.0	88.7
LAJ	52	4	23	29	80.8	85.8

Table 5: Performance comparison for different verification methods.

Complex to Simple, and AutoIF—on the original instructions and four perturbed types (Table 2).

Baselines improve performance on the original instructions but show limited gains on perturbed variants. For instance, From Complex to Simple improves only on "Containment-1", while other perturbed types remain close to the pre-trained model's performance. Without explicitly accounting for response-space relationships, these methods struggle to capture comprehensive constraint variations. Rejection Sampling improves more consistently across perturbation types, likely because its random sampling may incidentally cover such variations. In contrast, our method achieves significant improvement on both the original instructions and all perturbed types, demonstrating strong generalization. This underscores the value of training on diverse perturbed instructions for boosting instruction-following capability.

Ablation Study

Relationship Types We evaluate the contribution of each relationship type by removing individual relationships from the perturbation process. Table 3 presents the results on the IFEval dataset. We use w/o to denote the variant without a specific type. Compared to using the full set of relationships, each removal results in a performance degradation, indicating that including all relationship types during sampling is crucial. Specifically, the "Disjoint" relationship type plays the most significant role that provides strong contrastive signals to enhance training.

Cross-Region Pair Sampling Taking the "partial overlap" relationship as a case study, we assess the contribution of preference pairs drawn from different sampling regions. Specifically, we remove each of the four pair types in turn and report the results in Table 4. Performance degrades when any single pair type is excluded, demonstrating the importance of constructing preference pairs across these regions to ensure sampling diversity.

Verification Methods To evaluate the effectiveness of our Atomic Constraint-based Verification (ACV), we compare it with two alternative verification strategies: (1) using an LLM as a judge to directly assess whether a response follows the instruction (LLM-as-Judge, LAJ), and (2) generating a single verification function for an instruction (Instruction-level Verification, ILV).

Specifically, we first collect an LLM's responses to the IFEval test instructions and evaluate them using all three methods. Following the official IFEval validation protocol, we compute the numbers of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN) for each method, as reported in Table 5. ACV achieves the lowest misjudgment rate and yields zero false positives, indicating its superior effectiveness in evaluating a response's adherence to an instruction.

We then apply ILV and LAJ to the IFEval training set to construct preference data and train models on the resulting datasets. ACV achieves the highest performance, further demonstrating that our method more effectively ensure the quality of preference pairs.

6 Conclusion

We propose Cross-Relational Preference Learning (CRPL), a novel framework to enhance LLM instruction following. CRPL explicitly models fine-grained response-space relationships between different instructions, enabling the generation of preference data that captures a wide spectrum of constraint variations. The processes are rigorously guaranteed by an atomic constraint-based verification mechanism, which ensures the quality of the sampled pairs. Through experiments across diverse training methods and LLM backbones, our framework demonstrates strong performance on the instruction-following benchmarks.

Acknowledgments

This research was partially supported by the Key Research and Development Project in Shaanxi Province No. 2024PT-ZCK-89, the National Natural Science Foundation of China No. 62406242, 62476215, 62302380, 62037001, 62137002 and 62192781, Project of China Knowledge Centre for Engineering Science and Technology.

References

[An *et al.*, 2025] Kaikai An, Li Sheng, Ganqu Cui, Shuzheng Si, Ning Ding, Yu Cheng, Baobao Chang. UltraIF: Advancing instruction following from the wild. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 18722–18737, Suzhou, China, November 2025. Association for Computational Linguistics.

[Cheng *et al.*, 2025] Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, Minlie Huang. SPar: Self-play with tree-search refinement to improve instruction-following in large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[DeepSeek-AI *et al.*, 2025] DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.

[Dong *et al.*, 2025] Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Ethayarajh *et al.*, 2024] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, Douwe Kiela. Model alignment as prospect theoretic optimization. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[Grattafiori *et al.*, 2024] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models, 2024.

[He *et al.*, 2024] Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, Yanghua Xiao. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10864–10882, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[Hu *et al.*, 2022] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

[Hu *et al.*, 2025] Jian Hu, Xibin Wu, Wei Shen, Jason Klein Liu, Weixun Wang, Songlin Jiang, Haoran Wang, Hao Chen, Bin Chen, Wenkai Fang, et al. OpenRLHF: A ray-based easy-to-use, scalable and high-performance RLHF framework. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 656–666, Suzhou, China, November 2025. Association for Computational Linguistics.

[Huang *et al.*, 2025] Hui Huang, Jiaheng Liu, Yancheng He, Shilong Li, Bing Xu, Conghui Zhu, Muyun Yang, Tiejun Zhao. MuSC: Improving complex instruction following with multi-granularity self-contrastive training. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10667–10686, Vienna, Austria, July 2025. Association for Computational Linguistics.

[Jiang *et al.*, 2024] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, Wei Wang. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[Jiang *et al.*, 2025] Yuxin Jiang, Bo Huang, Yufei Wang, Xingshan Zeng, Liangyou Li, Yasheng Wang, Xin Jiang, Lifeng Shang, Ruiming Tang, Wei Wang. Bridging and modeling correlations in pairwise data for direct preference optimization. In *The Thirteenth International Conference on Learning Representations*, 2025.

[Lambert *et al.*, 2025] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James Validad Miranda, Alisa Liu, Nouha Dziri, Xinxi Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. In *Second Conference on Language Modeling*, 2025.

[Lou *et al.*, 2024] Renze Lou, Kai Zhang, Wenpeng Yin. Large language model instruction following: A survey of progresses and challenges. *Computational Linguistics*, 50(3):1053–1095, September 2024.

[Meng *et al.*, 2024] Yu Meng, Mengzhou Xia, Danqi Chen. Simpo: simple preference optimization with a reference-free reward. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc.

[Moon *et al.*, 2025] Hyeonseok Moon, Seongtae Hong, Jaehyung Seo, Heuiseok Lim. Metric calculating benchmark: Code-verifiable complicate instruction following benchmark for large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20809–20823, Suzhou, China, November 2025. Association for Computational Linguistics.

[Ouyang *et al.*, 2022] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin,

Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022.

[Peng *et al.*, 2025] Hao Peng, Yunjia Qi, Xiaozhi Wang, Bin Xu, Lei Hou, Juanzi Li. VerIF: Verification engineering for reinforcement learning in instruction following. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30312–30327, Suzhou, China, November 2025. Association for Computational Linguistics.

[Pham *et al.*, 2024] Chau Minh Pham, Simeng Sun, Mohit Iyyer. Suri: Multi-constraint instruction following in long-form text generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1722–1753, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[Pyatkin *et al.*, 2025] Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, Hannaneh Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2025.

[Rafailov *et al.*, 2023] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, Chelsea Finn. Direct preference optimization: your language model is secretly a reward model. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

[Ren *et al.*, 2025] Qingyu Ren, Jie Zeng, Qianyu He, Jiaqing Liang, Yanghua Xiao, Weikang Zhou, Zeye Sun, Fei Yu. Step-by-step mastery: Enhancing soft constraint following ability of large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 19581–19596, Vienna, Austria, July 2025. Association for Computational Linguistics.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov. Proximal policy optimization algorithms, 2017.

[Sun *et al.*, 2024] Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, Ruohui Huang. Conifer: Improving complex constrained instruction-following ability of large language models, 2024.

[Yan *et al.*, 2024] Tianyi Yan, Fei Wang, James Y. Huang, Wenxuan Zhou, Fan Yin, Aram Galstyan, Wenpeng Yin, Muhao Chen. Contrastive instruction tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 10288–10302, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[Yang *et al.*, 2025] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report, 2025.

[Yuan *et al.*, 2023] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023.

[Zhang *et al.*, 2025] Xinghua Zhang, Haiyang Yu, Cheng Fu, Fei Huang, Yongbin Li. IOPO: Empowering LLMs with complex instruction following via input-output preference optimization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22185–22200, Vienna, Austria, July 2025. Association for Computational Linguistics.

[Zhou *et al.*, 2023] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, Le Hou. Instruction-following evaluation for large language models, 2023.